Proceedings of the Conference on

LANGUAGE & TECHNOLOGY

2020

مرکز تحقیقاتِ لسانیات

Center for Language Engineering
Al-Khawarizmi Institute of Computer Science
University of Engineering and Technology
Lahore- Pakistan

CONFERENCE COMMITTEES

Organizing Committee:

Conference Chair	Dr. Sarmad Hussain, CLE, KICS-UET, Lahore, Pakistan
Chair, Technical Committee	Dr. Tania Habib, CLE, KICS-UET, Lahore, Pakistan
Co-Chair, Language Processing Track	Dr. Miriam Butt, University of Konstanz, Germany
Chair , Publication Committee	Dr. Amir Mehmood, KICS-UET, Lahore, Pakistan
Co-Chair, Script Processing Track	Dr. Faisal Shafait, NUST, Islamabad, Pakistan
Chair, Program Committee	Ms. Sana Shams, CLE, KICS-UET, Lahore, Pakistan
Chair, Workshop	Dr. Kashif Javed, UET, Lahore, Pakistan
Chair, Demo	Ms. Qurat-ul-Ain Akram, CLE, KICS-UET, Lahore, Pakistan

Technical Committee:

Dr. Sarmad Hussain	Center for Language Engineering, KICS-UET, Lahore, Pakistan
Dr. Amir Mehmood	Al-Khawarizmi Institute of Computer Science, UET, Lahore, Pakistan
Dr. Adeel Nawab	COMSATS University, Lahore, Pakistan
Dr. Agha Ali Raza	Lahore University of Management Sciences, Lahore, Pakistan
Dr. Annette Hautli	University of Konstanz, Germany
Dr. Barbara Schuppler	Graz University of Technology, Austria
Dr. Faisal Shafait	National University of Sciences & Technology, Islamabad, Pakistan
Dr. Farhat Jabeen	University of Konstanz, Germany
Dr. Hassan Sajjad	Qatar Computing Research Institute, Qatar
Dr. Imran Siddiqui	Bahria University, Pakistan
Dr. Kamran Malik	Punjab University, College of Information Technology, Lahore, Pakistan
Dr. Kashif Javed	University of Engineering & Technology, Lahore, Pakistan
Dr. Miriam Butt	University of Konstanz, Germany
Dr. Muhammad Abid	University of Peshawar, Peshawar, Pakistan
Dr. Muhammet Bastan	Amazon, Palo Alto, California
Dr. Nadir Durrani	Qatar Computing Research Institute, Qatar
Dr. Sheikh Faisal Rashid	University of Engineering & Technology, Lahore, Pakistan
Dr. Tafseer Ahmad Khan	Muhammad Ali Jinnah University, Islamabad
Dr. Tania Habib	University of Engineering & Technology, Lahore, Pakistan
Dr. Usman Ghani	University of Engineering & Technology, Lahore, Pakistan
Dr. Waqas Anwar	COMSATS University, Lahore, Pakistan
Ms. Saba Urooj	University of Engineering & Technology, Lahore, Pakistan
Ms. Sana Shams	Center for Language Engineering, KICS-UET, Lahore, Pakistan

Publication Committee:

Dr. Amir Mehmood	Al-Khawarizmi Institute of Computer Science, UET, Lahore, Pakistan

Mr. Usama Tahir	Al-Khawarizmi Institute of Computer Science, UET, Lahore, Pakistan

Program Committee:

Ms. Sana Shams	Center for Language Engineering, KICS-UET, Lahore, Pakistan
Ms. Kashaf Shahzad	Center for Language Engineering, KICS-UET, Lahore, Pakistan
Ms. Iqra Amjad	Center for Language Engineering, KICS-UET, Lahore, Pakistan
M. Kamran Khan	Center for Language Engineering, KICS-UET, Lahore, Pakistan
Mr. Zaeem Saqif	Center for Language Engineering, KICS-UET, Lahore, Pakistan

Workshop Committee:

Dr. Kashif Javed	University of Engineering & Technology, Lahore, Pakistan

Demo Committee:

Ms. Qurat-ul-Ain Akram	Center for Language Engineering, KICS-UET, Lahore, Pakistan
Ms. Farah Adeeba	Center for Language Engineering, KICS-UET, Lahore, Pakistan
Mr. Zaeem Saqif	Center for Language Engineering, KICS-UET, Lahore, Pakistan

FOREWORD

On behalf of the Organizing Committee, we welcome the authors and participants to the seventh Conference on Language and Technology.

Center for Language Engineering (CLE) at Al-Khawarizmi Institute Computer Science (KICS), UET, Lahore, is pleased to host the Conference on Language and Technology 2020 (CLT20). As the seventh CLT, we aim to help in the advancement of local language technology research in Pakistan and provide a platform for experienced researchers for active engagement and interaction, and also give an opportunity to aspiring students who plan to pursue research in the area of language and technology.

Forty-three papers were submitted for CLT20, of which ten have been accepted for oral presentation and four for poster presentation, through a rigorous review process by the technical committee. The papers cover multiple local languages and a number of areas including linguistics, speech processing and computational aspects of phonetics, phonology, syntax, and semantics. This CLT also presents exciting talks including recent research in speech processing, syntax, machine translation, and script processing. This time, we have introduced a new demo track keeping in mind the recent advances in the area of artificial intelligence and machine learning and their widespread applications in natural language processing. During this session, the academic researchers and industry experts will demonstrate their systems.

On behalf of the Organizing Committee, we would like to express our gratitude and appreciation to all who volunteered to plan and support the conference. We would like to thank the technical committee members for their diligent reviews of the research articles. We would also like to thank the conference sponsors, especially the Higher Education Commission of Pakistan (HEC), Punjab Higher Education Commission (PHEC), University of Konstanz, German Academic Exchange Service (DAAD), International Speech and Communication Association (ISCA), FabIntel and Contegris. We are grateful to the management of Al-Khawarizmi Institute of Computer Science and the University of Engineering and Technology, Lahore, for their unrelenting support to hold the conference.

We wish you all a very fruitful CLT20 and a pleasant stay in Lahore.

Sarmad Hussain
On behalf of the Organizing Committee

TABLE OF CONTENTS

A Multilayered Urdu Treebank

Tafseer Ahmed[1], Toqeer Ehsan[2], Almas Ashraf[3], Mutee u Rahman[4], Sarmad Hussain[2], Miriam Butt[5]

[2]*Centre for Language Engineering Al-Khawarizmi Institute of Computer Science, UET, Lahore*
firstname.lastname@kics.edu.pk

[3]*NEDUET, Karachi*
almasashraf@neduet.edu.pk

[4]*Isra University, Hyderabad*
mutee.rahman@isra.edu.pk

[5]*University of Konstanz, Konstanz, Germany*
miriam.butt@uni-konstanz.de

Abstract

The paper presents the design and construction of a multilayered phrase structure treebank. The treebank consists of three layers for phrases, grammatical functions and semantic roles. A small phrase tagset (consisting of 12 tags) is used as the primary label of the phrase. Phrase label is followed by grammatical function (mainly inspired by lexical functional grammar). It is followed by the semantic role label using propbank roles. 1,300 sentences from CLE Urdu Digest Corpus are annotated using the treebank guideline.

1. Introduction

Treebank is an important linguistic resource for syntax analysis of languages. Creating a treebank involves choosing the theoretical model, creating the annotation guidelines and then annotating the corpus. The annotated corpus is used to create syntactic parsers and other syntax analysis tools.

Urdu is an Indo-Aryan language spoken mainly in Pakistan and India [1]. Urdu and Hindi share common grammar. However, there are differences in script, vocabulary and phonology.

The current work is part of a bigger project introducing intonation in Urdu Text to Speech System. One goal of the project is creation of phrase structure parser for the system. For this reason, a phrase structure Urdu Treebank is planned. The treebank design introduces three different layers of annotation to model phrase structure (constituents), their grammatical functions and semantic roles. This paper presents a description of the treebank creation task.

In subsequent sections, Section 2 presents the important treebanks and major works for Urdu/Hindi treebanks. Section 3 compares different treebank design options to create our treebank. Section 4 describes the design principles and a brief description of different layers of the treebank. Conclusion and Future work is mentioned in Section 5.

2. Literature Review

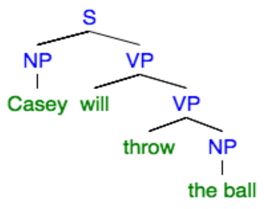
There are two major types of syntactic annotation: phrase structure and dependency structure. The phrase structure analysis of sentence was introduced by Chomsky [2]. The first major treebank, Penn Treebank has phrase structured annotation [3]. The Penn Treebank was inspiration for many treebanks for other languages. Penn Treebank (PTB) has around 25 phrase labels. Figure 1(a) shows a phrase structure of an English sentence annotated using PTB guidelines. Figure 1(b) has its representation in bracketed notation. The bracketed notation is in text format, so it can be processed by computer applications.

As different languages and different treebanks use different set of phrase labels in design, Han et. al. [4] introduced a common tagset after analyzing 25 different treebanks covering 21 languages. They introduced 9 universal phrase labels namely Noun Phrase (NP), Verb Phrase (VP), Adjectival Phrase (AJP), Adverbial Phrase (AVP), Prepositional Phrase (PP), Sentence (S), Conjunction Phrase (CONJP), Coordinated Phrase (COP) and others (X).

The other type of syntactic annotation is dependency structure. Its primary focus in not on the word order or constituency, but it deals with the syntactic relations between the words. A dependency structure along with corresponding phrase structure is presented in figure 1. The most important milestone is the introduction of universal dependencies [5]. There are more than 100 treebanks annotated using universal dependencies [6].

[1] The author was affiliated with *DHA Suffa University, Karachi* when this work was done.

a.



b. (S
(NP Casey)
(VP will (VP throw
(NP the ball))))

c.

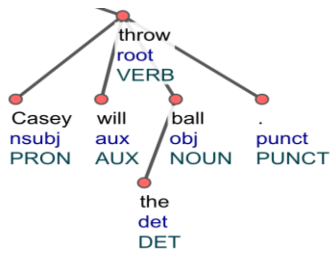


Figure 1: (a) phrase structure, (b) its bracketed notation and (c) dependency structure for the English sentence *Casey will throw the ball*.

For Urdu (and Hindi), there was no freely available treebank at the start of this project. There were earlier works on computational grammar of Urdu (and Hindi). Urdu Pargram implements major parts of Urdu grammar using Lexical Functional Grammar (LFG) framework [7]. An important syntactic structure bank is Hindi Urdu Treebank (HUTB) [8] that has Urdu corpus annotated using Panini style dependencies. The dependency structure can be automatically converted to the phrase structure. An additional layer of dependencies based on LFG's f-structure is also proposed for HUTB [9].

Urdu.Kon.TB [10] is another Urdu treebank that uses a rich feature based pos tagset and a big phrase tagset. A parser was also developed using this treebank of 1400 sentences.

3. Comparison and Design Principles

The previous section presented some important treebanks generally for all languages and specifically for Urdu and Hindi. In this section, we compare the approaches used in these treebanks and find which approach is better for the design of our treebank.

The first question is whether the syntactic bank will have phrase structure or dependency structure representation. The dependency structures have become more popular due to the introduction of universal dependencies in language processing applications. However, our team have a bigger goal of creating a text to speech system and using syntactic information to predict prosody/intonation of the system.

We find that most of the work related to syntax-prosody interaction involves phrase structure models [11],[12] as phrase structure grammar is more commonly used by linguists. Hence, we decided to adopt the classical method of phrase structure treebank. After deciding to create phrase structure treebank, we analyzed the existing treebanks (described in previous section) on the basis of following criteria. This analysis recommends the design principles for our treebank.

1. How is the structure of the tree?
2. What is the granularity of the phrase label?
3. How additional information is encoded?

These criteria are discussed in the following subsections.

3.1 Structure of Tree

There are many ways to construct parse tree corresponding to a given sentence. There are linguistic theories such as X-bar theory that ask for strictly binary trees. HUTB has binary trees because they are inspired by X-bar style work.

Other theories and traditions prefer flatter trees having head and all its dependents on the same level. Penn Treebank and Urdu Pargram have flat structures having head and all the adjacent modifiers on the same level of tree.

The simplicity of annotation scheme to facilitate the annotator is one of our primary design policies. Hence, we prefer the flat structures as they are easy to annotate and many members of treebank community use it for its simplicity.

3.2 Granularity

The second issue is the granularity of the tags. Some schemes e.g. Penn Treebank use more phrase tags (27 for PTB). Multiple tags for the same/similar phrases are used to highlight the difference in structure and/or words used in the phrase. In PTB, most of the phrases have two versions, one for the general usage and the other for the phrases having wh-word. For example, "my books" is an NP and "whose books" or "how many books" are WHNP.

The phrase tagset of Urdu.Kon.TB, in this regard, is inspired by Penn Treebank. It has 26 main tags. Some of these tags have function subtags. The tags NPQ, ADJPQ, QWP and SQ etc, are the tags for question sentences/phrases. Similarly, there are four tags corresponding to the verb complex i.e. VCmain, VP, VPI and VCP.

The other schemes e.g. Multilingual Tagset, HUTB and Urdu Pargram does not differentiate the phrases on the basis of their internal structure or usage. For this reason, PTB has 27 tags and Multilingual Tagset has 12 tags. So, following our design policy of choosing simpler annotation scheme, we prefer smaller tagset approach, and considered Multilingual Tagset as the starting point.

HUTB also uses a small tagset. We did not use some of its tags and the reasons are explained in the discussion of our tagset in section 4.

3.3 Additional Information

Penn Treebank introduced function tags that concatenate additional information e.g. grammatical role etc. to the phrase labels. The function part is attached with the main label by a hyphen. See the example.

```
1. (S (NP-SBJ He)
    (VP left
    (NP-TMP yesterday)))
```

HUTB uses -pred function tag for modeling small clause. So NP, AP, degP and NumP has -pred suffix e.g. NP-pred. Similarly, Urdu.Kon.TB uses function tags to encode case information of the phrase head.

Our treebank used the concept of function tag in a systematic way (as depicted in example 2 in section 4) to represent different layers of syntactic and semantic information.

4. Urdu Treebank Design

The basic design principles of Urdu Treebank were:

(a) a phrase structure bank, as it helps in syntax-intonation interface. However, a phrase to dependency convertor is part of the future work.
(b) smaller tagsets, if possible, to help annotators. The idea of smaller tagset is in line with the universal phrase labels [4] and propbank [13]
(c) a modular design, so different applications may retrieve the required annotation information from the treebank. The encoded semantic roles are not for immediate use. The parser will ignore this layer, however they can be used in the semantic parser in future.

The Urdu Treebank consists of three layers: phrase labels, grammatical function and semantic role. The text is annotated in the form of XML representation. In this paper, we show the equivalent bracketed notation that is widely comprehensible. The labels of each bracketed phrase encode all the three layers of the representation. The labels of each layer are separated by a hyphen. Following is the template of annotation scheme.

```
2. (PhraseLabel-GrammaticalFunction-
   Semanti cRole-ChunkId
        word1/pos1   word2/pos2   ….
        wordn/posn)
```

The chunkId part is explained in 4.2.9. An example from English using our representation scheme is following

```
3. (S (NP-SUBJ-Agent Casey)
      (VP will
      (VP throw
      (NP-OBJ-Theme the ball)     )))
```

Following section discusses the details of the corpus and the layers of annotation.

4.1 POS Tagged Corpus

We used CLE POS tagged Urdu Digest Corpus [14] for syntactic annotation. The corpus consists of sentences having unique ids. The corpus was manually edited to deal the common segmentation problems of Urdu text The token are separated by space and multiwords have Zero Width Non Joiner (ZWNJ) character between its components. The corpus was tagged by using CLE POS tagset [15].

The tasks of annotation was divided in three steps. The first step is of pilot annotation for testing and revising the annotation guidelines. In this step, 200 sentences were annotated. Annotation scheme and guidelines are revised according to the feedback of the annotators. In second step 1100 more sentences were annotated. In the third step, the whole of the remaining corpus (around 6,000 sentences) will be annotated[2].

4.2 Phrase Labels

The first layer of treebank consists of phrase labels. We are inspired by the small tagset introduced by Han et. Al [4]. At the design phase, a list of 10 phrase labels are identified. During the pilot annotation phase two

[2] This paper presents the design of the treebank and pilot annotation of 1300 sentences. The further work is mentioned, but that is not in the scope or not a contribution of this paper.

more phrase labels are added to the list. The description of phrase labels are given below.

4.2.1 S and SBAR. The phrase label S is used for main/matrix/independent sentences and clauses. SBAR is used for subordinate clause/sentence. Penn Treebank has SBAR, SINV and SQ for different types (and word order) of clauses, however we do not use these labels that are designed for English syntax. The main reason for SBAR is that the POS tagset differentiate between coordinating and subordinating conjunctions. So we want to keep this distinction in all the layers (if possible). Some examples of S and SBAR are following.

```
4. (S vuh[he] chAhtA[want] he[is]
      (SBAR kah[that]
      (S sEb[apple] kHAyE[eat]))) )
      'He wants to eat apple.'
5. (NP laRkA[boy] (SBAR jo[who]
      (S sEb[apple] kHA[eat]
      rahA[progressive] he[is]) ) )
      'the boy who wants to eat apple'
```

4.2.2 VC (Verb Complex). The phrase label VC is used for verbs, auxiliaries, light verbs and particles/adverbs of the verbs. The object is not part of verbal complex as we followed the analysis used in Urdu Pargram [7].

```
6. (S (NP vuh[he]) (NP kitAb[book])
      (VC parH[read] hi[intensifier]
      nahIN[not]   rahI[progressive]
      hE[is]) )
      'She is not reading the book.'
```

Urdu has Noun+LightVerb and Adjective+LightVerb complex predicates [16] e.g. *Yad* 'memory.noun' *kar* 'do.verb' for 'memorize' and *sAf* 'clean.adj' *kar* 'do.verb' for *clean*. In our annotation scheme, the noun or adjective is not the part of VC as these act syntactically as noun or adjective phrases.

```
7. (S (NP vuh[he]) (NP sabaq[noun]) (NP
   yAd[memory)          (VC      kar[do]
   rahA[progressive] tHA[was]) )
      'He was memorizing the lesson.'
```

4.2.3 Noun Phrase (NP). Noun Phrase has noun and its modifiers, specifiers and intensifiers. The CLE POS tagset considers the adverbials like *andar* 'inside' and *Aj* 'today' as a noun because these are syntactically similar to nouns. We use the same argument to label the following as a noun phrase. Following are some examples of NP.

```
8. (S (NP vuh[he) bHI[too] )
(NP ye[this] acHcHI[good]kitAb[book])
(VC parHtA[read] hE[is] ) )
      'He also reads this good book.'
9. (S (NP tum[you] (NP
kal[yesterday]) (NP andar[inside])
(VC AyE[come]  tHE[was]) )
      'You came inside yesterday.'
```

4.2.4 AdjP, QP, DMP and ValaP. Adjective Phrase (AdjP), Quantifier Phrase (QP), Demonstrative Phrase (DemP) and Vala Phrase (ValaP) are usually (not always) embedded inside the noun phrase (NP).

One of the goals of annotation guideline is to make speed of annotation faster, if possible, without compromising on the quality of representation/modeling. Hence, it is decided that if the phrase consists of a single word (e.g. an adjective only) inside the noun phrase then the annotator will not enclose the word with the phrase brackets and phrase label. In example 8, the adjective acHcHI is not enclosed by AdjP. However, if the adjective has modifier or intensifier then AdjP will be created. For example:

```
10. (NP
      (AdjP buhat [very] acHCHI[good]
      sI[particle]) kitAb[book] )
        'very good book'
```

The similar guideline applies for QP, DemP and ValaP used inside the NP. If any of these phrases appear at clause level i.e. directly inside S (or SBAR) then we always put the bracket even around the single word. See the following example.

```
11. (S (NP kitAb[book])
        (ADJP acHcHI[good]) hE[is])
         'Book is good.'
```

The labels DemP and ValaP were not part of the set of phrase labels listed in the design phase. However, the pilot annotation provides the cases for which these labels are required. Like other pos categories, demonstrative can also have particles like intensifiers and focus particles. Hence we use the general rule that if the category word has some other word attached to it as a modifier or particle then the whole sequence is enclosed in the phrase label. See the following example:

```
12. (NP
     (DMP   kOI[any]   bHI[intensifier])
            bAt[matter.noun] )
          'any matter'
```

The ValaP phrase is used in the constructions having the pos vAlA (roughly translated as 'one'). See the examples:

13. (**NP** (**ValaP** (**NP** tasvIr vAlI) kitAb[book]))
'books with/having pictures'

It must be noted that we introduced ValaP instead of VP, DMP instead of DP and VC (verbal complex) instead of VP as the later ones have their formal definition and usage in different syntactic theories and nomenclatures. Hence, we used longer or different names for the new labels introduced in our design.

4.2.5 Pre-and-Postpositional Phrases. Urdu has postpositions (that follow noun phrase). There are some borrowed positions from Arabic and Persian [17] that are rarely used in Urdu. Hence, the phrase labels PP (postpositional phrase) and PrP (prepositional phrase) are used in the treebank guideline. The examples are:

14. (**PP** tum[you] nE[*ergative*])
'You'

15. (**PP** gHar[home] tak[till])
'till home'

16. (**PrP** sivAE (**NP** mErE))
'except me'

It must be noted that neither the pos tagset nor the phrase labels distinguish between case markers and postpositions as distinguished in Urdu Pargram. This is done for the sake of simplicity (at phrase layer) and similar syntax. The functional difference between *nE* and *tak* is modelled through the grammatical function layer.

As described earlier, the adverbial nouns like *andar* 'inside' and *Upar* 'above' etc. are the head of the noun phrase as in the following example:

17. (**NP**(**PP** gHar[house] kE[of]) andar[inside]))
'inside the house'

4.2.6 Adverbial Phrase. The adverbial phrase has adverbs as the head word. For example:

18. (**S** vuh[he] (**ADVP** bA_AsAnI[easy]) (**VC** AyA[came]))
'He came easily.'

In Urdu, adverbial function is usually expressed by a prepositional phrase or noun phrase. For example, the following sentence has a PP. However, both (18) and (19) will the same grammatical function in the second layer of annotation.

19. (**S** vuh[he] (**PP** (**NP** (AsAnI[easy] sE[with])) (**VC** AyA[came]))
'He came easily.'

4.2.7 X. The phrase label X is used for fragments that cannot have a phrase label from the above list.

4.2.8 Conjunction. The conjunction is modelled by enclosing the components into a parent phrase label. For example,

20. (**NP** (**NP** sEb[apple]) yA (**NP** Am[mango]))
'apple and mango'

We do not introduce any phrase label e.g. conjunction phrase for enclosing the conjuncted components.

4.2.9 Discontinuous Phrases. We find examples of discontinuous phrases during the pilot annotation phase. The discontinuous NP in Urdu was earlier discussed in [17]. Consider the following example.

21. (**S** (**NP** vuh[he]) (**VC#1** rO[cry] (ADVP kiyoN[why]) (**VC#1** rahA hay))
'Why is he crying ?'

In this example, the VC is not contiguous. We assign the same chunk id to all the components of discontinuous phrases.

4.3 Grammatical Function

The second layer of treebank is of grammatical function. As depicted in (2), the grammatical function follows the phrase label separated by a hyphen. The set of grammatical functions is inspired primarily by lexical functional grammar. Following is a brief introduction of grammatical functions.

4.3.1 Subject and Object. The syntactic subject and object have the corresponding grammatical functions. See the following example.

22. (**S** (**NP-SUBJ** laRkI[girl]) (**NP-OBJ** kitAb[book[) (**VC** paRHtI[read] hE[is]))
'The girl reads book.'

Universal Dependencies have three different labels for subject. nsubj (nominal subject), csubj (clausal subject) and npaassubj (nominal subject of passive construction). However, we do not follow this scheme because the information about nominal (noun phrase) vs clause is already represented through phrase label.

4.3.2 Oblique (OBL). The oblique grammatical function (OBL) is used with those compulsory arguments that are not the syntactic subject or object e.g. the source/goal of the motion verbs, non-canonical second argument [18] and genitive marked argument in N+V complex predicate.

```
23. (S (NP-SUBJ vuh[she])
   (NP-OBL  gHar[home])
   (VC ponhcHI[reached]) )
       'She came home.'
```

```
24. (S (NP-SUBJ vuh[she])
       (PP-OBL      (NP   sANp[snake])
       sE[from])
       (VC dartI[fear] hE[is]) )
       'She fears snake.'
```

4.3.3 Adjunct (ADJ). The non-mandatory arguments are marked as ADJ (adjunct). Any adverbial function, whether syntactically realized as NP, PP or ADVP are marked as having ADJ grammatical function. For example, both ADVP and PP in examples (18) and (19) in 4.2.6 (Adverbial Phrase) are marked as having ADJ.

4.3.4 COMP. The dependent clauses have COMP grammatical function. We do not differentiate between COMP and XCOMP for the sake of simplicity. For example:

```
25. (S (NP-SUBJ vuh[he])
   (VC chAhtA[want] he[is])
   (SBAR kah[that] (S-COMP (NP-SUBJ
   sEb[apple]) (VC kHAyE[eat]))) )
       'He wants that he eats apple.'
```

4.3.5 Predicate Link (PDL). The grammatical function PDL (Predicate Link) is used in the copular constructions. For example:

```
26. (S (NP-SUBJ laRkI)girl] )
       (ADJP-PDL aqalmand[wise])
       (VC hE[is]))
       'The girl is wise.'
```

```
27. (S (NP-SUBJ vuh[he])
       (NP-PDL sadar[president])
       (vC banA[made])
       'He became president.'
```

4.3.6 INTJ. This grammatical function was introduced as the result of pilot annotation. It occurs with NPs having addressees. For example:

```
28. (S (NP-INJ bETI[daughter])
       (NP-SUBJ    tum[you])    (NP-ADJ
       kab[when])
       (VC AI[come]) )
       'Daughter, when did you came?'
```

4.3.7 POF (Part of Function). Part of function marks the noun or adjective part of the complex predicate. In 4.2.2, we mentioned that these noun/adjective are not phrasal part of the VC (Verb Complex). However these are functional related with the verb, hence we introduced a functionaltag to encode this relation. The example (7), described in 4.2.2, with the grammatical function layer becomes:

```
29. (S (NP-SUBJ  vuh[he])
(NP-OBJ sabaq[noun])
(NP-POF yAd[memory) (VC kar[do]
rahA[progressive] tHA[was]))
      'He was memorizing the lesson.'
```

4.3.8 Other grammatical functions. For the annotation guideline, we introduced only the sentence/clause level grammatical functions. The other types of grammatical function (e.g. modifiers/specifiers of the noun) are not part of the scheme. Our assumption is that there is one to one correspondence between such phrase labels and grammatical functions i.e. the grammatical function ADJ should follow the phrase label ADJP used inside NP and the grammatical function SPEC (as used in LFG framework) should attach with DMP etc.

4.4 Semantic Role

Semantic Role is the third layer of treebank. We used the Propbank roles, as these are (a) specially designed to have a small set of roles and (b) an Urdu corpus has already been tagged using these roles [19] and Urdu specific roles e.g. for dative subjects, causer and intermediate agent were already introduced. For example:

```
30. (S (NP-SUBJ-ARG0_GOAL Ali ko[dtv])
    (NP-OBJ-ARG1 THanD[cold])
    (VC lagi[hit])
         'Ali felt cold.'
31. (S (NP-SUBJ-ARGA Ali nE[ergative])
    (NP-OBL-ARG0_MNS Ahmed sE[from] )
    (NP-OBJ-ARG1 sEb[apple] )
    (VC katvayA[cut.caus])
        'Ali   caused   Ahmed   to   cut
      apple.'
```

5. Conclusion and Future Work

In this paper, we describe the design of a multilayer annotation scheme of Urdu corpus and then annotation of 1,300 sentence using this annotation scheme. The immediate purpose of this treebank is to create parse trees for the Text to Speech System.

We used small sets of tags to annotate the phrase, grammatical functions and semantic roles. Most importantly, we introduced demonstrative phrase, Interjection grammatical function and modeling of discontinuous phrases.

As further work, more sentences are annotated and probabilistic parser is created. However, the creation of the parser is not in the scope or contribution of this paper.

9. References

[1] J. E. Grimes and B. F. Grimes (eds.), *Ethnologue. Volume 1: Languages of the World; Volume 2: Maps and Indexes*. 14th edition, SIL International, Dallas, 2000.

[2] N. Chomsky, *Syntactic Structures*, Mouton, The Hague, 1957.

[3] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: the Penn Treebank", *Computational Linguistics, 19(2)*, 1993.

[4] A. Han, et al., "A Universal Phrase Tagset for Multilingual Treebanks", *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data,* Springer International Publishing, 2014.

[5] Marie-Catherine De Marneffe, et al, **"**Universal Stanford Dependencies: A cross-linguistic typology*"* in *Proceedings of LREC 2014*, 2014.

[6] J. Nivre, et al., "Enhancing Universal Dependency Treebanks: A Case Study", In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, 2018.

[7] M. Butt, et. al., "The Parallel Grammar project", In *Proceedings of COLING 2002. Workshop on Grammar Engineering and Evaluation*, 2002.

[8] R. Bhatt, et al., "A multi-representational and multi-layered treebank for Hindi/Urdu", In *Proceedings of the Third Linguistic Annotation Worksho*p, 2009.

[9] A. Hautli, et. al. "Adding an Annotation Layer to the Hindi/Urdu Treebank", *Linguistic Issues in Language Technology, 7(3),* Stanford: CSLI Publications, 2012.

[10] Q. Abbas, *Building Computational Resources : The URDU.KON-TB Treebank and the Urdu Parser*, KOPS, Konstanz, 2014.

[11] M. Steedman, "Information Structure and the Syntax–Phonology Interface", *Linguistic Inquiry, 31*, 2000.

[12] D. Büring, "Syntax, information structure, and prosody", *The Cambridge Handbook of Generative Syntax,* Cambridge University Press, 2013.

[13] P. Kingsbury and M. Palmer, "From TreeBank to PropBank", In *Proceedings of LREC 2002*, 2002.

[14] S. Urooj, et al. "CLE Urdu Digest Corpus". In *Proceedings of Conference on Language and Technology 2012 (CLT12)*, 2012.

[15] T. Ahmed, et al., The CLE Urdu POS Tagset, In *Proceedings of LREC 2014*, 2014.

[16] M. Butt, et. al., "Complex predicates via restriction", In *Proceedings of the LFG03 Conference*, 2003.

[17] G. Raza, *Subcategorization Acquisition and Classes of Predication in Urdu*, KOPS, Konstanz, 2011.

[18] T. Ahmed, *Spatial Expression and Case in South Asian Languages*, KOPS, Konstanz, 2009.

[19] M. Anwar, et. Al., A Proposition Bank of Urdu, In *Proceedings of LREC 2016*, 2016.

A Sentiment Lexicon for Urdu

Sahar Rauf, Kinza Rahim, Maryam Khalid, Ehsan ulHaq, Kashif Javed
Center for Language Engineering, Al-Khwarizmi Institute of Computer Sciences University of Engineering and Technology Lahore, Pakistan
firstname.lastname@kics.edu.pk

Abstract

Sentiment analysis is a data mining technique, which measures the inclination of people's opinions. Recent studies have shown that the sentiment lexicon can be developed using automatic and manual tagging techniques. The seminal works on Urdu lexicon done so far do not actually denote a broad Lickert scale for data tagging and also do not cover all the open word classes. The current study aims to develop a sentiment lexicon and test its validity using manual and automatic methods. The dictionary-based method is used to design this lexicon using three authentic Urdu dictionaries. The data was tagged on a five point lickert scale i.e. -2 to +2 using the formulated guidelines. The lexicon is composed of four-word classes namely nouns, verbs, adjectives and adverbs. Once the lexicon was developed using manual tagging techniques it was tested both manually and automatically. The manual testing yielded an inter annotator agreement of 75% while the automatic testing included the comparing of the sentiments of the developed lexicon with the UCI Corpus. The result yielded a percentage accuracy of 84%. The lexicon was validated for its accuracy by both the results.

1. Introduction

Sentiment analysis (SA) is one of the fastest growing fields of Natural Language Processing (NLP) and text mining under the umbrella of Artificial Intelligence (AI) [1]. SA measures the inclination of people's opinions through NLP, computational linguistics and text analysis. Sentiment analysis has emerged from human behavior of decision making by consulting friends or family about their opinions in daily life.

SA can also be used to extract and analyze subjective information from Web that includes social media comments, online reviews and other similar text sources. The analyzed data helps in calculating the public's sentiments or opinions toward certain product, people or ideas and reveal the contextual polarity of the information [2].

Two main approaches can be used for the sentiment analysis i.e. machine learning approach and lexicon-based approach. In machine learning approach, the data is classified by training a classifier on the labeled data [3]. In the lexicon-based approach, lexical items from dictionary are assigned positive and negative polarities. The lexicon is composed of a defined list of sentiment words along with their intensities and polarities [4]. The lexicon-based approach involves the calculation of sentiment from the semantic orientation of phrases and words that occur in a sentence [5]

Lexicon based approach is considered as a simple and reliable as compared to machine learning approach since it avoids the need to develop a labeled training set. Moreover, it is also difficult to ensure the correctness of labeled data in machine learning approach. To develop a sentiment lexicon, some researchers use dictionary or corpus-based approaches. Corpus based approaches involve the determining of the patterns of co-occurring of words to determine the sentiments of the words and phrases. The dictionary-based approach helps to compile the sentiment words and use the antonyms and synonyms in WordNet to determine the sentiments of the lexical items [6].

Depending on the nature of the data and choice of the users, the process of sentiment analysis can be performed on three different levels. These three levels of analyses are; 1) document level sentiment analysis, 2) aspect level sentiment analysis and 3) sentence level sentiment analysis [3]. In Sentence level sentiment analysis, each sentence is classified as positive, negative or neutral. Here the sentence is considered as a separate unit expressing a single opinion.

The aim of our research is to develop an Urdu sentiment lexicon marked on a five-component Likert scale ranging from -2 to +2. The developed lexicon comprehensively covers the four major word classes i.e. nouns, verbs, adverbs and adjectives. Conclusive guidelines are developed to annotate the data with different polarities. The study also aims to test the accuracy of the Urdu sentiment lexicon by using manual and algorithm-based approaches.

The current study is organized as follows: Section 2 labeled as literature review highlights some seminal research works related to the topic, Section 3 highlights the methodology undertaken for the research. Section 4 covers the results of data analysis, Section 5 provides the discussion of the results, and Section 6 concludes the research and discusses the future dimensions.

2. Literature Survey

Due to unavailability of resources in other languages, sentiment analysis in multiple languages often involves transferring knowledge from one resource-rich language to other resource-poor languages [7]. Majority of multilingual sentiment analysis systems employ English lexical resources such as SentiWordNet. A popular approach towards SA is to use a machine translation system to translate texts from languages into English. The original text is translated into English, and then English SA resources such as SentiWordNet are employed [7]. However, translation systems pose various problems such as; sparseness and noise in the data [8]. Sometimes, translation system fails to translate essential parts of the original text which can possibly reduce the text's original sentiments [9].

SentiWordNet assigns WordNet synsets to three categories: positive, negative, and neutral by using numerical scores ranging from 0.0 to 1.0 to indicate the degree to which the terms included in the synset belong to the corresponding category. SentiWordNet is built by performing quantitative analysis of glosses for synsets [10]. One drawback of SentiWordNet is that it assigns polarity at the syntactic level but fails to assign polarities to phrases such as "getting angry" or "celebrate a party" which correspond to concepts found in the text to express positive or negative opinions [11].

Moreover, multilingual lexical resources specific to sentiment analysis are also developed. The NTCIR corpus of news articles in English, Chinese, and Japanese is made up of information on sentiment polarity and opinions for sports and political news data [12].

Different techniques can be used in the extraction and tagging of lexicons. The extraction of SentiUnits using shallow parsing techniques in order to create a sentiment lexicon can be considered as one of the most authentic method [13]. SentiUnits are expressions which carry sentiment information in the sentence.

Cambria et al. [11] proposed a SenticNet, lexical resource based on a multi-disciplinary approach to identify, interpret, and process sentiment in the Internet. SenticNet is more suited for a concept-level sentiment analysis and can also be utilized to evaluate texts based on common-sense reasoning tools that require large input. It employs a Sentic computing methodology, in particular, to evaluate texts at document or sentence level. It performs the task of building a collection of concepts, including common-sense concepts, supplied with positive or negative polarity labels. Unlike SentiWordNet, SenticNet does not assume a neutral polarity. It guarantees high accuracy in polarity detection with the availability of multilingual tools as well.

Many researchers use Semantics in creation of lexicon for performing SA. For the SA of twitter, a lexicon-based approach is presented called SentiCircles [14]. This approach considers the patterns of words that occur mutually according to different contexts, get their semantics and then update the sentiment lexicon accordingly by updating the pre-assigned polarity and strength of these patterns. Sentiment Knowledge is encoded into pre-trained word vectors for improving the performance of SA, where the proposed method is based on external sentiment lexicon and a convolution neural network.

Remus et al [15] worked with German inquirer, which is a German sentiment lexicon supplied with positive and negative labels. It was constructed using Google translate by translating words and terms into the German language. The words without any sentiment were removed from the German Inquirer. SEL is a Spanish emotion lexicon that contains 2036 words marked with the Probability Factor of Affective use (PFA) as the measure of their expression of basic emotions: joy, anger, fear, sadness, surprise, and disgust, on the scale of null, low, medium, or high. This lexicon was marked manually by 19 annotators who had to agree on a certain threshold for a label on the word to be included in the lexicon. Probability Factor of Affective use was developed by the authors of SEL to incorporate agreement between annotators in the decision-making process of labeling the sentiment on a word.

Mobarz et al. [16] created a sentiment Arabic lexical Semantic Database (SentiRDI) by using a dictionary-based approach. The database has many inflected forms, i.e., it is not lemma-based. Moreover, the authors reported insufficient quality and plan to try other alternatives.

Different researchers have also developed Urdu sentiment lexicons. An Urdu corpus, labeled with semantic role by using cross lingual projection, is developed [17]. Syed [18] proposed an innovative sentiment annotated lexicon for Urdu based on SentiUnits. Syed started by extracting SentiUnits i.e. positive and negative expressions, from a given Urdu text, using shallow parsing technique. Hashim and Khan [1] developed a sentiment analyzer based on Urdu Nouns and Adjectives for sentence level sentiment analysis. Hashim used Urdu news data from headlines by using a lexicon based on nouns and adjectives. Mukhtar and Khan [19] used a lexicon-based approach for sentiment analysis of Urdu blogs, using a publicly available Urdu Sentiment Lexicon [20]. They included adjectives, nouns and negations; as well as verbs, intensifiers and context-dependent words. The developed Urdu sentiment analyzer applies rules, use lexicon and perform Urdu sentiment analysis by classifying sentences as positive, negative or neutral.

A lot of work has been done previously on the difficulties, strategies and the utilization of sentiment analysis of English language. The sentiment analysis techniques designed for English language cannot be utilized for Urdu language due to its different script, morphological and syntactic patterns. Because of the different structure of Urdu language, other languages SA cannot be employed for resolving the issues of Urdu language. Moreover, Urdu Sentiment Lexicons developed so far have covered the categories of nouns, verbs and adjectives while little or no attention has been paid on adverbs. However, this study stands out as it proposes comprehensive guidelines for developing an Urdu sentiment lexicon. This research also proposes a five component Likert scale ranging from -2 to +2 whereas the sentiment lexicons for Urdu language developed so far do not actually put forward such a comprehensive five level Likert scale. A maximum of three scale i.e. -1 to +1 Likert scale for Urdu sentiment Lexicon has been anticipated so far. Moreover, this developed lexicon comprehensively covers the four major word classes i.e. nouns, verbs, adverbs and adjectives.

3. Methodology

This section elucidates the research procedure, sampling technique and the aspects of data analysis tool and procedure.

The research is mixed method in nature and the research design is cross sectional. The methodological procedure is divided into two phases:

i. Development of Urdu Sentiment Lexicon
ii. Testing of Urdu Sentiment Lexicon

3.1. Development of Urdu Sentiment Lexicon

Manual tagging techniques such as dictionary-based methods were used to develop Urdu Sentiment Lexicon. For the manual tagging of the data, a comprehensive set of guidelines were also developed. The established guidelines provide an elaborative framework of tagging various word classes namely: nouns, verbs, adjectives and adverbs. Five point Likert scale comprising of -2, -1, 0, 1, 2 values was used to assign polarities to lexical items. The development of sentiment lexicon was further divided into two stages:

3.1.1. Data Extraction. At the first stage the lexical items to be tagged are extracted from a selected corpus [21] and around 35 M words were extracted. The sample undertaken for the development of Sentiment Lexicon included around 21556 words which mark the most frequently occurring words from the selected corpus which were already assigned with POS tags [21]. The extracted POS categories with their numbers are presented in Table 1 :

Table 1 POS categories and number of words in each category

POS Categories	**Number of words**
Nouns	15826
Verbs	3097
Adjectives	1986
Adverbs	646

3.1.2. Data Tagging. At the second stage, the extracted lexical items were tagged according to the developed guidelines by a team of linguists. Three authentic Urdu dictionaries [22] [23] [24] were also used to analyze the lexical items. The postulates of the formulated guidelines are as follows:

1. Assign higher polarities to the words that give clear sense of positivity or negativity while assign lower polarities to words that show rather vague sense. Neutral should be only those words that do not show any orientation.

2. Words that show some positive or negative sense without any context should be tagged according to their prior polarities. Moreover, words that depict strong positivity or negativity are assigned stronger polarities i.e. +2 or -2. For instance, the polarity of /دوست/ /friend/ /d̪o:st̪/ is +2 and the polarity of /دشمن/ /enemy/ /d̪uʃmən/ is -2.

3. Words having multiple parts of speech tags whose polarities change according to the POS category are assigned respective polarity for each tag. For instance; the polarity of the word / محسن / /mohsın/ (Noun) name will be assigned a 0 polarity, while /محسن/ /friend/ /mohsın/ (Adjective) will be marked as +2.

4. The polarity of the words increase with the increase in the degrees of adjectives. For instance, the words /بد/ /bad/ /bəd̪/ and / بدترین/ /pathetic/ /bəd̪t̪əri:n/ will be assigned -1 and -2 polarities respectively. For adjectives ending at 'ی', orientation of the root word should be checked and assign the polarity accordingly. For instance the adjectives like; /قومی/

/national/ /kɔ:mi:/ and / /اندھیری/ /dark/ /ənd̪ʰe:ri:/ will be assigned 1 and -1 polarities respectively.

5. **5.** The singular words will have low polarity strength whereas; their plurals will have higher strength of polarity. Polarity increases with the increase in numbers. For instance, the singular noun /مشکل/ /difficulty/ /muʃkɪl/ and its corresponding plural form /مشکلات/ /difficulties/ /muʃkɪla:t̪/ /will be assigned -1 and -2 polarities respectively.
6. **6.** Nouns that represent ranks, titles or respect will be tagged with higher polarities. For example, /صاحب/ /Sir/ /sa:hɪb/ and / حضرت /hazrat/ /həzrət̪/ will be assigned +2 polarity.
7. **7.** Words showing sense of certainty either positive or negative will be given higher polarities and those showing uncertainty will be assigned lower polarities. For instance, the adverbs /اچانک/ /suddenly/ /ətʃa:nək/ and /روزانہ/ /everyday /ro:za:na:/ will be assigned a polarity of -1 and 1 respectively since /اچانک/ represents a sense of uncertainty and /روزانہ/ represents a sense of certainty.
8. **8.** Verbs that convey some proper meaning as a word will be given higher polarities as compared to the words that themselves do not give proper meaning and need associating words with them. For instance /پڑھنا/ /study/ /pəɽhna/ and /رہتی/ /stays/ /ræht̪:/ will be assigned +2 and +1 polarities respectively since /پڑھنا/ has proper meaning attached to it whereas /رہتی/ needs associating words to deliver its complete meaning.

3.2 Testing of Urdu Sentiment Lexicon

The developed Urdu Sentiment Lexicon was tested both manually and automatically:

3.2.1. Manual testing process. Manual testing process comprised of two steps:

1) 10% reference of the untagged original data was sampled automatically and marked manually by an expert linguist according to the guidelines
2) Another linguist assigned polarities to the original data set. Then the results of 10% tagged reference were compared with the polarity assigned original data to depict an inter annotator agreement between the two data sets. A threshold of 75% accuracy was established to check the quality of the data. The tagged source data found below the above-mentioned accuracy was sent back to the linguist for the further review.

3.2.2 Automatic testing process. For the purpose of analyzing the reliability of our lexicon, a baseline system was also developed and the accuracies were computed. The lexicon was tested using a subset of 500 sentences extracted from Roman Urdu sentiment dataset UCI machine learning repository [25] .These 500 roman Urdu sentences were then transliterated into Urdu before being processed. Following algorithm was used for automatically tagging the sentiment of a given input sentence S :

a) Remove special symbols and punctuations from the sentence S
b) Compute total positive words (tpw) in a sentence S
c) Compute total negative words (tnw) in a sentence S
d) Now assign sentiment to S using the following rules
    i. If tpw>tnw, assign positive sentiment to S
    ii. If tnw>tpw, assign negative sentiment to S
    iii. if tpw=tpw, assign neutral sentiment to S

After the application of the above algorithm, the accuracy of the lexicon was determined.

4. Results

This section elaborates the results obtained after the lexicon is manually and automatically tested.

A lexicon of around 21556 words was developed. The developed lexicon was manually tested and a 75% inter-annotator agreement was achieved. Around 25% data showed a mismatch of polarities due to the subjectivity of the annotator and the influence of the pragmatic stance during the data tagging process.

For the further validation of the lexicon, an automatic process was also used. A sample of 500 sentences (already assigned with a positive, negative or a neutral sentiment) from UCI corpus was taken. Then, the sentences were automatically tagged using our lexicon and the marking of lexicon and UCI corpus was compared. Through this process, an overall accuracy of 84.0% was achieved. Both manual and automatic testing results validated the lexicon for its validity and accurate nature.

5. Discussion

Since the lexicon was tagged on a five-point Likert scale which implies that word can be assigned a sentiment of being positive, very positive, neutral, negative or very negative. The polarities of the lexical items can greatly be influenced by the annotators' subjectivity, opinion, pragmatics and contextual domain in which the word is occurring. For instance, words like; /حکومت/ /government/ /hʊku:mət̪/ can have different orienta tion. If the annotator is a supporter of the current government then he would mark it as +1 but if he is not a supporter so he will mark it as -1.

Also, if the adjective /تیز / /fast/ /t̪e:z/ is considered then the word would carry a positive sentiment but, in the sentence,

/وہ بہت تیز ہے/

vo:	bo:hət̪	t̪e:z	hæ:
she	very	cunning	is

She is very cunning,

The same word would carry a -2 sentiment due to the negative sense enacted by the word. Due to all these subjectivity problems, it was instructed to the linguists to not to think the contexts of the words rather mark them in their dictionary meaning that's why three online dictionaries were used for the analysis.

In addition to this, there were certain words whom meaning varied by the addition of diacritics. For instance the word سونا can either be سُونا/su:nɑ:/ desolate/ or سونا /sleep/ /so:nɑ:/. Also, the word classes vary due to the presence of diacritics where سُونا is an adjective and سونا is a verb. That is why, we used POS tagging and separated the words into their different classes to mark them easily and get the inter- annotator accuracy.

The data was also validated using a corpus-based approach and the polarity of the whole sentence was determined by the sum of the polarities of individual lexical items in the sentence. The testing also posed the good accuracy of 84%. However, there were certain sentences where the polarities assigned by the algorithm of the developed Urdu Lexicon and already assigned polarities of the UCI corpus showed a discrepancy. For instance,

/اللہ اللہ کر کے آئے/

/Thank God they came/

/əlla:h əlla:h kərke: a:e:/

The above sentence had a negative sentiment attached in the UCI corpus while the sentiment assigned by the developed lexicon using automatic means was positive. Since, we being the Muslims attach positive sentiment to اللہ i.e +2 and same was assigned in the Sentiment lexicon. So, a mismatch was found. The reason might be that they marked the sentence in the contextual sense but we did not consider the context while marking the lexical items.

6. Conclusion and Future Directions

The sentiment lexicons hold a great importance in defining the semantics of particular lexical items. The current study defines the development of an Urdu sentiment lexicon. The current research can be regarded as authentic since it consists of a wide repository of lexical items (nouns, adjectives, verbs and adverbs) marked on -2 to +2 Likert scale. Also, the manual testing results yielded an accuracy of 75% and the accuracy was further validated by the automatic testing method where the percentage accuracy attained was 84%. The findings of this lexicon can further be applied on bigrams including collocations and phrasal verbs and the repository of the words can be enhanced. However, this data can also be further validated using machine learning techniques in future.

The developed lexicon can also be used to create a business intelligence system and can provide aid in the sentiment analysis of a particular corpus, which can include online reviews, feedbacks and twitter comments. The opinion mining of news data can also be executed through it. Moreover, this lexicon can be integrated in any programming language to develop an automated sentiment analysis system.

Since the lexicon was automatically tested and an overall accuracy of 84% was achieved, this number could be further improved by modification of the algorithm and the percentage accuracy of the lexicon can be taken to 95% and above.

Different dimensions can further be explored e.g, polarity of modifiers, negations, pronouns and lexically borrowed words from English can be studied. The population of adjectives in the lexicon can be increased to further improve the lexicon since the adjectives hold a great stance in defining a sentiment.

7. References

[1] F. Hashim and M. A. Khan, "Sentence Level Sentiment Analysis using Urdu Nouns," in Conference on Languauge and Technology, Lahore, Pakistan, 2016.

[2] R. Eskander and O. Rambow, "SLSA : A Sentiment Lexicon for Standard Arabic," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015.

[3] B. Liu and L. Zhang, "A Survey of Opinion Mining and Sentiment Analysis," in Mining Text Data, Boston, Springer, 2012.

[4] P. Gonçalves, M. Araújo, F. Benevenuto and M. Cha, "Comparing and combining sentiment analysis methods," in Proceedings of the first ACM conference on

Online social networks, Boston, Massachusetts, USA, 2013.

[5] M. Taboada, J. Brooke, M. Tofiloski and K. Voll, "Lexicon-Based Methods for Sentiment Analysis," Computational Linguistics, pp. 267-307, 2011.

[6] X. Ding, B. Liu and P. Yu, "A Holistic Lexicon-Based Approach to Opinion Mining," in 15th ACM SIGKDD international conference on knowledge discovery and data mining, 2008.

[7] K. Denecke, "Using SentiWordNet for multilingual sentiment analysis," in IEEE 24th international data engineering workshop, 2008, 008.

[8] Balahur and M. Turchi, "Multilingual Sentiment Analysis using Machine Translation?," in Proceedings of the 3rd workshop in computational approaches to subjectivity and sentiment analysis, 2012.

[9] M. Bautin, L. Vijayarenu and S. Skiena, "International sentiment analysis for news and blogs," in ICWSM, 2008.

[10] . V. Singh, R. Piryani, A. Uddin and P. Waila, "Sentiment analysis of textual reviews; Evaluating machine learning, unsupervised and SentiWordNet approaches," in 5th international conference on knowledge and smart technology (KST). IEEE, 2013.

[11] E. Cambria, R. Speer, C. Havasi and A. Hussain, "SenticNet: A Publicly Available Semantic Resource for Opinion Mining," AAAI fall symposium: commonsense knowledge, p. 02, 2010.

[12] Y. Seki, D. K. Evans and L.-W. Ku, "Overview of Multilingual Opinion Analysis Task at NTCIR-7," in Proceedings of the 7th NTCIR workshop meeting on evaluation of information access technologies: information retrieval, question answering, and cross-lingual information access, 2008.

[13] D. Jayraj and S. Andhariya, "Sentiment analysis approach to adapt a shallow parsing based sentiment lexicon," in International Conference on Innovations in Information, Embedded and Communication Systems, 2015.

[14] H. Saif, Y. He, M. Fernandez and H. Alani, "Contextual Semantics for Sentiment Analysis of Twitter," Inf Process Manage , pp. 5-19, 2016.

[15] R. Remus, U. Quasthoff and G. Heyer, "SentiWS-a publicly available German-language resource for sentiment Analysis," in LREC, 2010.

[16] H. Mobarz, M. 7Rashwan and I. AbdelRahman, "Generating lexical Resources for Opinion Mining in Arabic Language Automatically," in The Eleventh Conference on Language Engineering (ESOLE), Cairo-Egypt., 2011.

[17] S. Mukund, R. Srihari and E. Peterson, "An information–extraction system for Urdu—a resource poor language," ACM Transactions on Asian Language Information Processing, pp. 1-43, 2010.

[18] S. Afraz Z, M. Aslam and A. M. Martinez-Enriquez, "Lexicon based sentiment analysis of Urdu text using SentiUnits," in Mexican International Conference on Artificial Intelligence, Mexico, 2010.

[19] N. Mukhtar and M. A. Khan, "Effective lexicon based approach for Urdu sentiment analysis," Artificial Intellegence Review, 2019.

[20] Athar, "Chaoticity," 14 June 2012. [Online]. Available: https://chaoticity.com/urdu-sentiment-lexicon/

[21] M. Farooq and B. Mumtaz, "Urdu Phonological Rules in Connected Speech," in Conference on Language and Technology, Lahore, Pakistan, 2016.

[22] "Urdu Lughat," 2007. [Online]. Available: http://udb.gov.pk/.

[23] "Online Urdu Dictionary," 2002. [Online]. Available: http://www.cle.org.pk/oud.

[24] "Urdu Lughat," 2013. [Online]. Available: http://urdulughat.info/.

[25] Z. Sharf and S. Rehman, "UCI Machine Learning repository," IJCSNS International Journal of Computer Science and Network Security, vol. 18, no. 1, January 2018.

[26] K. Dashtipour, S. Poria, A. Hussain, E. Cambria, A. Hawalah, A. Gelbukh and Q. Zhou, "Multilingual Sentiment Analysis: State of the Art and Independent Comparison of Techniques," Cognitive Computation, pp. 757-771, 2016.

[27] M. Humayoun, H. Hammarström and A. Ranta, "Urdu morphology, orthography and lexicon extraction," in 2nd Workshop on computational approaches to Arabic script-based languages, Stanford, 2007.

[28] M. Ijaz and S. Hussain, "Corpus Based Urdu Lexicon Development," in Conference on language technology (CLT 2007), 2007.

[29] Muaz, A. Ali and S. Hussain, "Analysis and Development of Urdu POS Tagged Corpus," in 7th Workshop on Asian language resources, Suntec, Singapore , 2009.

[30] R. Tatman, "Sentiment Lexicons for 81 Languages," 2017. [Online]. Available: https://www.kaggle.com/rtatman/sentiment-lexicons-for-81-languages.

[31] U. Mirchev and M. Last, "Multi-document summarization by extended graph text representation and importance refinement," Innov Doc Summ Tech Revolut Knowl Underst Revolut Knowl Underst, 2014.

A Study of Consonant Cluster Phonotactics of English Borrowed Words in Urdu Language

Muhammad Shaban Shoukat, Muhammad Bilal, Maria Fatima Dogar
University of the Punjab
{s.shoukat7, m.bilal4767}@gmail.com, maria.ier@pu.edu.pk

Abstract

Whenever a word is borrowed it is resyllabified according to the phonology of the language it is adopted. This research paper identifies resyllabification and discusses the phonotactics of the resyllabification of the consonant clusters in the English borrowed words of Urdu language. For this purpose, words having consonant clusters were selected from Urdu dictionary. All existing CC combinations at onset and coda positions were determined and most common word for each CC cluster combination was selected. The stimulus for data collection contained the target combination word in a sentence. The data was collected from respondents with different gender, age and educational background from Lahore, Pakistan. The data was analyzed by three researchers and PRAAT was used wherever the researchers found any confusion. It is found that epenthesis occurs at onset position except when the consonants in the consonant clusters have same place of articulation and /j/ comes at post-initial position in 2-cluster combination.

1. Introduction

English has developed an important part in communication throughout the world. In Pakistan, English is learnt and spoken as a second language as communication in English is a basic need of majority of jobs in Pakistan [13]. Due to this importance of English language, Urdu language has borrowed many words from English.

Every language has its own phonological rules. English follows its own phonotactic rules and stress patterns while Urdu has its own. In accordance to rhythm, English is stress-timed language [14]. The stress pattern of Urdu and English are also different. Rehman (2015) in his book points out some interference of L1 on second language which is English in case of Pakistan. He argues that the difference can be seen at two levels. One is the segmental features and the other at non-segmental features.

In segmental features, we see the replacement of certain sounds with others like the replacement of dental fricatives /θ/ and /δ/ with /th/ and /d/ or the non aspiration of /p,t,k/. in case of non-segmental features, we see a difference of rhythm in Pakistani English as language in south Asian countries are syllable-timed while English is stress-timed language [13]. This difference leads us to the conclusion that English is different from Urdu on Phonological grounds. Thus the phonotactic rules for Urdu are also different from English. Syllable templates of Urdu as described by Nazar (2002) are different from that of English. Urdu language has only one consonant cluster at the coda position [11] while in English, consonant clusters can be seen both at onset and coda positions [14]. As the phonotactics rules of every language are different, this difference might have effect on the pronunciation of a second language by a non-native speaker. The current study was aimed at discussing the study of consonant cluster phonotactics of English borrowed words in Urdu language. The study is specific with the resyllabification of consonant clusters that occurs due to borrowing of English words in Urdu language.

2. Literature Review

Whenever cultures come in contact with each other, they affect the languages which are seen by the phenomenon of borrowing [7].

2.1. Borrowing

Borrowing is the process in which linguistic items are imported from one linguistic system to another. The history of borrowing can be traced back to the work of Haugen in 1950. Haugen has defined borrowing (as cited in Hoffer, 2002) as the "the attempted reproduction in one language of the patterns previously found in another". The types of borrowing can be explained with reference to original pattern or model. If the borrowed item is similar to the model, it is termed as import. While if it is an inadequate version of the model i.e. the original speaker would not recognize the borrowed term, it is substitution. Borrowing can be discussed further by the terms used to express borrowing. The terms used in

borrowing relate to the process rather than the result (Hoffer, 2002).
Hoffer (2005) citing Hockett (1958) defines these terms as follows:
Loanwords: In this the speaker may adopt the idea or term and the source language for each.
Loanshifts: In this the native word is used in another linguistic system with new meaning.
Loan-transation: "The native language uses an item-for-item native version of the original".
Loan-blends: Loan-blends consist of two elements. One element is a loanword and the other element is from native language [8]

2.2. Resyllabification

Both English and Urdu possess different syllable structure and phonotactics. Like all other languages, Urdu also imposes certain restrictions on the syllabification of borrowed words. Thus these English words have become part of Urdu language after undergoing the process of resyllabification [1].
However Trask (1996) defines resyllabification as "a process which applies during a derivation to move segments from one syllable to another" [15]. Usman, Farooq, & Masood (200) in their article concludes that words of English when spoken in Urdu, they undergo resyllabification and are resyllabified according to the templates and phonological rules of Urdu.
English and Urdu contain different syllable patterns [1].

2.3. Syllable

Syllables are considered to be the basic and an important phonological unit. The definitions of syllable seem ambiguous but a native speaker can easily identify the number of syllables in a word by just tapping the fingers. This shows the importance of the syllable in the rhythm of speech. Roach defines syllable in two ways. He defines a syllable phonetically and phonologically. On phonetic grounds, syllables are defined as the sounds having a centre with little or no obstruction to the flow of air and which have relatively loud sound. The other way of defining a syllable is on phonological grounds.
By phonological grounds we mean as to how a syllable is defined with respect to neighboring sounds. This takes the review of looking at the possible combinations of sounds in the production of a syllable. The study of the possible combination of phonemes in a language is termed as phonotactics. This involves the study of what can come at the beginning of a syllable, at the centre and at the end of the study of how many entities can come in combination to form a part of syllable [14]. The centre of the syllable is said to be loud and in other words more prominent than other sounds in the syllable. This is also termed as the theory of prominence which points out that some sounds in an utterance are more prominent or sonorous than other sounds. This forms the sonority hierarchy of sounds [4]. Trask also defines syllable by chest-pulse theory. According to which syllable is defined as an utterance produced as a result of single respiratory movement or a single opening and closing of respiratory tract [15]. A syllable is also defines as a smallest unit of speech which consists of a single vowel or can exist in combination with consonants [9].

2.4. Structure of Syllable

The syllable can thus have an optional consonant at the initial and final position with a mandatory vowel at the centre of the syllable. This gives vowel a CVC structure. The division of syllable that is now usually followed is the division of syllable into onset and rhyme, with rhyme further divided into nucleus and a coda as shown in figure 1 [15]. The following terminology is widely used:
Onset: the opening segment of a syllable, coda: the closing segment of a syllable and nucleus: the central segment of a syllable [5]. The structure of syllable can thus be drawn as follows:

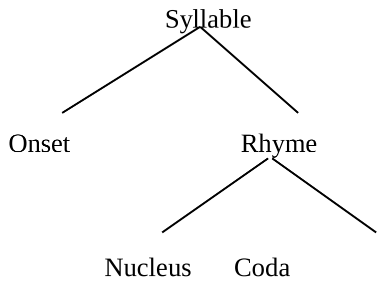


Figure 1: Syllable Structure

(Roach, 2003)

2.5 Syllable structure of English

English follows the same syllable structure. It differs on the basis of the different phonotactics rules.
Roach (2003) discusses the structure of syllable of English by starting with what comes at the onset of English words. According to roach, if a word start with a vowel, we can say that the word has zero onsets. And if the word has a single consonant at the start of the syllable or we can say that if the word has one consonant before the peak of the syllable, the consonant in the onset position will be referred as initial consonant.
In case of a consonant cluster having two consonants (CC) at the onset position, two sorts of combinations are

seen in English. One combination includes the combination of s with a small set of words /p, t, k, f, m, n/. The position of the s is named as pre-initial consonant position and the other words are said to be at the initial consonant position. The other sort of combination is when we see words like /pleɪ/, /traɪ/. These types of words begins with a set of about fifteen consonants followed by a small set of words /l, r, w, j / which are referred as post initial consonants.
In case of three consonant clusters, we have a distinct number of consonants that can come at the three positions; **s** at the pre-initial position, **p, t, k** at the initial consonant position and **l, r, w, j** at the post-initial consonant position [14].
A cluster of four consonants is seen at the end of the syllable. If we have no consonant at the coda position, we can say that there is a zero-coda. If there is just one consonant after the vowel at the end of the syllable, we call that consonant as final consonant. All consonant phonemes can take the position of the final consonant position except **h, r, w, j.** With a consonant cluster of two consonant at the end of the syllable we can have two combinations:

a) Pre-final consonant + final consonant
b) Final consonant + post-final consonant

The pre-final consonant position can be taken by a small set of consonants: **m, n, ŋ, l, s.** The examples of words having pre-final and final combination are **/bʌmp/, /bent/, /bænk/** etc. Similarly post-final position can be taken by the consonants: **s, z, t, d, θ.** The examples of final and post-final combination are **/bets/, /bedz/, bægd/** etc. we can have a combination of four consonants at the end of the syllable. With a combination of three consonants, we can have the following combinations:

a) Pre-final + final + post-final-consonant
b) Final + post-final 1 + post-final 2

Pre-final, final and post-final combination is seen in the following words: helped **/helpt/,** banks **/bænks/.** Post-final 1 and post-final 2 is found in the words: fifths **/fɪfθs/,** next **/nekst/.** With a consonant cluster of four consonants, we can have the following combinations:

a) Pre-final + final + post-final 1 + post-final 2
b) Final + post-final 1 + post-final 2 + post-final 3

The examples of (a) are seen in twelfths **/twelfθs/** prompts **/prɒmptd/** while (b) is found in sixths **/sɪksθs/** and texts **/teksts/.** [14]

2.6. Syllable structure of Urdu

Urdu is the national language of Pakistan and is spoken by almost 104 million people all around the world [6]. The vocabulary of Urdu is the result of the absorption of words and phrases from other languages and Urdu easily modify the words according to its own grammar [11]. The modification of these words is done by the process of resyllabification. For taking into account the process of resyllabification, we first need to study the syllable structure of Urdu language. Ghazali (2002) in his article writes that there is no as such significant research on the phonology of the Urdu language. However considerable work has been done on the syllable templates of Urdu. Nazar (2002) lists ten syllable templates that are found in Urdu language:

1. CVV
2. CVC
3. CVVC
4. CV
5. CVCC
6. VV
7. VVC
8. CVVCC
9. V
10. VCC

He also writes that Urdu uses both long and short vowels. He uses VV for a long vowel and V for a short vowel in the syllable. He, therefore, divides the above mentioned syllable templates into two groups based upon the presence of the vowel present.

CVV	CV
CVVC	CVC
CVVCC	CVCC
VV	V
VVC	VC
	VCC

Urdu language has a simple onset having a single consonant at the onset position. The presence of a short vowel at the start of the syllable is less favorable. At the start of the syllable, mostly long vowel exists. He also writes that Urdu also restricts a short vowel at the end of the word. The most popular templates in Urdu are CVV and CVC. At coda position, Urdu language has a consonant cluster of two consonants occurring together. Even this syllable template exists a little in Urdu language. The frequency mentioned by Nazar (2002) for the syllable template CVCC and CVVCC is 3.2 and 0.3 percent respectively which is much less as compared to the frequency percentage of the syllable templates CVV and CVC which is 39 and 20 percent respectively. Nazar (2002) declares the template CVVCC as super-super-heavy syllable. Nazar (2002) has worked out just the syllable templates of Urdu language [12].
Ghazali (2002) also lists eleven syllable templates of

Urdu in his article 'Urdu Syllable Templates':

1. CV
2. CVC

3. CVCC
4. CVV
5. CVVC
6. CVVCC
7. V
8. VC
9. VCC
10. VV
11. VVC

He however discusses that Urdu only has six syllable templates. The other syllable templates are derived ones. He further elaborates that there is no as such restriction on the consonants that can occur in the template CV however in templates if the consonant in the onset and coda position can be same if they belong to the set /t̪ /, /t /, /l/, /s/, /b/, /m/, /p/, /tʃ/. he further elaborates that by the reference of Hussain (1997) that when there are two consonant in a cluster, the first consonants is limited to voiceless fricatives or nasals and second consonant in that cluster will be limited to stops. Ghazali (2002) also enlists the consonants that can take the position of the first consonant in that consonant cluster: / l, z, r b, k, t̪ /. He also explains that CVV is the most frequent used syllable template used in Urdu. The possible reason that he gives to the question as to why CVV is more common as compared to CVC despite the fact that CVC is a complete template is the fact that saying CVV is easier as compared to the template CVC. In his conclusion, Ghazali (2002) argues that Urdu has only has six fundamental syllable templates (CV, CVC, CVV, CVCC, CVVC, CVVCC). Among these templates, CVV is most frequently spoken (37%) while template CVVCC is least (0.4%) used [6].

3. Methodology

The current investigation deals with the borrowed words having consonant clusters. A qualitative research was considered more suitable as a research design. The research was conducted in Lahore, Punjab where mostly Urdu is spoken as a native language and English is spoken as a second language. Owing to this importance of the language, it has affected the lexicon of Urdu language. A number of words from English language have become part of the lexicon of Urdu language. The population of the research was all those speakers who speak Urdu as their native language and learn English as second language. The sample was selected on the basis of the convenience in University of the Punjab. People from different fields of education were selected. On an average, data was collected from 8 students belonging to different educational background and four respondents were selected from the clerical staff of different institutes who have very less chance of speaking English as a second language.

As in qualitative research the main aim of data collection is saturation. After collecting data from this small sample, it was seen that there was no difference of results i.e. saturation was reached. The data collected from all the respondents was same. Words were selected from Urdu Dictionary “firoz-ul-lughat Jamiya” by Feroz Sons and a daily Urdu newspaper “Express”. From all the borrowed words, the borrowed words having consonant clusters were selected for the study. These borrowed words were selected on the assumption that place of articulation has any role on epenthesis in the consonant clusters. Each combination of consonant cluster was selected according to the manner of articulation. For pre-initial and initial position, combination of /s/ with every possible phoneme was selected. In case of initial and post-initial position, first the combination of voiced and voiceless plosives were seen with /l, r, w, j/ and one word for each combination was selected. The same was done for all manner of articulation i.e. fricatives, affricatives, nasals, lateral approximants and approximants.

The possible combinations of voiced and voiceless consonants were seen with post-initial consonants /l, r, w, j/ and one word having one of the combination was selected. The same was done at the **coda** position. After collecting these words, Urdu sentences were made using these words to avoid any interference of the English language during the pronunciation of these words. A closed room was selected for the recordings of the respondents to avoid any disturbance of noise. The respondents were asked to read the phrases thrice and they were recorded for the analysis of the pronunciation of the selected words. The pronunciation of the selected words was analyzed by three Urdu speaker researchers. In case of confusion PRAAT was used in analyzing any word that created confusion during the analysis of the results.

4. Analysis

The results of the research can be divided in to four groups:

i. Consonant clusters having different place of articulation:

The words having two consonants in consonant clusters at onset position showed epenthesis between them. The words that showed resyllabification in consonant clusters are speech, skirt, plaster, smuggle, sweeper, brown, and glass. The resyllabified pronunciation of the above mentioned words are: /səpi:tʃ/, /səkɜ:t/, /səmʌg(ə)l/, /səwiːpər/, /bəraʊn/ and /gəlɑ:s/ respectively. In case of “skirt” out of twelve respondents eight respondents inserted a vowel between /s/ and /k/ and pronounced the word as /səkərt/.

When the consonant cluster has three consonants in it, consonants having different place of articulation showed epenthesis in the first two consonants and a syllable break after them. The words in the data that showed this phenomenon are spring /səp.rɪŋ/, screen /sək.ri:n/, squash /sək.wɒʃ/.

ii. Consonant clusters at onset position having same place of articulation:

The words having consonant clusters in which the consonants shared the same place of articulation did not show epenthesis or resyllabification in them. This phenomenon was specific to only alveolar consonants. The words selected that showed these results are: staff, snow, slow, string. All the respondents pronounced these words without any epenthesis.

iii. Consonant clusters at onset position having /j/ at post-initial position:

The words having /j/ at post-initial position as in tube, duty, music, news, and lubrication also did not show any epenthesis or resyllabification in consonant clusters. All the respondents pronounced these words without any epenthesis.

iv. Consonant clusters at coda position:

The words having two or three consonant in consonant cluster at coda position did not show any epenthesis in coda position consonant clusters. The words that showed this result are: jump, tent, belt, risk, brand, gold, golf, conference, games, lunch, gift, script, tax, product, thanks, accounts, next, funds, torch. In all these words the consonant cluster structure remained intact.

Only one combination out of the selected words showed epenthesis at coda position i.e. /lb/. The word selected for this combination was **bulb** which all respondents pronounced as /bʌləb/. This strange result requires a detailed study.

5. Discussions

The research was started with the hypothesis that manner of articulation plays a role in the resyllabification of the consonant clusters. After collecting the data, the data showed that manner of articulation has no relation with resyllabification in consonant clusters.

The comparison of syllable structure of Urdu and English shows that Urdu also has a consonant cluster at coda like English. By looking at the results of the data during the research, it was seen that at coda position, a native Urdu speaker does not change the structure of the consonant cluster at coda position. The structure of the coda in the syllable of English borrowed words remains intact i.e. no epenthesis is seen at coda position in the English borrowed words in Urdu language. However the results show one exception in the form of epenthesis at coda position in "bulb" which all the respondents pronounced as /bʌləb/. This result requires a detailed study to find out the reason behind the epenthesis which was absent in all combinations in words selected according to manner of articulation.

However the research shows that epenthesis takes place at onset position with some exceptions. The consonants in the consonant cluster having same place of articulation did not show epenthesis in the consonant clusters as was seen in staff, snow, slow, street, and string. All these words have consonant clusters at onset position having consonants that have same place of articulation. It can be said that when consonants in a consonant cluster have same place of articulation, the tongue has to put less effort if the production of those sounds. We then see no insertion of vowel between consonant sounds that share same place of articulation.

Another interesting finding is seen when /j/ comes at post-initial position, no epenthesis is seen. This research is in contrast with the research conducted by Ahmed, Anwar, & Iqbal (2017) where they gave the result that an Urdu speaker always adds a vowel between consonant clusters at onset postion [1].

6. Conclusion

The findings of the research are:

1. The analysis of the words shows that epenthesis, which is the major reason of resyllabification in consonant clusters, is seen at onset position e.g. in case of words spring, skirt, shrink, etc. but in case of consonant clusters at coda position no epenthesis is seen like in words jump, belt, brown, glass, golf, games, tax etc except when the consonant clusters contains consonants /lb/ which all respondents pronounced as /ləb/ as the word used during the research 'bulb'. This strange result requires a detailed study.
2. In case of resyllabification at onset position, it is found that
   - Whenever the consonants in the consonant clusters have same place of articulation which is specific to alveolar position we see no epenthesis as was seen in words street, slow, staff, smuggle.
   - The other combination where we see no epenthesis is the combination of sounds with /j/. Whenever /j/ comes at post-initial position no epenthesis is seen. This phenomenon was seen in words news, duty, music, view, and human.

7. References

[1] Ahmed, S., Anwar, B., & Iqbal, T. (2017). Language Contact: a study of syllabic change of English Borrowed Wirds in Urdu. *International Journal of English Linguistics* , 140-147.
[2] Ashby, M., & Maidment, J. (2008). *Introducing Phonetic Science.* Singapore: Cambridge University Press.
[3] Collins, B., & Mees, I. M. (2013). *Practical Phonetics and Phonology.* Routledge.
[4] Cruttenden, A. (2011). *Gimson's Pronunciation of Englsih.* Hodder Education.
[5] Crystal, D. (2008). *Dictionary of Linguistics and Phonetics.* John Wiley & Sons, Incorporated.
[6] Ghazali, M. A. (2002). Urdu Syllable Templates.
[7] Hoffer, B. L. (2002). Language Borrowing and Language Diffusion: An Overview. *Intercultural Communication Studies XI:4* .
[8] Hoffer, B. L. (2005). Language Borrowing and Language Diffusion: An Overview. *Intercultural Communication Studies XI:4* .
[9] Kenworthy, J. (2000). *The Pronunciation of English: A workbook.* Oxford University Press.
[10] Lass, R. (1984). *Phonology: An Introduction to Basic Concepts.* Cambridge.
[11] Nazar, N. (2002). Syllable Templates In Urdu language. *Annual Report of Center for Research in Urdu Language Processing (CRULP)* .
[12] Nazar, N. (2002). Syllable Templates In Urdu language.
[13] Rehman, T. (2015). *Pakistani English.* National Institute of Pakistan Studies, Quaid-iAzam University, Islamabad.
[14] Roach, P. (2003). *Englsih Phnetics and Phonology.* Cambridge.
[15] Trask, R. L. (1996). *A Dictionary of Phonetics and Phonology.* Routledge.
[16] Usman, M., Farooq, S., & Masood, A. (n.d.). Syllabification od English Words when Spoken in Urdu. 49-53.
[17] Usman, M., Farooq, S., & Masood, A. (2002). Syllabification of English Words when Spoken in Urdu. *Annual Report of Center for Research in Urdu Language Processing (CRULP)* , 49-53.
[18] Yavas, M. (2011). *Applied English Phonology.* Wiley-Blackwell.

An Unsupervised Spoken Term Detection System for Urdu

Hafiz Rizwan Iqbal, Saad Bin Zahid, Agha Ali Raza
Information Technology University, Lahore, Pakistan
{rizwan.iqbal, mscs15019, agha.ali.raza}@ itu.edu.pk

Abstract

Over the past 40 years, Keyword Spotting (KWS) remained in focus by both academia and commercial companies. However, the majority of these systems were developed and evaluated for rich resourced languages like English, German, etc. This is because it requires thousands of hours of transcribed speech data to train KWS systems, which is not available for most of the under-resourced languages like Urdu. To address this challenge, the area of zero-resource or unsupervised speech processing emerged, i.e. to extract meaningful features and learning language structures directly from unlabeled raw speech data. This paper presents a completely unsupervised KWS system that searches all of the instances of an input keyword in reference audio file(s), given the keyword present in the reference file(s), without requiring any labeled data and speech recognition. PRUS corpus was used to train GMM without any supervision. Input keyword and reference audios Gaussian Posteriorgrams were compared using Segmental Dynamic Time Warping (SDTW). Top N minimum distances were taken to obtain the closely related segments of the reference file, which are more probable to be the desired keyword. The proposed system showed the precision up to 91.50 % and 79.20 % for cross-speaker and same speaker respectively.

1. Introduction

Spoken Term Detection (STD) a.k.a. Keyword Spotting (KWS) is a task of automatically detecting a spoken term (referred to as Query) along with its location within a continuous speech. It is on the rise due to its variety of applications such as shortlisting of audios from large repositories of online lectures like Coursera [27], conference recordings (e.g. TED talks), radio and television archives. Wake-word applications (to activate or initiate a voice interaction with devices), phone call monitoring and routing are some other important applications of KWS.

STD remained a hot research topic for more than four decades, and a lot of methods have been proposed which can be categorized into 1) Large Vocabulary Continuous Speech Recognition methods (LVCSR) – used for audio indexing and speech data mining, 2) Keyword/Filler Methods a.k.a. Acoustic Keyword Spotting and 3) Query-by-Example (QbyE) method. However, the majority of these techniques were developed and evaluated for resource-rich languages like English, German, etc. because of their reliance on thousands of hours of transcribed audio data. For example, in traditional Keyword/Filler models, word/phone level transcribed data is required to train a speech recognizer [8] [12] [21].

Unfortunately, such resources are not available for many of the world's languages such as Urdu. With the recent development of the internet, media technologies and smartphones, it is quite easy to obtain audio data than the transcription work. It's not only a time taking activity; but also requires a reasonable level of linguistic knowledge for performing annotations. This is the reason that most of the academic and commercial organizations develop STD systems for a few hundred languages [26].

As we are living in a digital and communication age in which digital media can be produced and gathered at a pace that far surpasses our capacity to transcribe it, a common question "how much can be directly learned from the speech signals alone, without any supervision"? In addition to this, speech applications are becoming popular and available for many languages on the cost of increasing method complexities and their dependency on transcribed resources [23], it is difficult to envision that the required resource collections would cover all 7,000 human languages around the globe [2]. This makes a related query that "what unsupervised techniques can be performed well in contrast to the traditional supervised training techniques". This creates a related question "what techniques can be performed well using unsupervised techniques in comparison to more conventional supervised training methods". Our motivation to answer these two questions lead us to explore the development of an unsupervised STD system for a low resource language Urdu.

Urdu is the 6th most popular Asian language, the national language of Pakistan and the authorized language of 6 Indians states with more than 175 million speakers all over the world [31]. As the national language of Pakistan, most of the educational material, radio and television programs, and conversational audios are available in Urdu. This plethora of available speech files creates a need for an efficient KWS for Urdu language. Limited efforts have been made in the past [17], but unfortunately, there are no publicly available automatic KWS for Urdu.

This paper presents an unsupervised STD system for Urdu language. To train the model from an unlabeled speech data, a Gaussian Mixture Model (GMM) was used to represent each audio frame with a Gaussian Posteriorgram (GP) vector, and a Segmental Dynamic Time Warping (SDTW) method is used to compare the GPs of the spoken query term (hereinafter called *Needle*) and the target speech utterance (hereinafter called *Haystack*) [36] to find one or more occurrences of the *needle*.

In addition to this, the proposed KWS system searches all of the instances of the *needle* with their locations in the *haystack(s)* without doing speech recognition, given the keyword is present in the reference file. For this purpose, a Phonetically Rich Urdu Speech (PRUS) Corpus [37] used to cluster speech frames without any transcribed data. Top *N* minimum distances were then taken to get the closely related segments of the *haystack* file with the assumption that these speech frames are the most probable to be the desired keyword. The proposed system showed the *precision* up to *91.50%* and *79.20%* for cross and the same speaker respectively, given the *needle* is present in the *haystack*.

2. Literature Review

STD has been a hot research area over the past 4 decades but in recent years STD has received increased attention by both academia and commercial communities [38]. Chen et.al [3] summarizes STD past research efforts and encapsulates proposed methods in three categories.

The first category defines Large Vocabulary Continuous Speech Recognition (LVCSR) based methods. LVCSR based methods have been extensively used in audio data mining and indexing and found to be well accurate for a variety of tasks [39]. Continuous speech files are transcribed into words using Automatic Speech Recognizer (ASR) and then text-based searching techniques used for efficient spotting of the required keywords [40].

The second type of STD methods are Keyword or Filler methods *aka* Acoustic Keyword Spotting, models the keywords and non-keywords using Hidden Markov Models (HMMs) and spotting is made through the decoding graphs where keywords and fillers appear in parallel [42] [43] [44]. This type of KWS mostly used in scenarios where keywords are pre-defined and speech data comes in real-time. Such types of applications are like voice commands and wake word applications (e.g. Hey Siri, Ok Google, etc.). Ketabdar et.al [14] proposed a system that used the HMMs posterior based scoring approach for keyword and non-keyword elements [7]. For each frame, the state posteriors are combined with the posteriors of keyword and non-keyword to identify the keyword for each frame resulted in identifying the presence of the keyword in the whole utterance.

Query-by-Example (QbyE), is the third type of techniques developed for the development and evaluation of STD systems. QbyE is one of the earliest KWS methods [32], have two main steps including 1) *template representation* – how audio files of *needle(s)* are to be represented (e.g. in the form of lattices or posteriorgram feature vectors, etc.), and 2) *template matching* – how needles are to be matched with the *haystack* to find the desired *needle*. Various research efforts have been over the past decades [28] [33] [34] for unique *template representation* methods and variants of Dynamic Time Warping (DTW) [20] used for *template matching* phase [26].

The recent resurgence of *Neural Networks* (NN) as *Deep Neural Network* (DNN) gives a high rise to the KWS research area. Recently, Abdulkader et.al [1] proposed a model for KWS in narrowband audio, for computationally constrained devices by making use of DNNs, cascading, multiple-feature representations, and multiple-instance learning. In order to reduce the rate of false positives, they trained two classifiers on two different representations, Mel Frequency Cepstral Coefficient (MFCC) and Perceptual Linear Prediction (PLP) features. Moreover, Chen et.al [3] proposed a novel QbyE-STD method using Long Short-Term Memory (LSTM) based feature extractor. They showed that their presented KWS approach has low computation cost with high precision, can be efficiently used for small computational power devices.

Although, all of the above described methods shown to be very effective for the KWS task, assume the availability of large quantities of labeled speech data for training and testing of complex statistical language and acoustic models. For instance, one major drawback of LVCSR based KWS systems is Out-Of-Vocabulary (OOV) words, which is the main reason that LVCSR based methods best performed for well-resourced languages [30][39][40]. Similarly, Keyword/Filler based methods require prior knowledge of keywords and non-keyword elements to build special decoding graphs [3]. Chen et.al [3], to train DNN based KWS system, 19,000 audio files from 200 individuals used as positive examples whereas for negative examples a repository of audio samples of various meeting recordings were used. QbyE techniques normally take thousands of examples of *needles*, decoded by using ASR to acquire their lattice representation as templates and make detection decisions by comparing them against the *haystacks*. Moreover, the available techniques are computationally expensive due to their base on ASR. Therefore, these KWS methods were not suitable in low-resource contexts, and the reasons commercial firms focus on a few hundred languages of the world.

Transcription of the speech files, a major barrier in producing resources for under-resourced languages because it is not only an expensive process but also a time-consuming task. To address this challenge, the area of zero-resource or unsupervised speech processing emerged, by extracting meaningful features and learning language structures directly from unlabeled raw speech data [9][11][26][33][35]. With the advancements of Internet and multimedia technologies, it is quite easy to get audio data without transcription which makes it possible to develop speech processing solutions for under-resourced languages such as Urdu.

In the past, there are limited research efforts for the development of KWS for Urdu language. Irtza et.al [17] reported a KWS for Urdu language using filler modeling to compute non-keyword elements. A *phoneme recognizer* (PR) was used to model all phones. The audio input file is processed using PR and KWS, an achieved overall accuracy of 94.59%. Another work [45] also has been carried out for Urdu KWS task, but was developed only for five words of Urdu and achieved an accuracy level of 98.1%.

As far as our background knowledge and literature review, currently there is no completely unsupervised publically available KWS system developed for Urdu language because there are very limited standardized audio dataset which can be used for the development and evaluation of the KWS for Urdu language. Keeping in view the high demand of Urdu KWS system, we have developed a completely unsupervised QbyE-STD system (using the baseline approach proposed in [26]) by using the currently limited available gold-standard resources [37].

3. Methodology

We have developed an unsupervised STD system for Urdu that output all the occurrences of a *needle* (Q) in a given *haystack* (X), provided the input keyword (i.e. *needle*) is already present in the reference audio file (i.e. *haystack*). Our approach is most similar to the one proposed in [26] with the difference that we have used this approach and tune the parameters for Urdu language which is more phonetically rich than the English. Instead of using any phoneme recognizer, raw speech files were modeled using a Gaussian Mixture Model (GMM) without any supervision and get Gaussian Posteriorgram (GP). Segmental DTW (SDTW) was used to compare the distance between *Q* and *X* and generate the list of minimum distances in descending order.

Figure 1 illustrates the abstract level architecture of the developed KWS for Urdu. The acoustic model was trained, resulted in GPs of the training data. GMM was applied to get GPs of both of the *Needle* Q and *haystack* X. SDTW window was moved over the X and get occurrences (x1, x2...) of Q in X. This task is done without doing any explicit speech recognition.

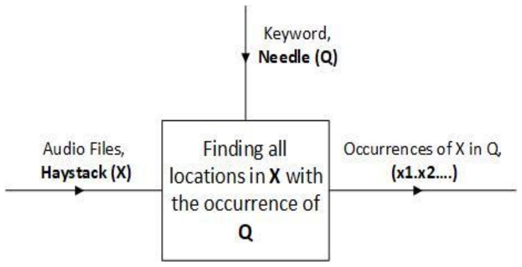


Figure 1: High-level architecture of the system.

3.1. Gaussian Mixture Model (GMM)

Posteriorgram is basically a probability vector which is used to represent the probabilities of the Gaussian components in a given speech frame. It is mostly used in the phonetic posteriorgram. Formally, if we represent speech by *n* frames:

$$S = (s_1, s_2, \dots, s_n) \qquad (1)$$

The Gaussian probability vector is defined as in [26]:

$$GP(S) = (q_1, q_2, \dots, q_n) \qquad (2)$$

Figure 2 demonstrates the process of computing GP vectors of both Q and X. Acoustic model was obtained by applying GMM on each frame of each audio file in the training data, to get a raw GP vector of each frame. This becomes a critical task when you do not have any labeled data. As reported in [26], training was performed by assuming that there are the same labels on all frames of the dataset which induces a problem of not discriminating between phonetic units in the posteriorgrams vector. Probability distribution on a large mass is concentrated on some dimension and the remaining dimensions have very little probability. To solve this problem, a speech/non-speech detector was applied to the training data by extracting the MFCC’s and then GMM on them.

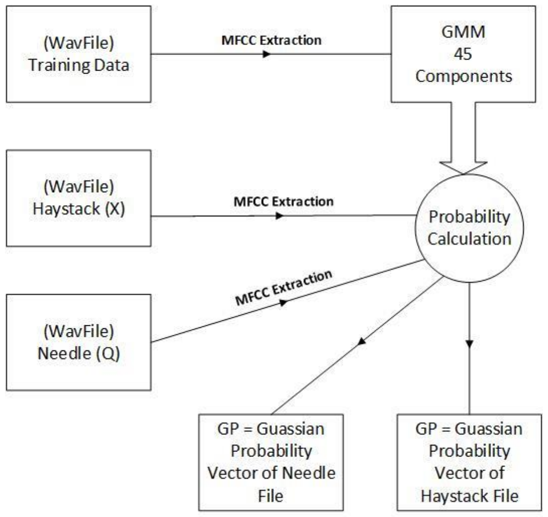


Figure 2: Computing Gaussian Posteriorgrams Vectors using GMM

Each element in GP(s) is representing a vector which can be calculated by using GMM. For example any q_j = $(P(c_1js_i),P(c_2js_i),---,P(c_mjs_i),)$, where c is representing the components of GMM and m is the size of Gaussian components. In this case, there are 45 Gaussian components which clustered the training data into 45 clusters. For *needle* Q and *haystack* X, probabilities were computed with respect to 45 clusters resulted in the probability vector of size 45 for each frame. Hence, the GP matrix *M* size is *number of frames time's Gaussian components (Matrix Size* (M) = *No. of frames* $*$ *Gaussian components)*

For both audio files of *needle* Q and *haystack* X, each speech file divided into windows *aka* frames of 25 *msec* along with the overlapping step size of 10 msec, to avoid missing any information of the signal at the window boundaries. For each frame, the probability vector of size 45 was obtained by passing it through the GMM processor to make it GP vector. Figure 3 provides the visualization of *needle* Q framing and its respective GP vector with 45 GP elements. Similarly, the probability vector of size 45 for *haystack* X will be computed, and the visual representation of file *X* is similar to *Q* file.

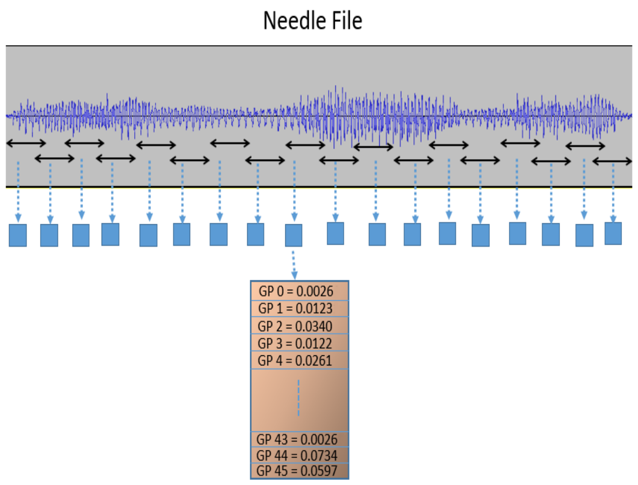


Figure 3: Visualization of needle file.

3.2. Segmental Dynamic Time Warping (SDTW).

SDTW is the modified version of well renowned DTW algorithm [20], and has demonstrated its success in unsupervised pattern discovery in audio files [26] [45] [47]. It works by finding the distance between the elements of both (*needle* and *haystack*) signals and then finds a path with minimum distance between these elements. To find Q in X, SDTW was applied to GP vectors of both Q and X. The distance between two GP vectors computed using equation (3):

$$D = -log(p.q) \qquad (3)$$

Where p and q are two posteriorgram vectors. As both *p* and *q* are probability vectors, dot product was used as a similarity measure to find the distance between them. By applying SDTW, there is a need to handle the following two constraints: 1) Adjustment window condition and 2) the step length of the start coordinate of the DTW search [26]. Fixation of the adjustment window size will restrict the shape and ending coordinate of the warping path, but if use different starting coordinates then the warping path will be automatically in the diagonal regions of the DTW grid. Therefore, we used overlapping window strategy and every time move window (adjustment window size) *R* steps for the next search. The reason for using the overlapping window is to avoid redundant computation and to check the warping path across the boundary of segments. Size of *Q* is fixed in this case and just need to care about the segments of the *X*. Window will be moved R steps forward in *X* and *no of warping path* (by equation (4)) will come as an outcome, where each path represents the warping between *Q* and *X*.

$$\frac{(n-1)}{R} \quad (4)$$

3.3 System Flow

Figure 5 demonstrates the flow of the reported Urdu KWS system. The steps are as follows:

1. Raw input speech of both Q and X given to the system.
2. Remove the silence from Q and X by using Voice Activity Detector (VAD), because while comparing Q with the frames of X, the silence was also compared ended up in false results.
3. MFCC (i.e. 13 coefficients) vectors are extracted from Q, X and training data.
4. GMM is applied to the MFCC vectors of training data (audio file of about 1 hour speech) to make the optimum number of phonetic clusters.
5. GP vectors (as shown in Figure 3) of MFCCs are calculated for both Q and X.
6. By taking overlapping frames from the GP vector of Q and X, SDTW is applied using the dot product (cosine similarity [15]) as the distance measuring method.
7. Results are sorted in ascending order of cost.

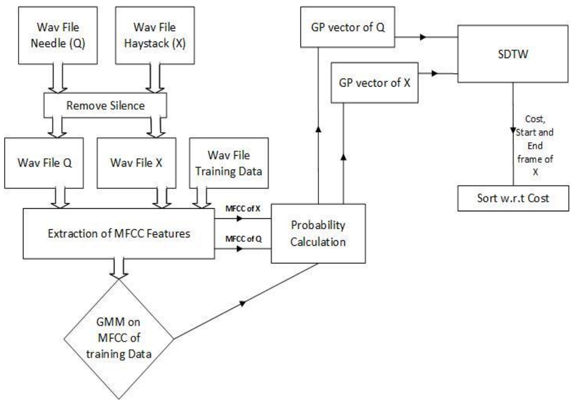


Figure 5: The system flow diagram

4. Experimental Setup

4.1. Types of Experiments

Two types of experiments were performed including 1) **Same Speaker** – training and testing audio files are in the voice of the same speaker, and 2) **Cross Speaker** – training audio files speaker is different from the test audio files speaker. For the same speaker experiments, 15 words (i.e. *needles*) were selected whereas for cross speaker experiment, 2 words were selected.

4.2. Dataset

For the development and evaluation of the proposed KWS system for Urdu language, PRUS [20] corpus was used. It is not as larger as the other available benchmark speech corpora for English (e.g. TIMIT [48], Librispeech [49], etc.), but for Urdu it is the only publically phonetically rich (covers almost all of the Urdu language sound) gold standard corpus. It contains 708 audio files in .wav format, in total 90 minutes of Urdu speech.

4.3. Training and Testing Data

For the same speaker experiments: for each needle, all of the audio files in the dataset were used for training of the GMM except for those files contained the selected needle in the respective experiment. For instance, the word "پاس" (occurred in 8 audio files out of 708) is selected as a *needle* in experiment number 1. GMM will be trained in those 700 audio files which do not contain the selected *needle*. Now, out of the 8 files having the *needle* in each file, 7 files will be selected as *haystack* files whereas *needle* word will be extracted from the remaining 1 audio file.
For cross speaker experiments: all of the 708 audio files were used to train GMM, whereas 2 words were recorded from another speaker as a *needle*.

4.4. Evaluation Measures

To evaluate the developed unsupervised KWS for Urdu language, Precision (P) was used as an evaluation measure because of the constraint "needle(s) must be present in the haystack(s)". This implies that only true positives and false positives can be computed for the proposed system. We have P@N [8]: the precision of the top N hits, where N is the number of occurrences of a needle in the haystack.

5. Results and Analysis

The summarized results of the experiments for same and cross speaker are shown in Table 1 and Table 2 respectively, where N shows the number of needles' occurrences, P@N shows the precision of finding N number of Q instances in X, and P@3 represents the precision of locating Q in those X files which contains 3 occurrences of Q. Similarly P@5 and P@10 indicates precision of respective X files. Cells with value 'NA' show that there is no X available with the required number of occurrences for that specific Q. The last row in both tables shows the average precision of all experiments.

It is clear from Table 1 that the average P@3 (82.30 %) outperforms whereas the mean P@10 (76 %) performed worse than all other. It can also be seen that the average P@N (79.20 %) is comparative to that of P@5 (80 %). It is clear from the average precision results and individual keyword results that the precision decreases as the number of occurrences of a needle increases. Another possible reason for this precision degradation could be the middle vowels, as GMM performed best on 45 phonetic clusters although there exist 67 different phonemes in Urdu language.

Table 1: Summarized experimental result for same speaker.

Needle	N	P@N	P@3	P@5	P@10
پاس	8	0.72	0.83	0.83	NA
ایک	28	0.7	0.75	0.6	0.55
بعد	26	0.9	1	1	0.71
نہیں	15	0.71	0.42	0.5	0.62
جاتا	6	1	1	1	NA
صاحب	6	0.75	0.75	0.71	NA
غیر	5	0.6	0.625	0.6	NA
ساتھ	23	0.85	1	1	0.9
والے	6	0.75	1	0.55	NA
پہلے	6	0.6	0.5	0.71	NA
کیلئے	14	0.73	0.75	0.71	1
جانے	6	1	1	1	NA
کرنے	11	0.73	0.75	1	0.71
سامنے	5	1	1	1	NA
بغیر	10	0.83	1	0.83	0.83
Avg. Precision		**79.20%**	**82.30%**	**80%**	**76%**

The cross speaker experimental results are shown in Table 2. It can be seen that the average P@5 (100%) outcompeted all other average precisions (i.e. 91.5 %). The proposed system located the needle "قسطنطنیہ" perfectly in all settings. The possible reason could be the uniqueness of the needle due to its larger phonemic counts, quite unique phonetic sequence, and this word is not commonly used in conversational Urdu speech. Whereas the needle "بغداد" correctly spotted only when the number of occurrences is 5 while in other settings the performance is decreased. The shorter phonemic length and common phonetic sequence could be the probable reasons for this low precision.

The proposed KWS performed better in cross speaker settings as compared to the same speaker. One obvious reason could be the number of needles chosen for the experiments, which are too less in cross speaker scenario. Another important observation for this low precision in same speaker context, could be the length of phonemes in each needle as in same speaker experiments the average needle length is 3 whereas in cross speaker settings it is 9 which implies that needles with shorter phonemic counts are harder to locate as compared to the needles with larger phonemic count.
It has also been found that the produced results seem to strongly dependent upon the type (unique) of words. Words present as substring may increase false-positive results. For example, the word "Tania" and "Aania" are almost the same because "Aania" is present as a substring. Our system saying these words are the same and that is not true. As far as our domain knowledge and literature review, this is the first attempt to develop an Urdu KWS system in a completely unsupervised manner. The initial results demonstrate that there is still a big room available to improve Urdu KWS.

Table 2: Summary of results of cross speaker experiments

Needle	N	P@N	P@3	P@5	P@10
قسطنطنیہ	10	1	1	1	1
بغداد	10	0.83	0.83	1	0.83
Avg. Precision		**91.5%**	**91.50%**	**100%**	**91.50%**

6. Conclusion and Future Work

Availability of the large datasets is a crucial requirement for the majority of the existing KWS techniques as they require huge datasets to train the model, which makes these methods unsuitable for low resource languages like Urdu. Keeping in view the high demand for the KWS system for Urdu, this paper reports an unsupervised STD system for Urdu.
Without any transcription, the model is trained by extracting MFCCs directly from speech files. GMM is applied to the training data to make the phonetic clusters, and generate GPs for both of the needle and haystack. Segmental DTW, a modified version of the well renowned DTW signal alignment method, used to compare the GP vectors of the input keyword and the reference audio file. Warping path with minimum score indicates the frames associated with this path are closer to each other. Experiments were performed for both same and cross speaker settings, and observed that the proposed system performed better in cross speaker scenarios as compared to the same speaker context.

The proposed system has some limitations such as 1) the major constraint "given the word is present in haystack", 2) it reports all the occurrences of a needle in a given haystack but, it does not tell either the word is present or not, and 3) length normalization of vectors because DTW returns different scores for different lengths of vectors. To overcome all of these limitations is the future goal of this work to obtain more satisfactory results along with examining this method on other low resource regional languages such as Pashto or Punjabi.

7. Conclusion and Future Work

The code and the dataset described in this article are freely available for research purposes and can be downloaded from https://github.com/ab-101/Key-Word-Spotter.

8. Bibliographical References

[1] Abdulkader, Ahmad, Kareem Nassar, Mohamed Mahmoud, Daniel Galvez, and Chetan Patil. "Multiple-Instance, Cascaded Classification for Keyword Spotting in Narrow-Band Audio." arXiv preprint arXiv:1711.08058 (2017).

[2] C. Richard C. A. Rose and B. Douglas. "A Hidden Markov Model Based Keyword Recognition System". In *Acoustics, Speech, and Signal Processing, ICASSP-90,*, pages 129–132, IEEE, 1990.

[3] G. Chen, C. Parada, and T. N. Sainath. "Query-by Example Keyword Spotting using Long Short-Term Memory Networks". In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5236–5240, April 2015.

[4] S. Cox and R. Rose. "A confidence measures for theˆ switchboard database. In *Proceedings of ICASSP*, volume 1, pages 511–515, 1996.

[5] S. Das. ”speaker dependent bengali keyword spotting in unconstrained english speech”. 2005.

[6] Samarjit Das. Speaker dependent Bengali keyword spotting in unconstrained English speech acknowledgement. 2005.

[7] B. Samy H. Ketabdar, V. Jithendra and B. Herve. Posterior based keyword spotting with a priori thresholds. In *Ninth International Conference on Spoken Language Processing*, 2006.

[8] T. J. Hazen, W. Shen, and C. White. Query-by-example spoken term detection using phonetic posteriorgram templates. In *2009 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 421–426, Nov 2009.

[9] M. Huijbregts, M. McLaren, and D. van Leeuwen. Unsupervised acoustic sub-word unit detection for queryby-example spoken term detection. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4436–4439, May 2011.

[10] P. Matejka L. Burget M. KarafiA¡t I. Szoke, P. Schwarz˜ and J. Cernocky. Phoneme based acoustics keyword spotting in informal continuous speech. In *International Conference on Text, Speech and Dialogue*, pages 302–309, Berlin, 2005.

[11] Aren Jansen and Kenneth Church. Towards unsupervised training of speaker independent acoustic models. pages 1693–1692, 01 2011.

[12] Jochen Junkawitsch, L. Neubauer, Harald Hoge, and¨ Gunther Ruske. A new keyword spotting algorithm¨ with pre-calculated optimal thresholds. *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96*, 4:2067–2070 vol.4, 1996.

[13] Jochen Junkawitsch, L. Neubauer, Harald Hoge, and¨ Gunther Ruske. A new keyword spotting algorithm¨ with pre-calculated optimal thresholds. *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96*, 4:2067–2070 vol.4, 1996.

[14] Hamed Ketabdar, Jithendra Vepa, Samy Bengio, and Herve Bourlard. Posterior based keyword spotting with a priori thresholds. 01 2006.

[15] R. Dehak P. Dumouchel N. Dehak, P. Kenny and P. Ouellet. Front-end factor analysis for speaker verification. *Trans. Audio, Speech, Lang. Process., vol. 19, no. 4*, pages 788–798, IEEE 2011.

[16] R. Rose and D. B. Paul. A hidden markov model based keyword recognition system. In *Acoustics, Speech, and Signal Processing ICASSP-90*, pages 129–132, IEEE, 1990.

[17] Irtza, S., Rehman, K., and Hussain S. ”Urdu Keyword Spotting System using HMM,”. 2012.

[18] I. Zaharakis S. Kotsiantis, Sotiris B. and P. Pintelas. Emerging artificial intelligence applications in computer engineering 160. In *Supervised machine learning: A review of classification techniques*, pages 3–24. 2007.

[19] T. Hain D. Kershaw G. Moore J. Odell D. Ollason D Povey V. Valtchev S. Young, G. Evermann and P. Woodland. *HTK Book*. Microsoft Corporation and Cambridge University Engineering Department, 3.2.1 edition, 2002.

[20] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. In *Proceedings of Trans. on Acoustic Speech, and Signal Processing ASSP 26*, pages 43–49, IEEE 1978.

[21] Wade Shen, Christopher M. White, and Timothy J. Hazen. A comparison of query-by-example methods for spoken term detection. In *INTERSPEECH*, 2009.

[22] Gyorgy Szasz¨ ak and Andr´ as Beke. Using phonological´ phrase segmentation to improve automatic keyword spotting for the highly agglutinating hungarian language. In *INTERSPEECH*, 2013.

[23] Igor Szoke, Petr Schwarz, Pavel Matejka, Lukas Burget, Martin Karafiat, Michal Fapso, and Jan Cernocky. Comparison of keyword spotting approaches for informal continuous speech. pages 633–636, 01 2005.

[24] T. Kemp T. Schaaf. Confidence measures for spontaneous speech recognition. In *Proceedings of ICASSP*, volume 2, pages 887–890, 1997.

[25] Javier Tejedor and JosA˜ c ColA¡s. Spanish keyword˜ spotting system based on filler models, pseudo n-gram language model and a confidence measure. 01 2006.

[26] Z.Yaodong and J. R. Glass. Unsupervised spoken keyword spotting via segmental dtw on gaussian posteriorgrams. In *Automatic Speech Recognition Understanding, IEEE workshop on ASRU*, pages 398–403, 2009.

[27] https://www.coursera.org/

[28] Ao, Chia-Wei, and Hung-yi Lee. "Query-by-example spoken term detection using attention-based multi-hop networks." In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6264-6268. IEEE, 2018.

[29] Chung, Yu-An, and James Glass. "Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech." arXiv preprint arXiv:1803.08976 (2018).

[30] Chen, Guoguo. "Low Resource High Accuracy Keyword Spotting." PhD diss., Johns Hopkins University, 2016.

[31] https://en.wikipedia.org/wiki/Languages_of_South_As ia

[32] Bridle, John S. "An efficient elastic-template method for detecting given words in running speech." In Brit. Acoust. Soc. Meeting, pp. 1-4. 1973.

[33] Huijbregts, Marijn, Mitchell McLaren, and David Van Leeuwen. "Unsupervised acoustic sub-word unit detection for query-by-example spoken term detection." In 2011 IEEE international conference on Acoustics, speech and signal processing (ICASSP), pp. 4436-4439. IEEE, 2011.

[34] Shen, Wade, Christopher M. White, and Timothy J. Hazen. A comparison of query-by-example methods for spoken term detection. MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 2009.

[35] Wang, Haipeng, Tan Lee, and Cheung-Chi Leung. "Unsupervised spoken term detection with acoustic segment model." In 2011 International Conference on Speech Database and Assessments (Oriental COCOSDA), pp. 106-111. IEEE, 2011.

[36] Zhang, Yaodong, and James R. Glass. "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams." In 2009 IEEE Workshop on Automatic Speech Recognition & Understanding, pp. 398-403. IEEE, 2009.

[37] Raza, Agha Ali, Sarmad Hussain, Huda Sarfraz, Inam Ullah, and Zahid Sarfraz. "Design and development of phonetically rich Urdu speech corpus." In 2009 oriental COCOSDA international conference on speech database and assessments, pp. 38-43. IEEE, 2009.

[38] C. Chelba, T. Hazen and M. Sarac¸lar, "Retrieval and browsing of spoken content," IEEE Signal Processing Magazine, vol. 24, no. 3, pp. 39–49, May 2008.

[39] D. Miller, et al, "Rapid and accurate spoken term detection," in Proc. Interspeech, Antwerp, Belgium, 2007.

[40] J. S. Garofolo, C. G. Auzanne, and E. M. Voorhees, "The TREC spoken document retrieval track: a success story," NIST SPECIAL PUBLICATION SP, no. 246, pp. 107–130, 2000.

[41] M. Sarac¸lar and R. Sproat, "Lattice-based search for spoken utterance retrieval," in Proc. HLT-NAACL, Boston, 2004.

[42] A. Mandal, K. P. Kumar, and P. Mitra, "Recent developments in spoken term detection: a survey," International Journal of Speech Technology, vol. 17, no. 2, pp. 183–198, 2014.

[43] R. C. Rose and D. B. Paul, "A hidden Markov model based keyword recognition system," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP). IEEE, 1990, pp. 129–132.

[44] J. Wilpon, L. Miller, and P. Modi, "Improvements and applications for key word recognition using hidden Markov modeling techniques," in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP). IEEE, 1991, pp. 309–312.

[45] Juang, Biing-Hwang. "Recent developments in speech recognition under adverse conditions." In First International Conference on Spoken Language Processing. 1990.

[46] A. Park and J. Glass,"Unsupervised pattern discovery in speech", in IEEE Trans. ASLP, 6(1), 1558–1569, 2008.

[47] Chan, Chun-an, and Lin-shan Lee. "Unsupervised spoken-term detection with spoken queries using segment-based dynamic time warping." In Eleventh Annual Conference of the International Speech Communication Association. 2010.

[48] Garofolo, John S. "TIMIT acoustic phonetic continuous speech corpus." Linguistic Data Consortium, 1993 (1993).

[49] Panayotov, Vassil, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. "Librispeech: an ASR corpus based on public domain audio books." In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5206-5210. IEEE, 2015.

Comparison of Parsers Dealing with Text Ambiguity in Natural Language Processing

Sareya Qayyum, Nimra Aziz, Waqas Anwar, Usama Ijaz Bajwa
Computer Science
COMSATS University Islamabad, Lahore Campus
{sareyaqayyum, nimraiftikhar1995}@gmail.com, {waqasanwar, usamabajwa}@cuilahore.edu.pk

Abstract

Parsing in Natural Language processing is a vast domain which serves as a pre-processing step for different NLP operations like information extraction, etc. Multiple parsing techniques have been presented until now. Some of them unable to resolve the ambiguity issue that arises in the text corpora. This paper performs a comparison of different models presented in two parsing strategies: Statistical parsing and Dependency parsing. The comparison has been made on very famous Penn Treebank corpus specifically involving its Wall Street Journal Portion.

1. Introduction

In Natural Language Processing (NLP), Parsing acts as an essential and key component to many problems. Parsing is the analysis of syntax or commonly called syntactic analysis in which we process the sentences following the rules of a formal grammar. Parsing involves uncovering the meaning, content and underlying structure that makes up a sentence. Every Language has a unique grammar thus making parsing process unique as well. Languages can be divided into two categories:
Segmented Languages: Those languages whose words are space-delimited e.g. English and Spanish language
Un-Segmented Languages: In un-segmented Languages word segmentation is required as a pre-processing step before further processing E.g. Chinese and Japanese Languages

A language is not just a 'bag of words' or else there would be no need for grammar. Grammatical rules apply to sentences where a sentence in a language is a group of strings that consists of two things: a subject and a predicate. A subject is defined as a Noun Phrase (NP) and predicate as a verb phrase (VP). E.g. In an English Sentence 'Sam went to school', 'Sam' is NP and 'went' is VP. Parsing has many applications in Natural Language Processing. For Example, Machine translation, text summarization and question answering are few areas of NLP in which immense work is being performed, etc. Parsing serves as an initial step for these problems. The result of parsing a sentence using a formal grammar is a tree structure. A sentence can have exactly one or many such tree structures. The grammar that is usually used is CFG (context-free grammar).
One of the major challenges in Parsing is dealing with ambiguities in the sentence. Ambiguity refers to sentences that are subjective, open to interpretation and can have multiple meanings. Three types of ambiguities are present in a sentence when parsing is performed namely Syntactic Ambiguity, Lexical Ambiguity, and Semantic Ambiguity.
Syntactic Ambiguity: Sentences can be parsed in multiple syntactical forms. E.g. 'I heard his cell phone ring in my office'. The Phrase 'in my office' can be parsed in a way that modifies the noun or vice versa modifies the verb.
Lexical Ambiguity: Sentences having Lexical ambiguity can have words with multiple assertions. E.g. 'book' is used as a noun when used in a sentence as 'He loves to read books'. On the other hand, it can also be used as a verb when used in a sentence as 'He books an appointment at the dentist'.
Semantic ambiguity: It is related to the interpretation of the sentence. E.g. 'I saw a man with the telescope'. It can be deduced as if I saw a man holding a telescope. Or I saw a man through a telescope. Parsing of sentences happens in multiple stages:
a) Dividing a sentence into tokens. These tokens are used as input to some other tasks like parsing.
b) Tagging each token with parts of speech.

Eight parts of speech are observed in the English language – verbs, nouns, pronouns, adverbs, adjectives, conjunctions, interjections and prepositions.

1.1. Types of parsing

Two main types of parsing will be discussed in this paper and a comparison of performances for both the types will be made. Following are the types:

i. Statistical Parsing
ii. Dependency Parsing

1.1.1. Statistical Parsing

In Natural Language processing statistical parsers are the type of parsers which associate grammar rules with probability. The Statistical parser is an algorithm that looks for a tree that maximizes the probability P (T|S). PCFG (Probabilistic

Context-Free Grammar) which is an extended form of CFG is used as an underlying grammar. The probability of a parse tree of a sentence can be computed by firstly calculating the probability of the productions used in the derivation of the tree and then taking the product of these probabilities. Following three main tasks are involved in statistical parsing:

1. Determine the likely parse trees for the sentence.
2. Assign probabilities to each derived parse
3. Select the most probable parse (highest probability)

Statistical parsing requires a corpus of hand-parsed text. For this purpose, we have Penn treebank (Marcus 1993). Penn Treebank is extensively used since it is the largest annotated dataset for English. In recent years Penn Treebank has been immensely used and considered as a standard for training and testing statistical parsers. Parseval measures are used to evaluate the Penn Treebank parsers. From many Parseval measures most commonly used ones are and labelled recall (LR) and labelled precision (LP). Sometimes Bracketed precision (BP) and bracketed recall (BR) are also used which are less strict measures then LP and LR.

1.1.2. Dependency Parsing

Identifying a sentence and allocating a syntactic arrangement to it is the major task of dependency parsing. In the dependency-based method, the head-dependent relation provides an estimate to the semantic relationship between arguments and their predicates.
The translation of a sentence to its dependency structure is done in two subtasks:

- Classify the structure for all head-dependent relationships.
- Classify these relations with their correct dependency relations.
-

A tree of dependency parsing is a coordinated diagram (a directed graph) which fulfills the stated following limitations:

- It consists of a single assigned root hub that does not have any approaching segments or arcs.
- With the exemption of the root hub, every vertex has precisely one approaching segment or arc.
- From every vertex in V, a unique way exits from the root hub.

In short, the stated requirements guarantee that every word has a distinct head, to which the dependency tree is linked, and a unique root hub by which one can pursue a unique guided path to each word of the sentence.

The idea of projectivity forces an extra restriction and is firmly identified with the setting free nature of human dialects. If there is a way from the head to each word that lies between the head and it's dependent, then an arc from a head to its dependent is considered as projective. A dependence tree is therefore said to be projective if every single one of the arcs is projective There are, in any case, numerous impeccably substantial developments especially in dialects with a generally adaptable word that leads to non-projective trees.

Presently the assignment of Syntactic parsing is very unpredictable because of the way that a given sentence can have numerous parse trees which we call as ambiguities. Consider a sentence "Book that flight." which can frame various parse trees dependent on its uncertain grammatical speech tags except if these ambiguities are settled. Picking a right parse from the numerous conceivable parses is called as syntactic disambiguation.
The two main methods of dependency parsing are:

a) Transition-based dependency parsing
b) MST (Maximum spanning tree) dependency parsing

Transition based parsers commonly have a linear or quadratic complexity. MST based parsers divides the dependency structure into small parts called 'factors'. The components of the principle MST parsing algorithm are edges that consists of the head, the edge name and the dependent (child). This algorithm has quadratic complexity. (Bernd Bohnet., 2010).
Treebanks have a critical job in the advancement and assessment of dependency parsers. Having human annotators legitimately create dependency structures for a given corpus. The most generally utilized syntactic structure is the parse tree which can be produced utilizing some parsing algorithms. These parse trees are valuable in different applications like sentence grammar checking, co-reference goals, question, and their answers, data extraction or all the more significantly it assumes a basic job in the semantic analysis stage. We can likewise utilize a deterministic procedure to decipher existing principal based treebanks into dependency trees using head rules. The significant English reliance treebanks have to a great extent been removed from existing assets, for example, the Wall Street Journal segments of the Penn Treebank (Marcus et al., 1993). The later OntoNotes venture (Hovy et al. 2006, Weischedel et al. 2011) expands this methodology going past customary news content to incorporate conversational phone speech, newsgroups, weblogs and talk programs in English, Arabic and Chinese.

2. Literature Review

Following is the previous work for statistical parsing and dependency parsing.

2.1. Statistical Parsing

Magerman [9] presented a statistical parser called SPATTER. It achieves the best accuracy by building a complete parse for every sentence in the corpus. SPATTER is based on a decision-tree learning technique. Using the PARSEVAL evaluation measure, SPATTER on the Penn Treebank Wall Street Journal corpus achieves 86% precision P (see equation 1) and recall R (see equation 2), and 1.3 crossing brackets CB (see equation 3) per sentence for sentences with a word length of 40 or less. For sentences having word length between 10 and 20, SPATTER achieves 91% P, 90% R, and 0.5 CB.

P = number of Correct Components / number of Components in parser output (1)
R = number of correct Components / number of Components in gold standard (2)
CB = number of Components in parser output that cross gold standard Components / number of Components in parser output (3)

In 1996 Collins [10] presents his first model for statistical parsing. Below equation 4 demonstrates a conditional model capable of parse selection, where for a given sentence S in the corpus, the probability of a parse tree T is calculated directly. Input to the model is a Part of Speech (POS) tagged sentence which produces a tree as an output. For a given sentence S in the corpus and its tree T, the conditional model for Collins represents the probability in the following manner:
The most likely parse under the model is then:

T (best) = argmax T (P (T |S)) (4)

Collins (1996) model showed an improvement in Precision and Recall when compared with Magerman's (1995) results.

Collins presented a new model in 1997 [11] which improved on the previous results of Collins conditional model (1996). This approach is based on a generative model. The Collins (1997) accounts for word-word dependencies when generating a parse tree. This model focuses on the modeling of the parses and deals with the flat trees of the Penn Treebank corpus. This generative version of Collins parser corpus shows an improvement of 2.3% on the conditional model of Collins (1996). It achieves 88.1% P and 87.5% R on Wall Street Journal.

In [12] Collins (1999) few problems were observed with the generative model for Collins (1997). It was noticed that Collins model is no longer a model for predicting maximum likelihood because of how the dependency probabilities were estimated. Another deficiency is its way of considering all dependency relations as independent. Due to these reasons, Collins presented another modification to his previous model.

Charniak [13] (2000, 1999) after presenting his first model for statistical parsing in 1997 Charniak presented another statistical Treebank parser. It outperformed the Collins generative model presented in 1997. Charniak's main advantage to Collin's is generating candidate parses using a simple probabilistic chart parser. Charniak's model showed an improvement of 0.45% in labelled recall (LR) and labelled precision (LP). The model achieved average Precision and an average recall of 91.1% on sentences with length less than 40. For sentences with length less than 100, the model achieved 89.5% average precision and recall on Penn Treebank corpus. Over the previously best results, Charniak's model achieves an error reduction of 13% for single parser on this test set.

Henderson and Brill [14] presented an approach where they combined the results/prediction of three current existing parsers. They combined Charniak's 1997 model with Collins 1997 and Ratnaparkhi 1998 model to better understand the capabilities of parsers and to check if they yield better results. This combination of three parsers gave the best results with Labelled precision of 92.1% and Labelled Recall of 89.2% on the development set while achieving LP of 92.4% and LR of 90.1% on the test set of Penn Treebank. These are best-known results up till now.

Parser presented by Bod [15] claims to give a better performance in terms of Parseval measures. It improved on the Charniak's result by achieving 89.7% LP and LR on sentences with 100 words. For sentences with 40 words or less Bod's model achieves 90.8% LP and 90.6% LR. Although it is debatable whether an increase of 0.2% in LP and 0.1% in LR is considered an improvement. Bods' model takes arbitrary structural and lexical dependencies into consideration when computing probabilities of a parse tree as it is based on Data-Oriented Parsing (DOP).

Collins in 2000 and Collins and Duffy in 2002 [16] presented two approaches in which they improved the performance for Collins 2000 model by re-ranking the parses using a different model on the outcome of Collins' 1999 model. LP improved from 88.1% to 88.3% while LR improved from 88.3% to 89.6% on Collins 1999 model. For Collins 2000 model using a variant for boosting LP improved to 88.3% and LR to 89.6%. For Collins and Duffy 2002 model by using a DOP like approach using a voted Perceptron LP improved to 88.6% and LR to 88.9 %.

2.2. Dependency Parsing

A huge interest has been seen in Dependency Parsing lately for applications such as relation extraction, machine translation, synonym generation, and lexical resource augmentation. The main reason for using dependency structures is because they are highly effective to study and parse while still training much of the predicate-argument information needed in a lot of applications.

Most of these parsing models have concentrated on trees that are projective, including the effort of Eisner (1996), Yamada and Matsumoto (2003), Collins et al. (1999), Nivre and Scholz (2004), and McDonald et al. (2005).
A parsing model presented by Nivre and Nilsson in 2005 allows to include edges that are non-projective, into trees using learned edge transformations in the memory-based parser. The method varies in examining efficiently the full span of non-projective trees. The main focus was that the dependency parsing can serve as the main search point for an MST in a directed graph. This specifies the regular projective models of parsing that are based on the Eisner algorithm (Eisner, 1996) to have a better efficient of O (n 2). By using the spanning-tree illustration, to cover non-projective dependencies we extend the work of McDonald et al. (2005) on online large-margin discriminative training methods (McDonald, Pereira, and Ribarov; 2005)

Nowadays there has been an increase in the usage of dependency representations through many tasks of natural language processing (NLP). Stanford dependency is extensively used in both NLP and biomedical text mining.

Stanford Dependency was originally extracted from constituent parses but the production of parse trees from the raw text was quite time. The approaches hence designed specifically for dependency parsing such as Covington, minimum spanning tree (MST), Eisner and Nivre should perform faster, assuming that they have low time complexity. The different approaches are compared in terms of their collective accuracy and speed and characteristic errors are reported. The parsing models are trained using the training set extracted from the Penn Treebank that consists of sections 2 through 21, different parsers of dependency were compared such as MaltParser package v1.3 selected models, the MSTParser 0.4.3b, and the rule Based RelEx parser 1.2.0. We used F1-score other than accuracy because the typical Stanford dependency representation parsers can generate a variety of different dependencies for every sentence. The fastest parsers were the Malt package, Nivre, and Covington. Nivre Eager and MSTParser (Eisner) and they achieved better F1scores when the interaction between model and features was not used. If parsing a huge amount of data, and speed is important, the experiments suggest that the top choice is to use parsers included in the Malt package. (Cer, D. M., De Marneffe, (2010, May)).

3. Performance Evaluation

Many different parsers have been presented in the above section having unique abilities and different performances each trying to perform well than the previous. The main question is what type of parsers should be used in which context. The decision to apply these models depends on the type of the problem under study. To facilitate the decision making a performance overview of all the parsers is given below.

3.1. Statistical Parsing

Table I shows the performance of multiple Statistical Parsers over the Parseval Measures.

• In [9] it was shown that previous syntactic natural language parsers used were not capable of handling ambiguous large-vocabulary text. They had poor performances on standard datasets like Wall Street Journal of Penn Treebank which led to the new approach presented by Magerman called SPATTER (Statistical Pattern Recognizer). It is based on decision-tree learning technique and has accuracy way better than any parser published up till 1995. SPATTER requires very less linguistic knowledge and is compared against state of the art grammar-based parsers. Decision trees provide a ranking system by assigning a probability distribution to the possible choices, which not only specifies the order of preference but also gives a measure of the relative likelihood that each choice is the one which should be selected. The limitation of the searching strategy of SPATTER is its possible consumption of available memory before completing the search. But conveniently this memory exhaustion occurs on sentences which SPATTER most likely will get wrong anyway. So little or no performance loss is observed due to this search errors.

• The study in [10] reveals that Collins and Magerman both use lexicalized PCFG which is associating a head word to every non-terminal present in the parse tree. Collins parser performs almost equally as SPATTER when it is trained and tested on Wall Street Journal portion of Penn Treebank. The advantage of Collins 1996 model over SPATTER is the simplicity of its architecture and working. Still, many improvements can be made by using a more sophisticated probability estimation techniques like deleted interpolation or estimation on relaxing the distance measure for smoothing could be used. Another limitation of Collins 1996 model is that it does not account for valency when calculating the parse.

• Collins [11] attempt to address the flaws of the model presented in 1996 by putting forth 3 models. It shows that sub-categorization and wh-movement can be given a probabilistic treatment thus resulting in the statistical interpretation of the concepts causing an increase in performance by adding useful information to the parser's output. The average improvement of Collins 97 over the previous model is 2.3%. Model 1 presented has clear advantages when handling unary rules and distance measures. Model 2 and 3 can apply condition on any structure that has been previously generated while Collins 96 lack in this treatment.

• [12] Is an update of previous works of Collins. It addresses the limitation of Collins 1996 and 1997 related to punctuation as surface features of the sentence. Previous models failed to generate punctuation and are considered a deficiency of the model. Collins 2000 uses a technique that is based on boosting algorithms for machine learning for re-ranking the best outputs using additional features.

• The major invention of Charniak's [13] 2000 model is the use of maximum entropy inspired model which results in an increase of 2% in performance due to its strategy of smoothing and to combine multiple conditioning events for testing. Maximum entropy inspired approached has certain advantages over the probabilistic model and has recommended itself for use due to its novel approach of smoothing. Most important progress accomplished by using Charniak's model over conventional deleted interpolation is the flexibility achieved due to simpler maximum-inspired-model which let us experiment with different conditioning events and to move up to Markov grammar without significant programming. This model uses Markov processes to generate rules. The additional features incorporated boost the performance. The main goal for Charniak's parser is to generate model flexible enough to allow changes for parsing to more semantic levels.

• To solve some of the fundamental problems of Natural Language processing like parsing some authors including Henderson and Brill [14] may adopt a unique approach to combine the previous parsers to obtain better results. Collins along, with Charniak and Ratnaparkhi model, are experimented to explore different parser combination. With poor parser being introduced during the experiments, Techniques like parser switching and parser hybridization still gave better results. For more powerful parser combinations the results can be improved further.

• Bod 2001 [15] presents results that are comparable to the results of previous models like Charniak and Collins 2000. The main goal of the Bod model is to achieve maximal parse accuracy by applying constraints of several words in a fragment and to the depth of lexicalized fragments. Many previous models applied constituent lexicalization on Wall Street portion of Penn Treebank while Bods 2001 DOP based model uses frontier lexicalized approach. The results obtained from Bods models claim that using frontier lexicalization yields better results and is a better alternative to constituent lexicalization. Another difference of Bod with other models is its use of treebank grammar as an underlying grammar of its DOP model. Another future area could be the application of Markov grammar on the DOP model which will further improve the results.

• The main advantage of this model presented by Collin and Duffy [16] is the application of the perceptron algorithm on exponentially big representations of parse trees. It is computationally efficient and leads to a polynomial-time 2 algorithm for training and testing phases of the perceptron. It can stretch to more complex domains. Due to its different parameter estimation when compared to Bod and other models the computation is manageable.

TABLE I: Performance of all statistical parsers on penn treebank corpus

Parsers	**Evaluation Measures**		
	LP	***LR***	***CB***
Magerman 1995 [9]	86%	86%	1..3
Collins 1996 [10]	86.3%	85.8%	1.14
Collins 1997 [11]	88.6%	88.1%	0.96
Henderson and Brill [14]	92.4%	92.1%	-
Collin 2000 [12]	90.4%	90.1%	0.73
Charniak 2000 [13]	90.1%	90.1%	0.74
Bod 2000 [15]	90.8%	90.6%	-
Colins and Duffy 2001 [16]	88.6%	88.9%	-

3.2. Dependency Parsing

3.2.1. Deterministic Dependency Parsing

Collins and Charniak are one of the best accessible parsers prepared on the Penn Treebank, utilize statistical models for disambiguation that utilize dependency relations. The Yamada and Matsumoto method of deterministic dependency parser and that of Collins and Charniak, when prepared On the Penn Treebank, gives a nearly equal accuracy. The parser depicted in this paper is like that of Yamada and Matsumoto in that it utilizes an algorithm of deterministic parsing in blend with a classifier actuated from a treebank. Be that as it may, there are likewise significant differences between the two methodologies. Most importantly, while Yamada and Matsumoto utilizes a severe algorithm of bottom-up(basically shift-reduce parsing), the present parser utilizes Nivre's algorithm, which uses bottom-up and top-down approaches together to increase the accuracy. The experiment was carried out on two sets whose result is shown in table II and IIA.

- Set G which contained grammatical roles from Penn
- Set B contained the function tags for grammatical roles with normal bracket labels (S, NP, VP, etc.).

And the evaluation metrics used are:

- Unlabeled attachment score is the measure of words that are root and are correctly identified as head.
- Labelled attachment score is the measure of words that are root and are correctly identified as head and their dependency type.
- Dependency accuracy is the measure of words that are non-root and are correctly identified as heads.
- Root accuracy is the measure of root words correctly identified as roots.
- Complete match is the measure of sentences whose unlabeled dependency structure is correctly identified.

TABLE II: PERFORMANCE OF DETERMINISTIC DEPENDENCY PARSERS ON PENN TREEBANK CORPUS (6. NIVRE AND M.SCHOLZ 2004)

Parsing Models	**Evaluation metrics**		
	Dependency Accuracy	***Root Accuracy***	***Complete Match***
Charniak	92.10%	95.20%	45.20%
Collins	91.50%	95.30%	43.30%
Yamada and Matsumoto	90.30%	91.60%	38.40%
Nivre and Scholz	87.30%	84.30%	30.40%

TABLE IIA: PERFORMANCE OF DETERMINISTIC DEPENDENCY PARSERS ON PENN TREEBANK CORPUS (6. NIVRE AND M.SCHOLZ 2004)

Evaluation metrics	**Data sets**		
	Grammatical Roles from Penn II (Experiment # 1)	***Function Tags for Grammatical Roles (Experiment #2)***	***Combination of both Experiments***
Unlabeled Attachment Score	85.8%	87.1%	-
Labelled Attachment Score	84.6%	84.4%	86.0%

3.2.2. Constituent-to-Dependency Parsing

PENN2MALT disposes of the deep information in the dependency tree. In the new strategy, the topicalized phrases and words are connected to their respective semantic head. Other than this the new approach used a richer collection of arced labels than that used in PENN2MALT. MALTPARSER depends on a parsing system that constructs a parse tree gradually while continuing through the sentence one token at any given moment. By utilizing this type of system, a rich history-based list of capabilities for the SVM classifier is made, that can be used for choosing activities. MSTPARSER predicts a parse tree by expanding a function of scoring over the space of all parse trees. The scoring function is a weighted sum of single connections or links. Table III describes MALT Parsers and MST Parser for different parsing sets.

TABLE III: PERFORMANCE OF MALT AND MST DEPENDENCY PARSERS ON PENN TREEBANK CORPUS (17.JOHANSSON,R., & NUGUES,P.(2007))

Parsing sets	**Parsing models**			
	MALT PARSER	***MALT PARSER***	***MST PARSER***	***MST PARSER***
	LABELED	***UN LABELED***	***LABELED***	***UNLABELED***
PENN2 MALT	90.30%	91.36%	92.04%	93.06%
NEW CONVE-RSION	87.63%	90.54%	86.92%	91.64%

3.2.3. MIRA (18. McDonald, R., Crammer, K., & Pereira, F. (2005))

It is an online learning algorithms which intuitive and easy to comprehend and implement. To form dependency structures the extraction rules of Yamada and Matsumoto were used. For the evaluation and development of sets, the tagging system of Ratnaparkhi was used and POS tags were assumed as the input for the system.

For large margin multi-class classification, Crammer and Singer established an approach (equation #5) which was then extended to structured classification by Taskar.

EQUATION # 5 (18. MCDONALD, R., CRAMMER, K., & PEREIRA, F. (2005))

$$\begin{aligned}&\min \|\mathbf{w}\|\\ &\text{s.t.}\quad s(\boldsymbol{x},\boldsymbol{y}) - s(\boldsymbol{x},\boldsymbol{y}') \geq L(\boldsymbol{y},\boldsymbol{y}')\\ &\forall(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{T},\ \boldsymbol{y}' \in \mathrm{dt}(\boldsymbol{x})\end{aligned}$$

The above-mentioned equation #1 optimization is directly mapped into the online framework by the margin infused relaxed algorithm (MIRA). On every attempt, while applying the change to the parameter vector MIRA tries to maintain the standard and keep it as small as possible, to classify an instance correctly the margin should be at least equal to the loss of classifying incorrectly. This can be done by substituting the following changes in the original algorithm (equation #5) present online and form another (equation #6).

EQUATION # 6 (18. MCDONALD, R., CRAMMER, K., & PEREIRA, F. (2005))

$$\begin{aligned}&\min \left\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\right\|\\ &\text{s.t. } s(\boldsymbol{x}_t,\boldsymbol{y}_t) - s(\boldsymbol{x}_t,\boldsymbol{y}') \geq L(\boldsymbol{y}_t,\boldsymbol{y}')\\ &\forall\boldsymbol{y}' \in \mathrm{dt}(\boldsymbol{x}_t)\end{aligned}$$

To apply dependency parsing using MIRA, we can just consider the parsing as a multi-class classification problem in which, every dependency tree is considered as one of the classes of the sentence. Nevertheless, this clarification fails in reality as a normal sentence has a lot of possible dependency trees thus making it exponentially complex. To overcome this issue another equation #7 was created.

EQUATION # 7 (18. MCDONALD, R., CRAMMER, K., & PEREIRA, F. (2005))

$$\begin{aligned}&\min \left\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\right\|\\ &\text{s.t.}\quad s(\boldsymbol{x}_t,\boldsymbol{y}_t) - s(\boldsymbol{x}_t,\boldsymbol{y}') \geq L(\boldsymbol{y}_t,\boldsymbol{y}')\\ &\forall\boldsymbol{y}' \in \mathrm{best}_k(\boldsymbol{x}_t;\mathbf{w}^{(i)})\end{aligned}$$

TABLE IV: PERFORMANCE OF MIRA DEPENDENCY PARSERS ON PENN TREEBANK CORPUS

Parsing Models	**Evaluation measures**		
	Accuracy	***Root***	***Complete***
Y&M	90.30%	91.60%	38.40%
N&S	87.30%	84.30%	30.40%
AVERAGE PERCEPTI ON	90.60%	94.0%	36.50%
MIRA	90.90%	94.20%	37.50%

Accuracy: number of words whose parents are correctly identified.

Root: number of trees in which the root is identified correctly.

Complete: number of sentences whose dependency was correctly identified.

4. Conclusion

Comparison between parsers leads us to examine the similarities and differences between multiple models and the scenarios in which they tend to perform better. The famous CKY parsing algorithm can represent the ambiguities that occur while parsing efficiently but it is not able to resolve them. So statistical and dependency models are designed to overcome the Limitations of the previous ones thus showing an increase in the overall measures of evaluation.

5. Acknowledgment

We would like to give our deepest appreciation to all the people who helped us complete this paper. A special gratitude to our instructor and co-author of this paper [Dr. WAQAS ANWAR], whose support helped us in achieving the goals of writing this paper.

6. References

[1] Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics, 19(2):313–330.

[2] Snow, D. Jurafsky, and A. Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In NIPS 2004.

[3] Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In Proc. COLING.

[4] Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In Proc. IWPT.

[5] Nivre and M. Scholz. 2004. Deterministic dependency parsing of English text. In Proc. COLING

[6] McDonald, Pereira, Ribarov. 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms.

[7] McDonald, R., & Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

[8] Magerman, D. M. (1995, June). Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics* (pp. 276-283). Association for Computational Linguistics.

[9] Collins, M. J. (1996, June). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics* (pp. 184-191). Association for Computational Linguistics

[10] Collins, M. (1997). Three generative, lexicalized models for statistical parsing. ArXiv preprint cmp-lg/9706022.

[11] Collins, M. (2003). Head-driven statistical models for natural language parsing. Computational linguistics, 29(4), 589-637

[12] Charniak, E. (2000, April). A maximum-entropy-inspired parser. In Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference (pp. 132-139). Association for Computational Linguistics.

[13] Henderson, J. C., & Brill, E. (2000). Exploiting diversity in natural language processing: Combining parsers. ArXiv preprint cs/0006003

[14] Bod, R. (2001, July). What is the minimal set of fragments that achieves maximal parse accuracy? In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (pp. 66-73). Association for Computational Linguistics.

[15] Collins, M., & Duffy, N. (2002, July). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In Proceedings of the 40th annual meeting on association for computational linguistics (pp. 263-270). Association for Computational Linguistics

[16] McDonald, R., Crammer, K., & Pereira, F. (2005). Online large-margin training of dependency parsers. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05) (pp. 91-98).

[17] Johansson, R., & Nugues, P. (2007). Extended constituent-to-dependency conversion for English. In Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007) (pp. 105-112)

Corpus of Aspect-based Sentiment for Urdu Political Data

Ehsan ul Haq[1],Sahar Rauf[2],Sarmad Hussain[3],Kashif Javed[3]
Center for Language Engineering
Al-Khawarizmi Institute of Computer Science
University of Engineering and Technology, Lahore
[1]*{ehsan.ulhaq},*[2,3]*{firstname.lastname}@kics.edu.pk*

Abstract

We present a corpus of Urdu political data annotated at aspect and sentiment level. The corpus contains 8760 tweets regarding four different aspects (Members, Projects, Party and Actions) of three political parties (PTI, PMLN and PPP) of Pakistan. We also present the results of a baseline system developed using the corpus for analyzing its reliability. It can be seen that the classifiers have achieved reasonable scores for aspects categorization and sentiment classification tasks.

1. Introduction

Sentiment analysis is defined as a task of automatically identifying opinions expressed in a text [1]. The text to be analyzed can be a feedback regarding a product or service, a political review or a social media comment [2]. The sentiment analysis can be done at document level, sentence level and also at aspect level [3]. In case of document level sentiment analysis, the task is to assign an overall sentiment to the document. On the other hand, sentence level sentiment analysis assigns a polarity value to each individual sentence of the document. The aspect level sentiment analysis provides an in-depth analysis by assigning sentiments to different aspects of entities mentioned in a text.

There are different approaches that have been used for performing sentiment analysis. These approaches include lexicon based approach; machine learning based and hybrid techniques [4]. In lexicon based approach, a sentiment lexicon is used for assigning sentiment to text by aggregating the sentiment scores of words present in the text. In machine learning based approaches, a sentiment tagged corpus is used to train machine learning models using supervised learning approach. After training models, they are used for assigning sentiments to input text. Another commonly used approach is using a hybrid technique. In hybrid techniques, a combination of machine learning models and lexicon is used for performing sentiment analysis.

Nowadays sentiment analysis has become an important task in the area like business intelligence (BI) in which a company wants to know the sentiment of customers towards their products or services. They use this analysis in decision making and overcoming their weaknesses. Another important area in which sentiment analysis is used is called social media monitoring (SMM) in which social posts are analyzed for finding opinions towards different entities [5]. Sentiment analysis is also used in disease surveillance systems for monitoring social media content mentioning symptoms, prevention and fear regarding a disease in different areas [6]. Another important area in which sentiment analysis is used is analysis of political data [7].

In this work, we are presenting a corpus for aspect based sentiment analysis (ABSA) task for Urdu political data which can be used as a gold-standard for automatic aspect-based sentiment annotation.

The rest of paper is organized as follows. Section 2 contains related work. Section 3 explains the methodology used for corpus development and its statistics. Section 4 contains baseline results and Section 5 is based on conclusion and future work.

2. Related Work

A corpus for ABSA in political debates has been presented in [7]. The corpus consists of transcribed speeches from the two presidential debates of the 2016 US election. The authors have annotated the corpus and provided baseline results for aspect based sentiment analysis using Support Vector Machine (SVM) algorithm.

The authors in [8] have presented an Italian corpus for aspect based sentiment analysis of movie reviews. The corpus contains sentences that have been manually annotated according to various aspects of movies and also polarities expressed toward them.

Two French language datasets for the purpose of development and testing of aspect based sentiment systems have been presented in [9]. The first dataset consists of 457 restaurant reviews (2365 sentences).The second contains 162 museum reviews (655 sentences). Both datasets were developed as part of SemEval-2016 Task 5 "Aspect-Based Sentiment Analysis" where seven different languages were represented, and are publicly available for research purposes.

A Turkish sentiment corpus comprised of user reviews annotated using semi-automatically is constructed in [10]. The corpus contains Turkish hotel reviews dataset which has 1000 reviews and 5364 sentences. The corpus also contains root forms of words, their usage, POS tags and sentiments.

An Arabic Laptops Reviews (ALR) dataset [11] for ABSA has been prepared according to the annotation scheme of SemEval16-Task5. The annotation scheme addresses two problems: prediction of aspect category and sentiment polarity label prediction. An evaluation procedure that extracts n-grams' features and uses a Support Vector Machine (SVM) classifier has also been described. in order to allow researchers to gauge and compare the performance. The results of evaluation show that there is a need for improvements in the performance of the SVM classifier for the aspect category prediction problem. On the other hand, the SVM's accuracy is actually high for sentiment polarity label prediction.

The work in [12] provides a human annotated Arabic dataset (HAAD). HAAD comprises of books reviews in Arabic which have been annotated by humans with aspect terms and their polarities. The paper also reports a baseline results with common evaluation techniques for the purpose of future evaluation of ABSA systems.

3. Urdu ABSA Corpus for Political Domain

This section describes the methodology that has been used for developing the corpus.

3.1. Corpus Collection and Preprocessing

We have collected 8760 tweets containing user comments regarding the following three political parties of Pakistan: Pakistan Tehreek-e-Insaaf (PTI), Pakistan Muslim League Nawaz (PMLN) and Pakistan People Party (PPP). The tweets have been collected in Urdu language using tweeter API. For searching relevant tweets, we have used the keywords indentified for each aspect in search queries.

After collecting the data, the next step is preprocessing. In preprocessing step, we have done the following tasks:

- Removal of special characters like hash tags, emoticons and punctuation marks like; '?, !, ;'.
- Resolving segmentation issues like incomplete words, issues of extra spaces between letters of words and presence of Zero Width Non Joiner (ZWNJ).
- Removal of duplicate tweets
- Removal of URLs and hyper links from the tweets.

3.2. Aspect-based tagging

The corpus has been designed for conducting aspect based sentiment analysis of the reviews of people regarding the above mentioned political parties of Pakistan. The reviews are analyzed for finding sentiments regarding the following four aspects:

3.2.1 Member

This aspect refers to the positive, negative and neutral comments of users regarding members of a party. For example, consider the following sentence:

/عمران خان کی نیت صاف ہے/
/ɪmrɑ:n xɑ:n ki: ni:jjət̪ sɑ:f hæ:/

Imran Khan's intentions are pure

Here, the quality of a PTI member has been mentioned in a positive way as the word 'pure' is very positive.

Consider another example:

/عمران خان کی سونامی تباہی ہے/
/ɪmrɑ:n xɑ:n ki: so:nɑ:mi: t̪əbɑ:hi: hæ:/

Imran Khan's tsunami is disastrous

Here, a negative sentiment has been expressed in the form of words 'tsunami' and 'disastrous'.

3.2.2. Projects

This aspect refers to the projects of a party. The followings could be the examples of party's projects; /نیا پاکستان/ /nəjɑ: pɑ:kɪst̪ɑ:n/ /New Pakistan/, / اورنج ٹرین/ɔ:rɪndʒ tre:n/ /Orange train/ etc.

Consider the following review as an example:

/اورنج ٹرین ایک ناکامیاب پروجیکٹ ہے/
/ɔ:rɪndʒ tre:n e:k nɑ:kɑ:mjɑ:b pro:dʒækt hæ:/
Orange train is an unsuccessful project

In this sentence, a negative sentiment is attached with the project of PMLN in the form of word 'unsuccessful'.

3.2.3. Actions

This aspect refers to policies of a party. These policies and actions could be foreign policies, economy policies, price control policies and health policies etc. The following keywords could be examples of action aspect; /مہنگائی/ /mæhŋgɑ:i:/ /Inflation/ and /قرضہ//qərzɑ:/ /Debt/ etc.

Consider the following example:

/ پی ٹی آئی کی حکومت کی وجہ سے مہنگائی بڑھ رہی ہے/

/pi: ti: ɑ:i: ki: həku:mət̪ ki: vəʤɑ: se: mæhŋgɑ:i: bəɽʰ rəhi: hæ:/

Inflation is increasing due to PTI's government

In the above mentioned sentence, a negative sentiment is expressed towards the action of PTI.

3.2.4. Party

This aspect refers to feedback of people regarding party as a whole.

Consider the following example:

/پی ٹی آئی کی کارکردگی اچھی نہیں/

/pi: ti: ɑ:i: ki: kɑ:rkərd̪əgi: əʧʰʧʰi: nəhi:/

The performance of PTI is not good

This review contains a negative feedback regarding performance of PTI as a whole party rather than individual members.

3.3. Sentiment Polarities

For the purpose of assigning sentiments to each aspect, we have used a five point scale from -2 to 2, where -2 means more negative, -1 means less negative, 0 means neutral, 1 means positive and +2 means more positive.

3.4. Corpus Tagging

A team of expert linguists with Mphil and PhD degrees in the area of Applied Linguistics and Urdu Literature respectively has tagged the corpus. The tested data achieved Inter Annotator Accuracy (IAA) of 75% at aspects and sentiment level tagging.

3.5. Corpus Statistics

This section explains the statistics of corpus that has been tagged as aspect and sentiment level. Table 2 below is presenting the statistics of each aspect in the developed corpus.

Table 2. Statistics of aspects in developed corpus

PARTY	MEMBER	PROJECT	PARTY	ACTION
PTI	1622	445	798	669
PMLN	2215	758	621	308
PPP	1497	306	919	535

The statistics of aspect wise sentiments is given in Table 3 below.

Table 3. Statistics of aspect wise sentiments in the corpus

		Sentiment		
Party	**Aspects**	**POS**	**NEG**	**NEU**
PTI	MEMBER	314	1090	218
	PROJECT	50	391	4
	PARTY	151	521	126
	ACTION	81	570	18
PMLN	MEMBER	758	1257	200
	PROJECT	406	205	147
	PARTY	135	375	111
	ACTION	157	140	11
PPP	MEMBER	295	914	288
	PROJECT	89	126	91
	PARTY	295	501	123
	ACTION	34	489	12

4. Automatic Aspect-Based Sentiment Annotation

The developed corpus has been used for measuring the performance of machine learning based algorithms and baseline results have been reported. This evaluation is useful for indicating the reliability of the developed corpus.

Hence, for the purpose of analyzing that whether the corpus can be used for developing an ABSA classifier, we have trained classifiers for aspects recognition and sentiment tagging using an SVM library, namely LIBSVM in Weka [13].

We have used One-Vs-All approach and trained classifiers for each aspect and sentiment separately. So, we have trained 12 models for aspects and 12 models for sentiments classification. We have evaluated the results by performing 10-fold cross validation.

We have also experimented with different n-gram features with n ∈ {1,2,3}. For the purpose of vectorizing the data, we have used a binary scheme in which the vector contains 1 if word is present and 0 otherwise.

The results for sentiments classification system are presented in Table 3 below.

Table 4. F1-scores for sentiment models using CV

		Parties		
Features	**Aspects**	**PTI**	**PMLN**	**PPP**
Unigram	MEMBER	70.7	70.6	65.2
	PROJECT	85.7	62.8	60.7
	PARTY	77	65.3	77.5
	ACTION	80	75.1	89.9
Bigram	MEMBER	64	70.1	65.7
	PROJECT	84.1	62.8	60.9
	PARTY	90.1	64.2	71.9
	ACTION	80.3	73.6	90.2
Trigram	MEMBER	63.1	63.9	61.1
	PROJECT	82.6	47.6	59.9
	PARTY	61.1	53.3	60.4
	ACTION	78.9	59.8	89.9

The results of aspects classifiers are presented in Table 4 below.

Table 5. F1-scores for aspects models using CV

		Parties		
Features	**Aspects**	**PTI**	**PMLN**	**PPP**
Unigram	MEMBER	88.5	86.3	90
	PROJECT	96.2	94.6	98.3
	PARTY	90.7	92.9	91.6
	ACTION	91.2	94.1	95.6
Bigram	MEMBER	89.9	87.7	88.4
	PROJECT	96.8	95.6	98.9
	PARTY	92.9	92.7	93.7
	ACTION	90.6	94.6	94.1
Trigram	MEMBER	79.1	76.1	77.1
	PROJECT	94.3	91.3	92.3
	PARTY	90.7	90.6	89.9
	ACTION	88.9	94.7	85

5. Conclusion

A corpus for ABSA for Urdu political data has been presented in this paper. The developed corpus has been tagged at aspect and sentiment level with IAA score of 75%. A baseline system has also been developed using the corpus for analyzing its reliability in future use. It can be seen from the above mentioned results that the aspect classifiers have achieved a F1-score of more than 90% in most of the cases. Moreover, the sentiment classifiers have also achieved a F1-score of more than 70% in many cases

6. References

[1] Walaa, A. Hassan, and H. Korashy Medhat, "Sentiment analysis algorithms and applications: A survey," Ain Shams engineering journal, vol. 5, no. 4, pp. 1093-1113, 2014.

[2] Maria, D. Galanis, H. Papageorgiou, I. Androutsopoulos and S. Manandhar Pontiki, "Semeval-2016 task 5: Aspect based sentiment analysis," in Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016), 2016, pp. 19-30.

[3] D. Mohey and El-Din Mohamed Hussein, "A survey on sentiment analysis challenges," Journal of King Saud University-Engineering Sciences, vol. 30, no. 4, pp. 330-338, 2018.

[4] "Automated sentiment analysis in tourism: Comparison of approaches," Journal of Travel Research, vol. 57, no. 8, pp. 1012-1025.

[5] Muhammad, M.Diab, and S.Kübler Abdul-Mageed, "SAMAR: Subjectivity and sentiment analysis for Arabic social media," in Computer Speech & Language, 2014, pp. 20-37.

[6] V. Kumar, and S.Kumar. Jain, "Effective surveillance and predictive mapping of mosquito-borne diseases using social media.," Journal of Computational Science, pp. 406-415, 2018.

[7] M.Bexte and T.Zesch D.Gold, "Corpus of Aspect-based Sentiment in Political Debates," in Proceedings of the 14th Conference on Natural Language Processing (KONVENS 2018), 2018.

[8] Antonio, et al. Sorgente, "An italian corpus for aspect based sentiment analysis of movie reviews," in CLICIT2014, 2014.

[9] Marianna, X.Tannier, and C.Richart Apidianaki, "Datasets for aspect-based sentiment analysis in french," in Proceedings of the Tenth International Conference on Language Resources and Evaluation, 2016.

[10] S. İlhan, E. Ekinci, and H. Türkmen Omurca, "An annotated corpus for Turkish sentiment analysis at sentence level.," in In 2017 International Artificial Intelligence and Data Processing Symposium (IDAP).IEEE., 2017, pp. 1-5.

[11] Al-Ayyoub and Mahmoud, "Aspect-Based Sentiment Analysis of Arabic Laptop.," , 2017.

[12] Mohammad, et al. Al-Smadi, "Human annotated arabic dataset of book reviews for aspect based sentiment analysis," in 2015 3rd International Conference on Future Internet of Things and Cloud. IEEE, 2015., 2015.

[13]https://www.cs.waikato.ac.nz/~ml/weka/index.html.

Development and Automation of Phrase Model for Urdu Speech Corpus

[1]Aneeta Niazi, [1]Saba Urooj, [1]Benazir Mumtaz, [1,2]Tania Habib
[1]Center for Language Engineering (CLE),
Al-Khwarizmi Institute of Computer Science (KICS),
[2]Computer Science and Engineering Department
University of Engineering and Technology (UET),
Lahore, Pakistan.
aneeta.niazi@gmail.com, {first name.last name}@kics.edu.pk

Abstract

A phrase model for the annotation of Break Indices (BI) in Urdu speech corpus has been presented. A detailed acoustic analysis has been carried out to understand the patterns of phrase breaks in 1 hour of recorded Urdu speech. A four level phrase model has been proposed, including BI levels 0, 1, 2 and 4. From the outcomes of this analysis, rules have been formulated for automating the process of BI tagging in Urdu speech corpus. For this purpose, the annotated information of word boundaries, Part Of Speech (POS), intonation, stress and pauses from the Urdu speech corpus has been utilized. As the features indicating the prosodic behavior of the pitch contour, including stress and intonation, have already been accurately tagged in the speech corpus, the results obtained from the automatic BI tagging are quite promising. The automatic BI labeling system has provided coverage of 97.8%, and accuracy of 98.3% for the tagging of unseen data.

Keywords: *Break Index, BI, Phrase model, Urdu speech, Automatic annotation, prosodic modeling*

1. Introduction

In a language, a word or a group of words co-existing as a single conceptual unit is known as a phrase [1]. Human speech contains words clustered together to form phrases. These phrases are separated by pauses, or a change in the speaker's tone.

In written text, punctuation is commonly used to indicate phrase boundaries e.g. comma, full stop etc. However, while speaking, humans insert phrase breaks, even in the absence of punctuation. These breaks usually occur in speech while moving from one word to another, mostly for expressing emotions and intentions [2]. A phrase break is also referred to as Break Index (BI).

For building Text-to-Speech (TTS) systems, an adequately large, well annotated speech corpus is one of the basic requirements. The corpus should contain accurate annotations for prosodic features, such as BI, stress and intonation, to make the TTS sound as human-like as possible.

Several efforts have been made to develop an international standard for annotating prosody in speech. One of the earliest and most popular prosody tagging standards is ToBI (Tones and Break Indices) [3]. In ToBI, the different types of pauses in speech are represented by numbers from 0 to 4. A typical break between two adjacent words is represented by BI level 1. A break in place of a comma is indicated by BI level 3, whereas a distinctive pause in between speech segments is represented by BI level 4. This tagging convention is used by many languages. However, this could not be generalized for all the languages. Researchers developed variations of ToBI to suit the requirements of speech annotation for their particular languages e.g. J-ToBI for Japanese, G-ToBI for German, ToDI for Dutch, B-ToBI for Bengali etc. [4].

The existing Urdu speech corpus [4] has been annotated for all the necessary prosodic features of speech. A 5 level scale (from 0 to 4) has been used for marking BI, based on, duration of breaks, and lengthening of phrases, pitch contour and glottalization. During an acoustic analysis for understanding the tonal patterns of Urdu speech, it has been observed that the structure of Urdu phrases is very different from English phrases in the spoken Urdu sentences. In Urdu, word and phrase boundaries are not marked in accordance with the rules followed by English; i.e. the behavior of Urdu BI is very different from the conventions followed by English. Due to the presence of accentual phrase, Urdu phrase structure resembles with the phrase structure of South Asian languages. Therefore, there is a need to re-design the phrase model, and develop new standards for marking BI for Urdu speech.

The manual labeling of phrase boundaries in speech corpus is a time consuming and laborious activity. Machine learning based systems can be used for making the annotation process efficient, but such systems require large amount of annotated data, which is not readily available for under-resourced languages, such as Urdu.

In this paper, we present a detailed acoustic analysis that has been carried out for developing a phrase model for Urdu speech. An automatic BI labeling system has been proposed to annotate the BI in the Urdu speech corpus. Section 2 presents a survey of the existing work for the topics of phrase modeling and break index annotation. The results and discussions are covered in section 4, whereas the findings of this research are concluded in section 5. In section 6, we propose the future directions which can be pursued to further investigate the process of phrase modeling.

2. Literature Review

For long-form reading, phrase model serves as one of the most important components for improving the naturalness of a TTS system [2]. The phrase model has been constructed by analyzing textual features such as dependency tree features, Part Of Speech (POS) and word embeddings. For improving the prediction of phrase boundaries, these features have been given as input to train Bidirectional Long Short Term Memory (BiLSTM) and Classification And Regression Trees (CART) based systems. Both subjective and objective testing has been carried out to compare the performance of BiLSTM and CART systems. The evaluation results have shown that better performance has been obtained by using word embeddings and BiLSTM.

A language independent BiLSTM-CRF (Conditional Random Fields) model has been proposed for prosodic boundary prediction [5]. The architecture consists of three layers, i.e. word embeddings, BiLSTM and CRF. These three layers learn from task-specific embeddings, past and future features and sentence level information respectively. The system has been evaluated for Mandarin and English speech. The results show that using the proposed model, the intonational phrase prediction has been significantly improved as compared to the traditional BiLSTM method.

BI labels have been automatically annotated for Japanese and English speech, using only the information extracted from the speech signal [6]. The automatic labeling is carried out without using any other prior information, such as transcriptions or word boundaries. For this purpose, spontaneous Japanese speech has been used to train BiLSTMs. The trained system is used to annotate Japanese and English speech, and a cross-lingual comparison is made with the monolingual English labeling system. The evaluation results have shown that the system trained with Japanese speech performed better for the BI labels 1 and 2, while the system trained with English speech performed better for the Break Index label 3. The less frequent labels in the data have not been accurately detected. The proposed cross-lingual model can be applied when sufficient amount of data is not available for training a monolingual break index labeling system.

An analysis is carried out to observe the impact of the size of focus constituents on phrase boundaries in French [7]. The experimental results have shown that an accentual phrase boundary gets converted into an intermediate phrase boundary, if it forms the right edge of a narrow focus constituent. However, an intermediate phrase boundary remains unaffected in the presence of a narrow focus constituent in its surrounding context.

Intonational phrase break prediction models have been developed to automatically predict phrase breaks in American English [8]. Binary classifiers, based on logistic regression from the LLAMA machine learning toolkit are used. 50 hours of recorded speech are used for building the system. The prediction models are data driven, based on features including lemmatized words, POS, punctuation, distance from punctuation, as well as dependency-relation features. An overall prediction accuracy of 84.7% has been obtained.

A model for detecting prosodic boundaries in Russian speech, using syntactic as well as acoustic information, has been presented [9]. It is based on a two level architecture, where the possible phrase boundaries are marked by using syntactic information, with the help of a dependency tree parser in the first step. In the second step, a Random Forest (RF) classifier uses a small set of acoustic features, such as tempo, pitch range and amplitude etc., to mark the actual prosodic boundaries. The duration of pauses has been reported to be the best amongst all acoustic features used for predicting prosodic boundaries.

For Indian languages, the analysis of phrases becomes very difficult if there is no punctuation in the text [10]. In read sentences, the units in between the pauses are considered as phrases for analysis. It has been observed that the length of inter-pausal units follows a Gamma distribution. An analysis of shape and scale parameters of speech has shown that these parameters have dependence on the location of inter-pausal units. This information is utilized to improve the prosody modeling of TTS system for four Indian languages. The results have shown considerable improvement in the naturalness of synthesized speech.

An automatic prosodic transcription system has been reported for Bengali and Odia languages [11]. 3 levels of breaks are annotated, i.e. word breaks are represented as B1, phrase breaks as B2 and sentence breaks as B3. For labeling BI automatically, short term energy (STE) of speech signal is considered. The energy associated with silence is negligible as compared to unvoiced and voiced regions. Also, unvoiced segments have very small duration as compared to silence and voiced segments. The duration thresholds for B1, B2 and B3 have been determined by histograms. From the results, it is observed that the automatic BI tagging system detected many spurious breaks, which were not perceived during manual tagging.

For Urdu speech, a 5 level scale (from 0 to 4) has been presented for annotating break indices [4]. Acoustic features including pitch contour, duration of pauses and glottalization have been considered for analysis. 1036 files from CLE Urdu speech corpus are used; comprising of simple sentences. An automatic BI labeling system is developed to annotate 10 hours of speech. The reported analysis does not include complex predicates and compound sentences.

3. Proposed Methodology

This section includes the details of the data collection, analysis and rules developed to formulate phrase model for Urdu.

3.1. Data Acquisition

From CLE Urdu Speech Corpus, 1403 files have been acquired for carrying out break index analysis. Out of these files, 983 files are used as training data to develop the automatic BI labeling utility. The remaining 420 files have been kept as unseen data for testing the performance of the automatic BI labeling system.

Table 1 shows the counts of the BI tags found in the training data.

TABLE 1 Training data counts

Tag	Total Count
0	194
1	1320
2	3948
4	1410
Total	**6872**

Table 2 shows the counts of the BI tags found in the testing data.

TABLE 2 Testing data counts

Tag	Total Count
0	86
1	1097
2	2663
4	1012
Total	**4877**

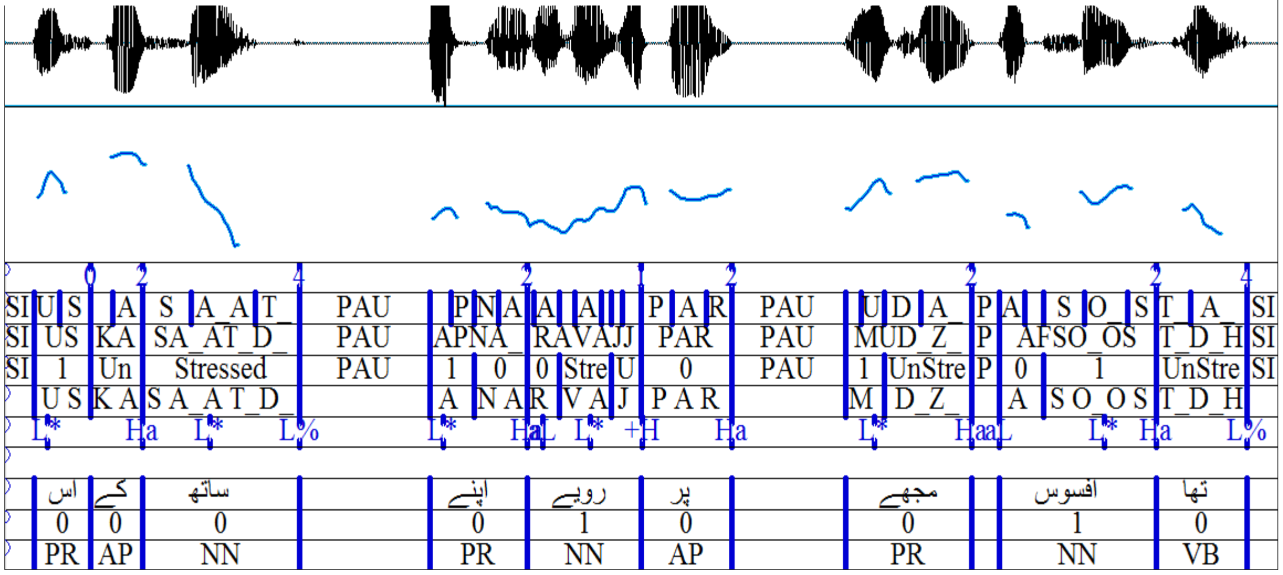
From tables 1 and 2, it can be observed that in Urdu speech corpus, BI level “2” tag has the highest frequency, whereas BI level “0” tag has the lowest frequency. This shows that accentual phrase boundary i.e. level '2' BI occurs most frequently as accentual phrase is the basic unit of Urdu prosody. BI level '0' i.e. the words using zair-e-izafat, vao izafat and the pronoun case marker combinations occur less frequently in the data selected for the phrase model analysis of Urdu.

3.2. Rules for Automating BI Tagging

As BI is marked between words and from words to silence, so the onset of each word of a sentence would not be assigned any BI level. The stress [12], intonation [13] and POS [14] tiers have already been accurately marked in the Urdu speech corpus. The information provided by these tiers is used to formulate rules for automatic BI marking.

Figure 1 shows an example of a speech file containing the Urdu sentence, “US KA_Y SA_AT_D_H APNA_Y RAVAJJA_Y PAR MUD_Z_HA_Y AFSO_OS T_D_HA_A”. (Translation: *I was sorry for my behavior with him*),

FIGURE 2 An example of a speech file that has been automatically annotated for all BI levels (0, 1, 2 and 4).



in which all the 4 levels of break indices (0, 1, 2 and 4) have been automatically labeled, according to the automatic BI labeling rules.

The rules formulated for automating the process of BI tagging are given as follows.

3.2.1. Break Index 4

1. In the first run, mark "4" if the following tag is "SIL".
2. In the second run, mark "4" aligning with every "%" symbol on the intonation tier i.e. "LH%", "H%" and "L%" as % symbol denotes a full intonation phrase boundary and is used at the end of clauses and sentences only.

In Figure 1, it can be observed that BI level '4' is marked at the end of the word "T_D_HA_A", in accordance with the first rule, as it is the last word of the sentence, followed by "SIL". At the end of the word "SA_AT_D_H", BI level '4' is marked in accordance with second rule, as it aligns with the intonation tag "L%".

3.2.2. Break Index 2

Mark 2 aligning with every "Ha" or "La" tag on intonation tier. "Ha" and "La" tags show an accentual phrase boundary. An accentual phrase usually comprises of a pitch accent and a boundary tone and it is the smallest unit of Urdu prosody instead of a word as multiple words are joined to form one accentual phrase in Urdu. In other languages e.g. English, BI level '2' is used to mark strong juncture with no tonal markings. But we have used BI level '2' for accentual phrases as accentual phrase is found in South Asian languages only.

In Figure 1, it can be observed that BI level '2' is marked at the end of words, "KA_Y", "APNA_Y", "PAR", "MUD_Z_HA_Y" and "AFSO_OS", aligning with the "Ha" intonation tag.

3.2.3. Break Index 0

Mark "0" in the following contexts:

1. At the onset boundary ofzair-e-izafat, A_Y with the POS tag CN
2. At the onset boundary of vao-izafat, O_O with the POS tag CN
3. Between the following pronouns and case markers:
   a. Between US/اس with the pos tag PR and KA_Y/کے with the pos tag AP
   b. Between UN/ان with the pos tag PR and KA_Y/کے with the pos tag AP

Personal pronouns in Urdu completely lose their word boundary when followed by certain case markers taking BI level '0' between them and behaving as one prosodic word. See Appendix A for the list of such pronouns and case markers.

In Figure 1, it can be observed that BI level '0' is marked at the end of the word "US", as its POS tag is "PR" and it is followed by the word "KA_Y" with the POS tag "AP", in accordance with the rule 3 (a).

3.2.4. Break Index 1

Mark 1 at all the remaining word boundaries as BI level '1' is used for default word boundary, when two words are not merged as in BI level '0', or there is no accentual phrase as in BI level '2', or there is no silence or full intonation phrase as in BI level '4'.

In Figure 1, it can be observed that BI level ‘1’ is marked at the end of the word “RAVAJJA_Y”, as it does not follow the rules mentioned for BI levels ‘0’, ‘2’ and ‘4’.

4. Results and Discussion

Table 3 shows the results obtained after automatically labeling the unseen testing data, and comparing it with the manually labeled gold standard corpus.

TABLE 3 Automatic BI labeling results obtained with unseen testing data.

Tag	**Total Count**	**Marked Count**	**Coverage (%)**	**Accuracy (%)**
0	86	86	100	100
1	1097	1084	96	97
2	2663	2657	99	97
4	1012	983	96	99
Total	**4887**	**4810**	**Avg=97.8**	**Avg=98.3**

From the above table, it can be observed that the level “0” tag has been automatically marked with 100% accuracy, whereas its coverage is also 100%. A very high percentage of coverage has been obtained for all of the four BI tags, with an overall coverage of 97.8%. The results obtained for accuracy are also quite promising, as an average accuracy of 98.3% is obtained.

The reason for obtaining such high quality performance is the fact that the automatic BI labeling system only utilizes the information from already accurately annotated tiers i.e. stress, intonation and part of speech (POS) tags from the speech corpus, and does not rely on extracting information from the pitch contour at run time for BI tagging.

The stress tier contains information about the stressed and unstressed syllables. The intonation tier indicates the high and low tones of the pitch at accentual phrase boundaries and pitch accents. This annotated information has been used to get an idea of the pitch contour, for marking BI in the speech corpus at any point of time.

5. Conclusion

A detailed acoustic analysis has been carried out for understanding the behavior of phrase breaks, to develop a phrase model for Urdu speech. The features considered for this purpose include POS, annotated intonation and stress information.

It has been observed that Urdu speech contains four levels of break indices i.e. 0, 1, 2 and 4, for establishing prosodic relationships between words.

The outcomes of this analysis have been used to develop an automatic BI labeling system. The developed system has provided coverage of 97.8%, and an accuracy of 98.3% with unseen testing data, which is quite promising.

6. Future Work

In future, the automatic BI labeling utility developed during this research will be used to annotate phrase breaks in the remaining 9 hours of CLE Urdu speech corpus. This annotated corpus will be used as input to train the speech synthesis module of Urdu TTS system, to improve the naturalness of synthesized Urdu voice.

Analysis for phrase modeling of long-form Urdu speech can be carried out in order to observe the patterns of phrase breaks during the reading of long Urdu paragraphs. The outcomes of such analysis can be utilized to improve the naturalness of Urdu TTS for audio books and screen readers.

References

[1] (2019) Lexico powered by Oxford. [Online]. https://www.lexico.com/en/definition/phrase

[2] Viacheslav Klimkov et al., "Phrase break prediction for long-form reading TTS: exploiting text structure information," in Interspeech, Stockholm, Sweden, 2017.

[3] Joe Crumpton and Cindy L. Bethel, "A Survey of Using Vocal Prosody to Convey Emotion in Robot Speech," International Journal of Social Robotics, vol. 8, no. 2, pp. 271-285, April 2015.

[4] Benazir Mumtaz, Saba Urooj, Sarmad Hussain, and Ehsan Ul Haq, "Break Index (BI) Annotated Speech Corpus for Urdu TTS," in Conference of The Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Technique (O-COCOSDA) , Bali, Indonesia, 2016.

[5] Yibin Zheng, Jianhua Tao, Zhengqi Wen, and Ya Li, "BLSTM-CRF Based End-to-End Prosodic Boundary Prediction with Context Sensitive Embeddings in A Text-to-Speech Front-End," in Interspeech, Hyderabad, India, 2018.

[6] Marco Vetter, Sakriani Sakti, and Satoshi Nakamura, "Cross-lingual Speech-based Tobi Label Generation using Bidirectional LSTMs," in International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, United Kingdom, 2019.

[7] Amandine Michelas and James S. German, "Focus Marking and Prosodic Boundary Strength in French," International Journal of Phonetic Science (Phonetica), 2018.
[8] Taniya Mishra, Yeon-jun Kim, and Srinivas Bangalore, "Intonational phrase break prediction for text-to-speech synthesis using dependency relations," in International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, QLD, Australia, 2015, pp. 4919-4923.
[9] Daniil Kocharov, Tatiana Kachkovskaia, and Pavel Skrelin, "Prosodic boundary detection using syntactic and acoustic information," Computer Speech and Language, vol. 53, pp. 231-241, 2019.
[10] Jeena J. Prakash and Hema A. Murthy, "Analysis of Inter-Pausal Units in Indian Languages and Its Applications to Text-to-Speech Synthesis," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 27, no. 10, pp. 1616-1628, June 2019.
[11] R. Ravi Kiran et al., "Automatic Phonetic and Prosodic Transcription for Indian Languages : Bengali and Odia," in 10th International Corference on Natural Language Processing, Noida, India, 2013.
[12] Benazir Mumtaz, Saba Urooj, Sarmad Hussain, and Wajeeha Habib, "Stress Annotated Urdu Speech Corpus to Build Female Voice for TTS," in 18th Oriental COCOSDA/CASLRE Conference, Shanghai, China, 2015.
[13] Benazir Mumtaz, Saba Urooj, and Sarmad Hussain, "Urdu Intonation," Journal of South Asian Linguistics, vol. 10, October 2019.
[14] Tafseer Ahmad et al., "The CLE Urdu POS Tagset," in Language Resources and Evaluation Conference (LERC 14), Reykjavik, Iceland, 2014.

Appendix A

Lists of Urdu pronouns and case markers to be considered for Break Index 0 rule.

Pronouns
MA_E_N
MUD_Z
T_DUD_Z
A_AP
T_DU_U
T_DUM
HAM
IS
US
UN
SAB
VO_O
IN

Case Markers
SA_Y
NA_Y
KO_O
KA_A
KI_I
KA_Y
MA_Y_N

Development of Annotated Corpus Resources of Sindhi

Mutee U Rahman[1], Tafseer Ahmed[2], and Muhammad Shaheer Memon[3]
[1,3] Isra University, Hyderabad, [2]Mohammad Ali Jinnah University, Karachi
muteeurahman@gmail.com, tafseer@gmail.com, shaheer.memon@isra.edu.pk

Abstract

We present ongoing work on the development of an annotated corpus resources project for Sindhi. A multi-layer annotation model is presented and experimentally applied on a subset of an existing plaintext Sindhi corpus. The multilayer model may possibly include different annotation layers like part-of-speech, morphological features, phrase structure, and dependency structure, etc. A compact POS tagset based on universal pos tags is considered for the POS annotations layer. Initially, a gold standard of 0.1 million words balanced corpus is created by using manual tagging tools with inter-annotator agreement considerations. A model is also trained with this gold standard corpus. Testing and evaluation show precision, recall, and F-measure accuracies with 97%, 96.7%, and 96.9% respectively.

1. Introduction

Annotated corpus is an important language resource used in theoretical and computational linguistics to reveal the deep linguistic structures and capture the computational properties of a natural language. Modern language technologies use these insights to develop high performance software systems with natural language processing and understanding capabilities [1]. Being under resourced language, annotated corpus resources for Sindhi are rarely available. This work presents an initiative of annotated corpus resources development project for Sindhi. Main objective is to lay down the foundations of multipurpose annotated corpus development model. A corpus development model with possibility of multiple annotation layers is presented. The proposed model is based on James Pustejovsky & Amber Stubbs model [2] with some changes. Initially this model is used to develop part-of-speech (POS) tagged corpus of Sindhi. Subset of an existing Sindhi corpus [3] is used for experimental development of pos-tagged corpus. At the outset first layer is annotated with part-of-speech tags. An obligatory POS tagset based on universal POS tags is used for annotations. Webanno [4] was initially used for manual annotations to create a gold standard for machine learning. Later on, Stanford tagger [5] was used for machine learning and automatic pos tagging. Gold standard is incrementally developed by automatic tagging and manual tweaking of wrongly tagged words in different sub-sets of corpus under consideration.

subsequent sections discuss the existing work, proposed multilayer annotation model, development of pos-tagged corpus, results, future work, and conclusion.

2. Existing Work

Only few corpus development studies for Sindhi are there which include Rahman (2010) [3], Mazhar, et., al. [6], and Syed & Bhatti (2018) [7]. In first study Rahman (2010) presented Sindhi corpus construction project. The corpus collection cleaning and organization process is discussed with plain text corpus analysis results including unigram, bi-gram, and tri-gram frequencies. This work lacks the annotation model and its implementation. In second study Mazhar, et. al., (2019) presented the development and analysis of Sindhi corpus for feature attributes and sentiment analysis. This corpus is made available as a dataset with around seven thousand (7000) entries annotated with universal POS tagset. Entries mostly include discrete sentences without any continuity of topic. Dataset includes universal pos-tags, with morphological (number, gender, and person) information, negative, positive sentiment and polarity values. The third study Syed & Bhatti (2018) presented an XML based document structure for development of Sindhi corpus. However, only document structure model is presented, and linguistic annotations are not discussed in this study.

Other related studies are mostly about pos tagger development and training, and development of tagsets for Sindhi Language [8]. [9] and [10] present POS taggers with reasonable accuracy results, however, there is no publicly available annotated corpus except [6] discussed above.

3. Annotated Corpus Development Model

As discussed above, particularly for this corpus development project a subset of an existing plain text corpus [3] is selected for experiments and final annotations. However, the overall corpus development model is shown in Figure 1. Various phases of corpus development process are summarized in the figure. Guidelines include the necessary documentation regarding what annotators need to know about the corpus and its overall design including the corpus subset selection criteria, annotations, and annotation process guidelines. Selected corpus segments, and tagset alongwith guidelines are given to annotators for manual annotations. Different phases of the presented model are discussed in subsequent sections.

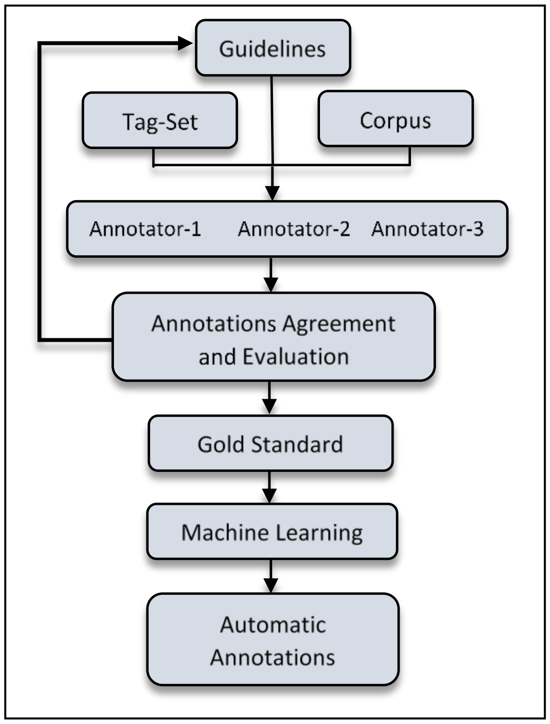


Figure 3. Annotated Corpus Development Model

3.1 Annotation Structure & Representation

The annotation model is designed as a multilayer model where each annotation layer is independent of other layers. This model is inspired by Stand-off annotation by Character Location [11]. This not only solves the white-space tokenization problems but allows simultaneously different layers on same text token/entity with possibility of links between them. Table 1 shows a three-layer sample of layered annotation model with part-of-speech tag, morphological feature tag, and syntactic function tag layers.

Table 6. Layered Annotations

Text	ڪيو	پروسو	تي	مون	چوڪريءَ
POS Tags :	VERB	NN	ADPP	PRON	NN
Morph Tags:	SMPAST	SMNOM	OBL	SGOBL	SFNOM
FUNC-Tags:	VC	NP-POF	PP-OBL		NP-SUB

XML representation of above model are as given below:

```
<TEXT> ڪيو پروسو تي مون چوڪريءَ</TEXT>
<POSTAGS>
 <NN id="N0" start="1" end="7" text="چوڪريءَ" />
 <PRON id="P0" start="9" end="11" text="مون" />
 <ADPP id="A0" start="13" end="14" text="تي" />
 <NN id="N1" start="16" end="21" text="پروسو"/>
 <VERB id="V0" start="23" end="25" text="ڪيو"/>
</POSTAGS>
<MORPHTAGS>
 <SFNOM id="SFO0" start="1" end="7" text="چوڪريءَ"/>
 <SOBL id="SO0" start="9" end="11" text="مون" />
 <OBL id="O0" start="13" end="14" text="تي" />
 <SMNOM id="SMN0" start="16" end="21" text="پروسو" />
 <SMPAST id="SMP0" start="23" end="25" text="ڪيو" />
</MORPHTAGS>
<FUNCTIONALTAGS>
 <NPSUB id="NS0" start="1" end="7" text="چوڪريءَ"/>
 <PPOBL id="PO0" start="9" end="14" text="مون تي"/>
 <NPPOF id="NPF0" start="16" end="21" text="پروسو"/>
 <VC id="VC0" start="23" end="25" text="ڪيو" />
</FUNCTIONALTAGS>
```

Layers (<POSTAGS>, <MORPHTAGS>, <FUNCTIONALTAGS>) contain tags of different categories. For example, <POSTAGS> layer contains NN (Common Noun), PRON (Pronoun), ADPP (Postposition), and VERB tags. Multiple tags within same category have unique id attributes followed by starting and ending position of a token being annotated in the text. It can be seen that multiple layers can mark same location (token) with different tags without disturbing each other. For example, in case of token "چوڪريءَ" ("girl" a common noun with singular, feminine, nominative features) pos tag layer marks it as a common noun tag "NN", morphtags layer marks it with singular feminine and nominative features (SFNOM), and functional tags layer marks the same token as noun phrase subject (NPSUB) function. Overlapping can also be observed where multiple tags of one layer are part of single tag of another layer. This can be seen in functional tags layer where PPOBL (Postpositional Oblique Phrase) spans over the start position 9 to ending position 14 marking single token at functional layer, whereas other layers have two different tags within the same span.

3.2 Tag-Set Considerations

Sindhi has rich morphological constructions as compared to its neighboring languages. Along-with various sub-classes of different parts of speech morphological feature include number, gender, and case in nouns. Morphology also includes rich pronominal suffixation system with nouns, verbs, postpositions, and adverbs. Verbs also have complex morphological causative system. To avoid extra granularity levels initial experimental design of tag-set includes only major parts of speech categories. Morphological features are considered as a separate layer and are not discussed in this paper. POS tagset considered for tagging is based on Universal POS tags [12] and is shown in Table 2.

Table 7. Obligatory Tagset Based on Universal POS Tags

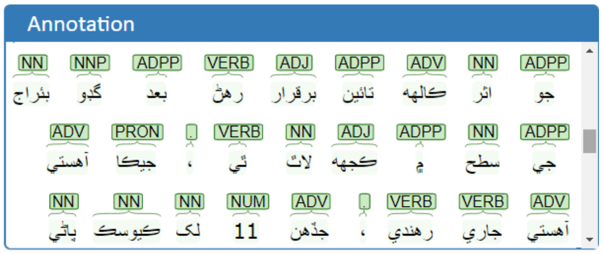
S.No.	POS	POS-Tag
1.	Common Noun	NN
2.	Proper Noun	NNP
3.	Pronoun	PRON
4.	Adjective	ADJ
5.	Adverb	ADV
6.	Preposition	ADP
7.	Postposition	ADPP
8.	Conjunction	CONJ
9.	Interjection	INTJ
10.	Particle	PRT
11.	Negation	NEG
12.	Punctuation	.
13.	Number	NUM
14.	Other Symbols / Unknown	X

3.3 Corpus Selection for Annotations

Two sections (representing two different genres of text) of existing corpus [3] are selected for annotations. Selected corpus sections include news and folk stories Reason behind the selection of these two genres is that news section contains written language with well-formed sentences and folk stories contain vocabulary used by common people in everyday life. Together these two genres represent the Sindhi language of everyday use. 0.1 million words corpus from these two genres (approximately half from each genre) is annotated and used as gold standard for machine learning to automate the pos-tagging process.

3.4 POS Tagging Process

As discussed earlier that selected corpus is annotated with parts of speech tags. Three different annotators were given segments of text for manual POS tagging. WebAnno [4] tool was used for manual POS tagging. Figure 2 shows the snapshot of pos tagging screen in WebAnno.



NN	NNP	ADPP	VERB	ADJ	ADPP	ADV	NN	ADPP
ٻڙاج	گڏو	بعد	رهڻ	برقرار	تائين	ڪالهه	اثر	جو
ADV	PRON	.	VERB	NN	ADJ	ADPP	NN	ADPP
آهستي	جيڪا	،	ٿي	لاٽ	ڪجهه	۾	سطح	جي
NN	NN	NN	NUM	ADV	.	VERB	VERB	ADV
پاڻي	ڪيوسڪ	لک	11	ڄڏهن	،	رهندي	جاري	آهستي

Figure 2. Screenshot of Webanno Tagging Window

Manually annotated segments were then discussed among three annotators to sort out the differences in annotations. These three agreed upon tagged segments were finally combined to have an initial gold standard for machine learning. Stanford pos-tagger was trained on this data and training model was used to tag text segments automatically. These automatically tagged segments were again given to annotators for review and corrections. Correct segments were incrementally added to gold standard. This process is shown in Figure 3. During this process the usability of compact POS-

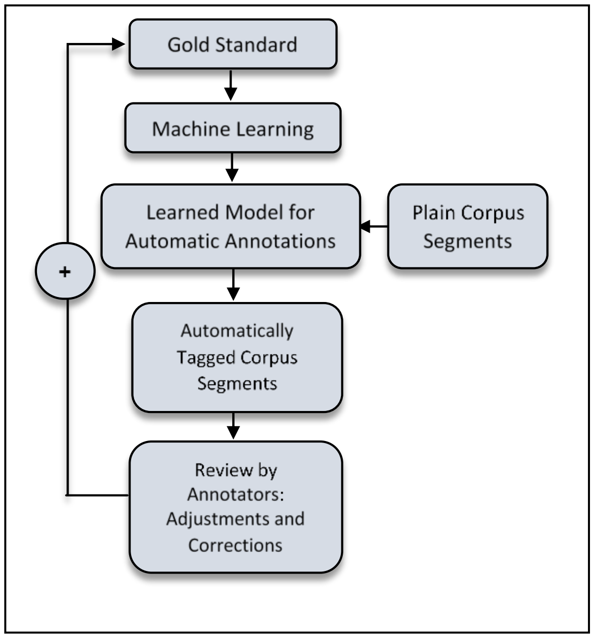

Gold Standard
Machine Learning
Learned Model for Automatic Annotations
Plain Corpus Segments
+
Automatically Tagged Corpus Segments
Review by Annotators: Adjustments and Corrections

Figure 3. Gold Standard: Incremental Development Process

Tagset being considered was also discussed and evaluated by annotators.

4. Discussion, Results, and Evaluation

Selected subsets of Sindhi corpus include newspaper and folk stories genres. The major reason behind selecting these two genres was their representativeness of language. During corpus analysis it was found that newspaper corpus represents well formed written language which does not necessarily include the spoken language flavor. Folk stories on the other hand are transcriptions of stories narrated by folk storytellers and include rich linguistic features of language used in everyday life. It was found that most interesting linguistic features including the pronominal suffixes, causatives, and inflectional variations were more frequent in folk stories. In contrast newspaper corpus rarely included those features and is mostly comprised of formal written language.

As discussed earlier, internal structure of developed corpus is represented as XML based standoff annotation by character location. This notation is LAF (Linguistic Annotation Framework an ISO standard) [13] compliant. Despite of internal XML based representation, the annotated corpus is easily representable by using common inline tagged notation format. Screenshot of an automatic annotation result generated by Stanford Tagger is shown in Figure 4. It may be noted that this output shows the POS layer in inline format where "_" underscore is used as a tag separator.

```
هڪڙي_ADJ ڏينهن_NN دستور_NN موجب_ADPP
گدڙ_NN ۽_CONJ شينهڻ_NN جيئن_PRON جهنگ_NN
مان_ADPP شڪار_NN لاءِ_ADPP پيچرو_NN
ڏئي_VERB پئي_VERB ويا_VERB ،_.
ته_PRT اوچتو_ADV کين_ADPP شينهن_NN
جون_ADPP گجڪارون_NN ٻڌڻ_VERB ۾_ADPP
آيون_VERB ._. شينهڻ_NN ٻڌايس_NN ته_PRT :_.
شينهن_NN خوشيءَ_NN ۾_ADPP نچن_VERB
ٽپن_VERB پيا_VERB ._. اتي_ADV گدڙ_NN ڏيپ_NN
کان_ADPP ڏڪڻ_VERB لڳو_VERB ،_. ڏڌهن_NN
شينهڻ_NN پڇيس_VERB ته_PRT :_. ميرخان_NNP ،_.
خير_NN ته_PRT آهي_AUX ،_.
```

Figure 4. Output of Trained Stanford POS Tagger

By using incremental approach shown in Figure 3 and discussed in section 3.4, 0.1-million-word corpus is tagged and verified as a gold standard. Model trained by using this gold standard is used for automatic POS tagging. Sample output of such tagging is shown in Figure 4. Tagger produces results with 97.0% and 96.7% precision and recall respectively. The given F-measure results are 96.9%. Reasonable accuracy is achieved by trained POS tagger. Most of the error patterns are with the tokens where same token form has more than one POS tags. For example, token "بند" can either be a common NN (stanza) or a VERB (close). Few errors are due to probabilities of noun clusters where inner proper noun NNP or adjective ADJ is tagged as common noun NN. For example, in cluster "صورتحال مڪمل طور" the inner token "مڪمل" is tagged as common noun NN instead of adjective ADJ due to higher probability of three common noun clusters in training data.

5. Conclusion and Future Work

Linguistic resources for Sindhi are rarely available, this work provide the basis for annotated Sindhi corpus resources development with different kinds of annotations. As an experiment compact version of POS tags based on Universal POS tags is used to annotate the selected segments of existing pre-processed and cleaned corpus. First the usability of the compact POS tagset was analyzed and annotators did not find any major problem while annotating the corpus using compact POS tagset. Second, the tagged corpus was used to develop gold standard for machine learning this helped the annotators to speed up the annotation process. Gold standard is evaluated by using machine learning model of Stanford POS tagger and results show reasonable precision and recall accuracy between 96 – 97%. These experimental results are encouraging, and the corpus development is being extended to other layers incrementally. Corpus distribution model is also being worked out to share the corpus resources. This will help to build more robust computational resources and models for Sindhi language processing. We also plan to use this model to develop annotated corpus resources for other Pakistani languages.

9. References

[1] P., James, and A., Stubbs. "*Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*." O'Reilly Media, Inc.", 2017. pp 1 – 23.
[2] P., James, and A., Stubbs. "*Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*." O'Reilly Media, Inc.", 2017. pp 106.
[3] M. URahman ., Towards Sindhi Corpus Construction, *Linguistics and Literature Review* 1(1): UMT. 2015., 39- 48.
[4] S. M., Yimam, I. Gurevych,., R. E., de Castilho, & C. Biemann, "WebAnno: A flexible, web-based and visually supported system for distributed annotations." in *Proc. of the*

51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations August 2013, pp. 1-6.
[5] C., Manning, M., Surdeanu, J., Bauer, J., Finkel, S., Bethard, & D., McClosky. “The Stanford CoreNLP natural language processing toolkit.” In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations* June, 2014. pp. 55-60.
[6] M. A. Dootio, & A. I. Wagan. “*Unicode-8 based linguistics data set of annotated Sindhi text*”. Elsevier Data in brief, *19*, 2018. pp.1504-1514.
[7] Z. Bhatti, and M. Shah. "Sindhi Text Corpus using XML and Custom Tags" *Sukkur IBA Journal of Computing and Mathematical Sciences* 2.2, 2018. pp.30-37.
[8] M., URahman. “Developing a Part of Speech Tagset for Sindhi”. In proc. Of the Conference on Language and Technology UET Lahore. 2012.
[9] J. A., Mahar., and G. Q., Memon. "Probabilistic Analysis of Sindhi Word Prediction using N-Grams." *Australian Journal of Basic and Applied Sciences* 5.5, 2011. pp. 1137-1143.
[10] R., Motlani, H., Lalwani, M., Shrivastava, & D. M. Sharma. “Developing part-of-speech tagger for a resource poor language: Sindhi.” In *Proc. of the 7th Language and Technology Conference LTC 2015, Poznan, Poland*.
[11] P., James, and A., Stubbs. “*Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*.” O'Reilly Media, Inc.", 2017. pp 87 – 103.
[12] S., Petrov, D., Das, R., McDonald. "A universal part-of-speech tagset" arXiv preprint arXiv:1104.2086 2011.
[13] N. Ide, and R. Laurent. "International standard for a linguistic annotation framework." *Natural language engineering* 10.3-4. 2004. pp. 211-225.

Improving Time Efficiency of TF-IDF Algorithm for Dynamic Data Streams

Sidra Saher, Qurat-ul-Ain Akram*, Kashif Javed and Sarmad Hussain
Al-Khawarizmi Institute of Computer Science
(University of Engineering and Technology, Lahore)
{sidra.sehar, ainie.akram, kashif.javed, sarmad.hussain}@kics.edu.pk*

Abstract

Different applications including search engines, plagiarism detection systems and recommender systems need to crunch the data frequently after a specific period of time. The data indexing and retrieval for such applications is becoming popular research area due to availability of huge electronic content through Internet, which is growing rapidly on daily basis. TF-IDF based term weighting scheme is commonly used to extract features of a document which are used for relevant document searching. In this paper, TF-IDF algorithm is analyzed and an efficient implementation of TF-IDF algorithm is proposed to handle such dynamic text data. Two major improvements of the traditional TF-IDF algorithm are proposed; (1) Algorithm-1: Expansion of IDF using logarithm properties and (2) Algorithm-2: Store lookup of term frequency, document frequency and IDF for reuse in next batch and efficiently update IDF of terms. The systems are evaluated on dataset of 100,000 Urdu documents. Traditional TF-IDF algorithm takes 2676520 ms (10256 ms for IDF calculation) to process complete dataset. Algorithm-1 takes 2676272 ms (10008 ms for IDF calculation), showing time efficiency in IDF calculation. Algorithm-2 is specifically designed to handle the dynamic growth of the data, which calculates the new IDF of a term using the previously computed IDF. This algorithm takes 707823 ms to process 100,000 documents. Another contribution of this study involves reduced memory consumption for TF-IDF vectors using the sparse representation of vectors. This reduces 8,945,445 MB to 353 MB to store TF-IDF of 11,724,976 terms computed from 100,000 Urdu documents.

1. Introduction

Due to rapid advancement in the development of Information and Communication Technologies (ICTs), electronic content is easily accessible in the form of blogs, online articles, books, magazines, newspapers, research papers and theses etc. To provide real time accessibility of the relevant content, an efficient information retrieval system is important. The research in the fields of data mining, search engines and similarity computation among documents for plagiarism detection is becoming popular to handle huge dynamic text data which is increasing on a daily basis. Processing of such large dataset is a challenging task. To develop such applications, text data is processed in such a way so that meaningful information can be extracted efficiently.

Various term weighting schemes such as Mutual Information, Okapi, LTU [1-3]and TF-IDF are used to process massive datasets. Among all these schemes, TF-IDF based term weighting scheme is commonly used technique [4] to extract features of the document which are used for searching the relevant document, text mining and text classification etc.

To process the text for similarity computation, data is processed and converted into the structured form which can easily be used for similarity computation. For this, three structures are used for document representation, which include inference network [5], probabilistic [6] and Vector Space Model (VSM). Among all these models, VSM is most widely used model [7]. The VSM contains a vector representation of weighted terms of a document.

There are two ways to find the similarity between the documents; (1) Local similarity and (2) Global similarity. Local similarity refers to the technique which tries to find the sequence of words which are same in two documents. Usually cosine similarity based on vector space model of terms (*n-grams, n>1*) is used to find the similarity between two documents. However, Global similarity based techniques rely on the unigram do not cover the context. The similarity results of such technique are meaningless in cases when two documents have exactly similar terms but their sequence is never the same. Therefore, researchers prefer to use local similarity for applications which requires sequence of the words to be matched [8].

For huge datasets, term weighting schemes are applied for query retrieval or similarity calculation purposes. A lot of research in this area has emphasized on importance of using phrases as terms. Due to improved accuracy of results, bigram and trigram based phrases are preferred over unigrams [9]. In case of Global similarity [8], similarity score between two documents is highly unreliable. Therefore, we cannot predict the documents to be exactly similar irrespective of high similarity score. Due to this reason, trigrams are used as terms for enhanced accuracy of the system.

To develop real time system for search engine, plagiarism detection system etc. huge amount of dataset is required. In addition, it is also essential to find the document similarity with minimum time. Hence, term weighting scheme needs to be implemented efficiently. The re-computation of term weights in case of dynamically growing data on daily basis is also challenging, even if there is small change in the actual content. The major contributions of this study are given below

- Algorithm-1 incorporates an efficient implementation of TF-IDF calculation for huge data.
- Algorithm-2 is an efficient implementation of TF-IDF calculation for dynamically growing data on daily basis.
- The memory consumption of TF-IDF vectors is also reduced by using sparse representation of vectors instead of dense vector representation.

The rest of the paper is organized as follows: Section 2 describes related work. Section 3 provides explanation of proposed algorithm by using logarithm properties and lookup storage to improve computational time of TF-IDF. Section 4 involves a brief discussion of dataset used for experimentation. The experimental evaluation and their results are discussed in Section 5 and finally Section 6 concludes research study.

2. Related work

The idea of inverse document frequency was first proposed by Jones [10]. Jones emphasized the idea of specificity by introducing the concept of collection frequency i.e. the words which are less frequent in a collection should have more weight [10]. Salton and Yu [11] used same technique in information retrieval and observed retrieval effectiveness by using precision and recall as evaluation metrics [11]. In addition, Salton and Yu [11] also figured out that the TF-IDF algorithm performed well in document retrieval. TF-IDF is a term weighting scheme which assigns weights to terms depending on their significance in the corpus. Its major principle is based on the fact that a more frequent term in a document with less frequency in overall documents (i.e. document frequency) will have high TF-IDF weight and vice-versa.

The calculation of TF-IDF is carried out using (1) which is based on the term frequency (TF) and inverse document frequency (IDF), and are calculated by using following equations

$$TF - IDF(t,d) = TF(t,d) * IDF(t) \qquad (1)$$

Term frequency is calculated as follows

$$TF(t,d) = count\ of\ t\ in\ d \qquad (2)$$

Where *t, d* stands for term (i.e. trigrams in this study) and document respectively. The IDF is calculated using (3)

$$\mathrm{IDF(t)} = \log\left(\frac{\mathrm{TotalDocSize}}{\mathrm{DF(t)}}\right) \qquad (3)$$

Where, TotalDocSize is the total number of documents in the corpus, which is processed to compute the TF-IDF of all terms in corpus. In addition the DF is the Document Frequency which is computed using (4).

$$\mathrm{DF(t)} = \sum_{i=1}^{\mathrm{TotalDocSize}} \mathrm{td} \quad \begin{cases} \mathrm{td} = 1 & \text{if t in } \mathrm{d_i} \\ \mathrm{td} = 0 & \text{otherwise} \end{cases} \qquad (4)$$

Where *td* denotes the presence of a term in a document.

The calculation of the IDF is further improved by using the concept of smoothing introduced in [12] as can be seen in (5).

$$\mathrm{IDF(t)} = 1 + \log\left(\frac{\mathrm{TotalDocSize+1}}{\mathrm{DF(t)+1}}\right) \qquad (5)$$

TF-IDF is extensively explored to utilize TF-IDF for text classification, document retrieval and plagiarism detection systems [13-15]. A lot of research is done to improve TF-IDF algorithm in terms of accuracy [14, 16, 17]. In addition, a few efficiency improvements of the TF-IDF are also suggested. Bin and Yuan [18] presented a technique for the efficient computation of TF-IDF from large data using Hadoop as main framework. Hadoop supports data distribution on multiple machines. In addition Map/Reduce scheme was also used for fast calculation of TF-IDF. The main shortcoming of the work [18] is that the data is assumed to be static which means data will not be updated once it would be indexed. Gu et al. [19] used parallel cloud computing framework which is based on GPU and MapReduce to improve the efficiency of TF-IDF algorithm.

3. Methodology

In context of document relevance, n-grams based similarity calculation is used to provide very accurate results with emphasis on use of trigrams as a term [9]. TF-IDF algorithm uses VSM to represent TF-IDF computed over each document in complete corpus. In this study, an efficient implementation of TF-IDF is proposed. As a first step, Traditional Algorithm of TF-IDF calculation is implemented with efficient lookup of term frequency and document frequency. Algorithm-1 is proposed which involves improvement in time by use of logarithmic expansion for IDF formula. Most of the real world applications like search engines, plagiarism detection systems and other information retrieval applications have dynamically growing data. Addition of a few more documents compels recalculation of weights of all terms in all the documents. Therefore, Algorithm-2 is designed in such a way that when a huge corpus is indexed using TF-IDF term weighting scheme. The term frequency, document frequency and inverse document frequency lookups are stored. These lookups are then used to update TF-IDF on increment of new documents. In this way, term frequency of only newly updated documents is calculated. The IDF of new terms is calculated using document frequency lookup. The terms which do not occur in newly updated documents undergo a slight update in their already computed IDF values. It reduces the overhead of re-computation and improves efficiency. The overhead of memory is also reduced by conversion of dense representation of TF-IDF vector to sparse representation.

3.1. Traditional TF-IDF Algorithm

A very simple implementation for TF-IDF calculation is carried out in [20]. The implementation of this algorithm is computationally analyzed and an efficient implementation is proposed to reduce the redundant calculations. In order to reduce computational time, TF and DF for all documents is calculated within the same iteration over all documents. This reduces a lot of computational time. Separate lookup is maintained for each document to store the TFs. For document frequency a global DF lookup is created to maintain the document frequency of each term.

Once, the complete global DF lookup is maintained over all the documents, another lookup is also maintained for IDF (using (5)) by iterating over global DF lookup. For TF-IDF calculation, it will iterate over each term of every document just once and use the values of TF and IDF stored in respective lookups.

3.2. Algorithm-1

The original equation for IDF i.e. (5) is analyzed further. By applying the logarithm this computation is further reduced as can be seen in (6).

$$\mathrm{IDF(t)} = 1 + \log(\mathrm{TotalDocSize} + 1) - \log(\mathrm{DF(t)} + 1) \qquad (6)$$

As can be seen in (6), $(1 + \log(\mathrm{TotalDocSize} + 1))$ will be calculated only once for all the terms of all the documents. In addition $\log(\mathrm{DF(t)} + 1)$ requires to be computed for each term and another subtraction operation is required. In addition, running time of division operation for n-digit number is $O(n^2)$ and running time of subtraction operation is $O(n)$, thus saving more computational effort [21].

In traditional TF-IDF algorithm, the expression $log((TotalDocSize + 1)/(DF(t) + 1))$ is calculated P times i.e. time complexity is *O(P)*, where P denotes total number of terms in global term lookup but due to this algorithmic modification redundant calculation of $\log(\mathrm{TotalDocSize} + 1))$ will be reduced to single time calculation i.e. time complexity is reduced to *O(1)*.

3.3. Algorithm-2

Various real time information retrieval and online text similarity based applications such as plagiarism detection system and search engines are based on indexing of huge data. Majority of these systems are developed to handle dynamic data, resulting in incorporation of the results computed on the newly indexed and already indexed data. This is usually carried out by indexing the complete data (new and old documents). Once, a significant amount of data is processed (e.g. 5.5 million documents) then number of new documents is minimum may be 1-3% of the already indexed content. The document metadata information (document name, URL, last modified date, etc.) is also maintained while developing such huge systems. Therefore, before starting indexing of complete dataset (existing and new), document filtering can be applied on recently crawled data to filter all those documents which are not indexed previously based on metadata information. These documents are referred to as NewBatch and remaining documents which are already indexed documents are referred to as PreviousBatch. The NewBatch and PreviousBatch terminology is used throughout this paper.

In order to re-index complete data including data of PreviousBatch and NewBatch, a lot of computational effort and time will be utilized. Although, the number of documents in NewBatch is minimal. Based on analysis, a term can be categorized into one of the following categories

- New Terms: Terms which are only present in NewBatch.
- Common Terms: Terms which are present in both PreviousBatch and NewBatch.
- Old terms: Terms which are only present in PreviousBatch.

For new terms, term frequency, document frequency and inverse document frequency are computed using (2), (4) & (6) respectively, from documents in NewBatch. The DFs of terms which are common between PreviousBatch and NewBatch are computed from NewBatch and will be added in DFs of respective terms already stored in the respective DF lookup of PreviousBatch. Then, the IDF and TF-IDF are computed using the same traditional way.

Inverse document frequency of old terms which are only present in PreviousBatch is not required to be recomputed for obvious reasons. As document frequency lookup, inverse document frequency lookup and term frequency lookup of each term of the PreviousBatch are also maintained and stored separately to minimize the re-computation time. Thus IDFs of old terms can be calculated from already computed IDFs in PreviousBatch by addition of an expression dependent only on document size of PreviousBatch and NewBatch.

In order to calculate the IDF for old terms during re-indexing of the dynamically growing data, the respective IDFs of the PreviousBatch are processed in such a way that DocSize of the NewBatch denoted with NewDocSize, is incorporated. More precisely the IDFs are calculated using (6). Equation 6 for PreviousBatch can be written as follows

$$\mathrm{IDF_{Previous}} = 1 + \log\big((\mathrm{PreviousDocSize} + 1)\big) - \log(\mathrm{DF(t)} + 1) \qquad (7)$$

We can modify (7) for incorporating TotalDocSize when NewBatch is indexed with PreviousBatch.

$$\mathrm{IDF_{New}} = 1 + \log\big((\mathrm{PreviousDocSize} + 1) + \mathrm{x}\big) - \log(\mathrm{DF(t)} + 1) \qquad (8)$$

Where *PreviousDocSize* denotes the number of documents in PreviousBatch and *x* denotes the number of documents in NewBatch.

The term $\log\big((\mathrm{PreviousDocSize} + 1) + \mathrm{x}\big)$ can be further solved by simplifying the expression in logarithm. Let *k* denotes the term $(\mathrm{PreviousDocSize} + 1)$, then the term $log(k + x)$ can be written as [22]

$$log(\mathrm{k} + \mathrm{x}) = log\left(\mathrm{k}\left(1 + \frac{\mathrm{x}}{\mathrm{k}}\right)\right)$$

Substituting value of *k* in (8), following expression is obtained

$$\mathrm{IDF_{New}} = 1 + log\left((\mathrm{PreviousDocSize} + 1) \times \left(1 + \frac{\mathrm{x}}{(\mathrm{PreviousDocSize+1})}\right)\right) - \log(\mathrm{DF(t)} + 1) \qquad (9)$$

Using Multiplicative property of Logarithm in (9)

$$\mathrm{IDF_{New}} = 1 + \log\left(1 + \frac{\mathrm{x}}{(\mathrm{PreviousDocSize} + 1)}\right) + \log(\mathrm{PreviousDocSize} + 1) - \log(\mathrm{DF(t)} + 1)$$

$$= 1 + \log(\mathrm{PreviousDocSize} + 1) - \log(\mathrm{DF(t)} + 1) + \log\left(1 + \frac{\mathrm{x}}{(\mathrm{PreviousDocSize} + 1)}\right)$$

Clearly, the bold part is the same as (7). It can be replaced with Previous IDF value.

$$\mathrm{IDF_{New}} = \mathrm{IDF_{Previous}} + \log\left(1 + \frac{\mathrm{x}}{(\mathrm{PreviousDocSize} + 1)}\right) \qquad (10)$$

Equation 10 shows that we can update IDF of old terms by addition of an expression dependent on NewDocSize and PreviousDocSize. This expression needs to be calculated only once before update of IDF of all the old terms.

The main advantage of using this approach is that instead of calculating IDF of old terms incorporating the total document size, we just need to calculate the expression $\log\left(1 + \frac{\mathrm{x}}{(\mathrm{PreviousDocSize+1})}\right)$ only once which is based on *PreviousDocSize* and *x* i.e. NewDocSize. This approach works well for vast dynamic data streams when data is being updated frequently. In such cases IDFs can be efficiently updated without any redundant re-computations.

3.4. Avoiding Extra Memory Consumption

Next step involves intelligent representation of TF-IDF vector so that it occupies less space in memory. The TF-IDF vector size for each document is the total terms computed from the complete dataset. Since each document does not contain all the terms therefore majority of the terms contain zero in a document. To solve this issue, dense representation of TF-IDF vector of a document is converted to sparse by excluding terms having TF-IDF value of zero. An example of dense to sparse vector representation is given in Fig.1.

$$dense: [5.0\ \ 0.0\ \ 0.0\ \ 4.0\ \ 0.0\ 2.0]$$

$$Sparse = \begin{cases} indices: 0{,}3{,}5 \\ values: 5.0\ 4.0\ 2.0 \end{cases}$$

Fig.1 Dense and sparse representation

4. Dataset

Traditional Algorithm, Algorithm-1 and Algorithm-2 are tested on dataset of Urdu web pages and results are evaluated.

5.7 Million Urdu web pages are crawled [23]. Their dataset is not publically available as it is crawled from various authenticated Urdu websites. A subset of this dataset is selected for the performance evaluation of the proposed algorithms. Therefore, 0.1 Million Urdu documents are used for testing. The detailed statistics of selected data is given in Table 1.

Table 1. Data statistics

Total Documents	100000
Average Words per Document	438
Average Lines per Document	10
Average words per line	34

5. Experiments and Results

Urdu has space insertion and deletion issues. Hence, unlike English, the words cannot be extracted by processing the space. To handle this issue, a pre-processing is applied on the complete dataset to resolve such issues. Urdu word segmentation is applied to the dataset used for evaluation purpose, which converts the sequence of Urdu ligature to the best sequence of Urdu words of a sentence.

In addition, pre-processing is applied which involves normalization, diacritics and punctuation marks removal. Then content of a document is processed and trigrams as terms are extracted and stored so that TF-IDF weighting can be applied.

As a first experiment, 100,000 documents are processed to compare the performance of Traditional Algorithm and Algorithm-1. The results are given in Fig.2. As can be seen in Fig.2, the Algorithm-1 outperforms Traditional Algorithm with efficient computation results using the properties of logarithm. Traditional Algorithm takes 2676520 ms to process complete dataset for the computation of IDF, whereas Algorithm-1 takes 2676272 ms to process same dataset for IDF calculations.

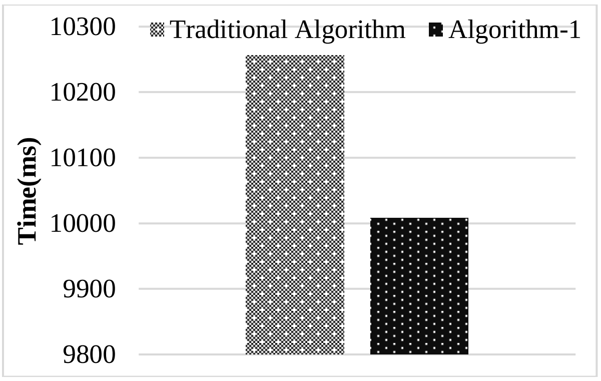


Fig. 2. Time comparison for IDF calculation

As Algorithm-1 reduces redundant computations of expression $log((TotalDocSize + 1)/(DF(t) + 1)$ by employing logarithm properties, Moreover, it converts division operation for IDF computation of each term to subtraction operation. So, the difference between the time taken for IDF calculation between Traditional Algorithm and Algorithm-1 is somehow evident.

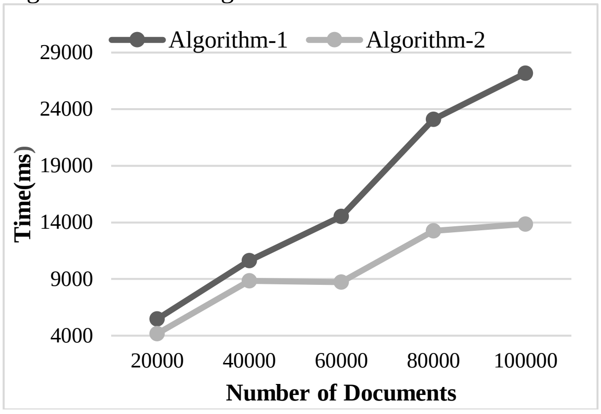


Fig.3. Time comparison for IDF calculation

The second experiment is carried out to find the difference between execution time of Algorithm-1 and Algorithm-2. While comparing IDF calculation time of Algorithm-1 with Algorithm-2, we will also include the execution time for document frequency calculation along with IDF calculation time because we store the lookups of document frequency along with IDF after the execution of each batch. Due to this reason, redundant document frequency and IDF calculation for common and old terms in PreviousBatch and NewBatch are minimized. This technique also reduces a lot of computational effort and time.

By visualizing the trend in the graph as can be seen in Fig. 3, it can be observed that increment of 20,000 documents within each new batch results in increased execution time for both algorithms. However, this increment in execution time is very mild in case of Algorithm-2 and very rapid in case of Algorithm -1.

Another point worth noticing is the trend of trigrams within each batch. The total number of unique trigrams in 100,000 documents is given by about 11 Million. If each batch introduces 20,000 new documents.

Table 2. Number of trigrams in each batch

Document Batches	***Total Trigrams***	***New Trigrams in current batch***	***Trigrams common with previous batch***	***Trigrams only in previous batch (%of total trigrams)***
Batch1(20,000)	3,239,453	None	None	None
Batch2(40,000)	6,238,318	2,998,865	732,171	2,507,282 (41%)
Batch3(60,000)	7,558,982	1,320,664	319,048	5,919,270 (78%)
Batch4(80,000)	9,743,136	2,184,154	833,066	6,725,916 (69%)
Batch5(100,000)	11,724,976	1,981,840	715,117	9,028,019 (77%)

There must be some terms which are not present in PreviousBatch and hence their IDF is calculated using (6). In our experiment 26% new terms are introduced in each batch on average. Some terms are present in previous and new batch as well hence their document frequency is updated and then their IDF is calculated. In our experiment, on average 7% common terms are generated with each NewBatch. However, the old terms are actually not being used in NewBatch and hence their document frequency does not change compelling us to update their IDF using the expression $\log\left(1 + \frac{x}{(\text{PreviousDocSize}+1)}\right)$. For each new batch, the IDF of 66% of total trigrams need to be updated using (10) on average. Their detailed statistics are shown in Table 2.

When evaluating Algorithm-2 in terms of accuracy, we find out that Algorithm-2 exhibits 100% accuracy. Although, it shows drastic reduction in computational time but its accuracy is evaluated to be same as that of Algorithm-1.

Third experiment as shown in Fig.4 is the pictorial view of time efficiency of two proposed algorithms executed over 5 batches for end to end TF-IDF calculation.

While dividing batches, it is ensured that Algorithm-1 is executed by increment of 20,000 documents in each batch because Algorithm-1 does not store any lookup for each batch. However, execution of Algorithm-2 is carried out by dividing 100,000 documents into 5 batches. Each NewBatch contains 20,000 new documents only and does not contain any document from previous batch.

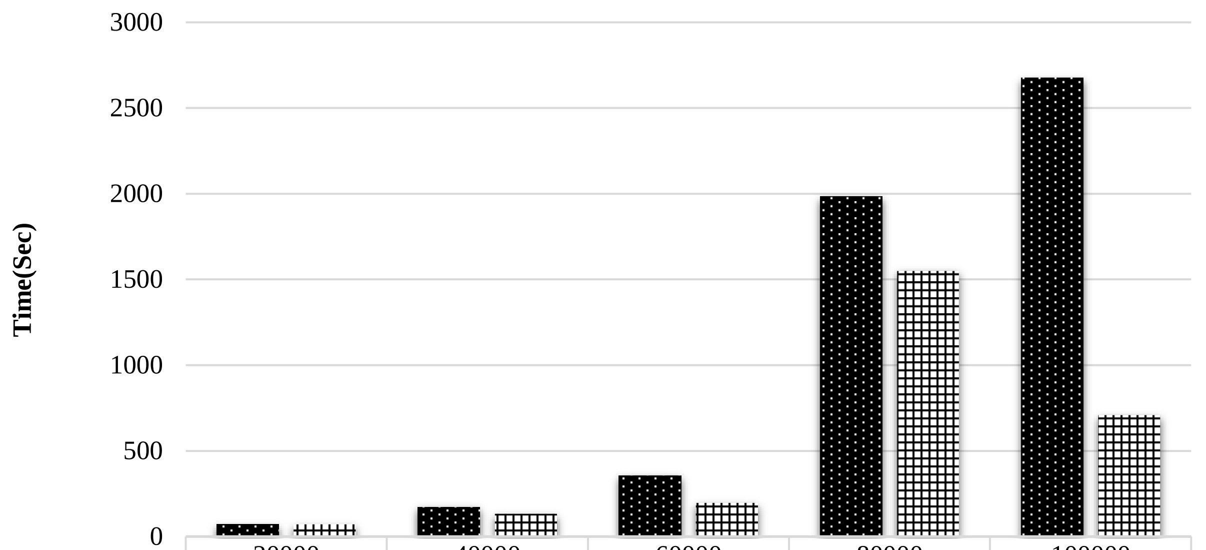
By viewing the bar values for Algorithm-1 in Fig.4, it is evident that the time for TF-IDF calculation increases with the increase in number of documents in each batch. Algorithm-1 involves variation in (5) and the effect of this variation is evident while IDF calculation. Algorithm-2 involves storage of term frequency, document frequency and IDF lookups after execution of each batch. These lookups are then used in NewBatch for IDF calculation, update and TF-IDF calculation. Similarly, we can observe in Fig.4 that Algorithm-1 takes 44 minutes for executing 100,000 documents whereas Algorithm-2 takes 11.7 minutes for execution of 100,000 documents.

Forth experiment involves observing the extent of memory reduction by incorporating sparse representation of TF-IDF vectors created after TF-IDF calculation. As total number of unique trigrams in 0.1 million documents is 11,724,976. So, creating TF-IDF matrix will occupy 1×10^5 rows and about 11.7×10^6columns. This will make the total entries of matrix as

$$\textit{No. of rows} \times \textit{No. of columns} = 11.7 \times 10^{11} \textit{entries}$$

As each TF-IDF entry is stored as a double in memory, so total memory consumed for TF-IDF matrix of 0.1 million documents is given by 8735.787 GB. This is practically almost impossible to store in main memory

By observing the vector of TF-IDF for each document, it was found that they contain a lot of null values and they are redundantly occupying memory. By converting the dense representation to sparse representation for each document. It was concluded that the total number of non-zero terms in TF-IDF vectors of 100,000 documents are 30,841,481. So, for sparse representation of 100,000 documents we need 30,841,481 double entries and same amount of integer entries as shown in Fig. 1. So, total memory occupied by TF-IDF vectors of 0.1 million documents will be 353 MB which is far less than space occupied by dense representation. Thus, we save 8735.558 GB. This is almost 99.996% reduction in memory being used in case of dense representation.



	20000	40000	60000	80000	100000
■ Algorithm-1	71.0471	172.567	356.594	1983.7024	2676.272
⊞ Algorithm-2	70.338	132.275	196.96	1549.054	707.8235

Fig. 4 TF-IDF calculation using Algorithm-1 and Algorithm-2 over 5 batches

6. Conclusions and Future Work

In this research study, the efficiency of TF-IDF algorithm is improved. The existing approaches for efficiency improvements of TF-IDF algorithm for huge amount of data involve hardware level enhancements for parallel computing. Most of the work is based on static data. In this paper, two algorithms are presented. Algorithm-1 is slight modification of efficient implementation of Traditional Algorithm. For 100,000 documents Traditional Algorithm takes 2676520 ms, whereas Algorithm-1 shows an improvement of 248 ms compared to Traditional Algorithm. On the other hand, Algorithm-2 contains stored lookups of term frequency, document frequency and IDF after execution of each batch and these lookups are used for TF-IDF calculation when each NewBatch is uploaded. It performs very well when data for TF-IDF calculation is being updated dynamically. In our experiment, for Algorithm-2, 100,000 documents are processed divided into 5 batches. Each batch exhibits an increment of 20,000 documents. Final batch has 100% accuracy and shows drastic time efficiency compared to Algorithm-1 for processing 100,000 documents. Algorithm-1 takes 2676272 ms whereas algorithm-2 takes 707823 ms for execution of 100,000 documents.

Another major contribution involves employing sparse representation of TF-IDF vectors. It saves a lot of memory and reduces 8,945,445 MB to 353 MB to store TF-IDF of 11,724,976 terms computed from 100,000 Urdu documents.

Future enhancements in this work include modifying TF-IDF algorithm in such a way that we can execute it on a number of machines concurrently and thus it will divides the execution time of TF-IDF calculation equivalent to the number of machines used for this process.

7. References

[1] M.-G. Jang, S. H. Myaeng, and S. Y. Park, "Using mutual information to resolve query translation ambiguities and query term weighting," presented at the Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, College Park, Maryland, 1999.

[2] S. E. Robertson and S. Walker, "Okapi/Keenbow at TREC-8," in TREC, 1999.

[3] C. Buckley, A. Singhal, and M. Mitra, "New Retrieval Approaches Using SMART: TREC 4," in TREC, 1995.

[4] W. Na, W. Pengyuan, and Z. Baowei, "An improved TF-IDF weights function based on information theory," in 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering, 2010, vol. 3, pp. 439-441.

[5] H. Turtle and W. B. Croft, "Evaluation of an inference network-based retrieval model," ACM Trans. Inf. Syst., vol. 9, no. 3, pp. 187-222, 1991.

[6] K. Sparck Jones, S. Walker, and S. E. Robertson, "A probabilistic model of information retrieval: development and comparative experiments: Part 2," Information Processing & Management, vol. 36, no. 6, pp. 809-840, 2000/11/01/ 2000.

[7] L. Guoping, K. Y. Lee, and H. F. Jordan, "TDM and TWDM de Bruijn networks and ShuffleNets for optical communications," IEEE Transactions on Computers, vol. 46, no. 6, pp. 695-701, 1997.

[8] B. Stein and S. M. zu Eissen, "Near Similarity Search and Plagiarism Analysis," in From Data and Information Analysis to Knowledge Engineering, Berlin, Heidelberg, 2006, pp. 430-437: Springer Berlin Heidelberg.

[9] H. Chim and X. Deng, "Efficient Phrase-Based Document Similarity for Clustering," IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 9, pp. 1217-1229, 2008.

[10] K. Sparck Jones, "A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL," Journal of Documentation, vol. 28, no. 1, pp. 11-21, 1972.

[11] G. Salton and C. T. Yu, "On the construction of effective vocabularies for information retrieval," SIGIR Forum, vol. 9, no. 3, pp. 48-60, 1973.

[12] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," Computer Speech & Language, vol. 13, no. 4, pp. 359-394, 1999/10/01/ 1999.

[13] S. Basnayake, H. Wijekoon, and T. Wijayasiriwardhane, "Plagiarism detection in Sinhala language: A software approach," Gloria Scientiam - Golden Jubilee Commemorative Volume, Faculty of Science, University of Kelaniya (2017), 10/01 2017.

[14] H. Wu and N. Yuan, "An Improved TF-IDF algorithm based on word frequency distribution information and category distribution information," presented at the Proceedings of the 3rd International Conference on Intelligent Information Processing, Guilin, China, 2018.

[15] K. Sugathadasa et al., "Legal Document Retrieval Using Document Vector Embeddings and Deep Learning: Proceedings of the 2018 Computing Conference, Volume 2," 2019, pp. 160-175.

[16] L. Cheng, Y. Yang, K. Zhao, and Z. Gao, "Research and Improvement of TF-IDF Algorithm Based on Information Theory," in The 8th International Conference on Computer Engineering and Networks (CENet2018), Cham, 2020, pp. 608-616: Springer International Publishing.

[17] I. Yahav, O. Shehory, and D. Schwartz, "Comments Mining With TF-IDF: The Inherent Bias and Its Removal," IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 3, pp. 437-450, 2019.

[18] L. Bin and G. Yuan, Improvement of TF-IDF Algorithm Based on Hadoop Framework. 2012.

[19] Y. Gu, Y. Wang, J. Huan, Y. Sun, and W. Jia, An Improved TFIDF Algorithm Based on Dual Parallel Adaptive Computing Model. 2018, pp. 657-663.

[20] AdnanOquaish. (2019, 17- 09- 2019). AdnanOquaish/Cosine-similarity-Tf-Idf-. Available: https://github.com/AdnanOquaish/Cosine-similarity-Tf-Idf-

[21] En.m.wikipedia.org. (2019, 18-09-2019). Computational complexity of mathematical operations. Available: https://en.m.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations

[22] E. Series and S. Art. (2019, 28-11-2019). Mathematics Stack Exchange. Available: https://math.stackexchange.com/questions/2033324/express-lnx-with-a-3-as-taylor-series

[23] H. M. Shafiq, B. Tahir, and M. A. Mehmood, "Towards building a Urdu Language Corpus using Common Crawl," presented at the LKE 2019 : 7th International Symposium on Language & Knowledge Engineering, Dublin, Ireland, Oct 29, 2019 - Oct 31, 2019, 2019.

Removing the Gender and Tense Discrepancies in Online English to Urdu Translators

*Tariq Naeem[1], Shanza Salomi[2], Aman Ullah Khan[3]

[1,2,3] Department of Computer Science & Engineering, Air University Multan Campus, Pakistan

*[1]naeemtarik@aumc.edu.pk (*corresponding author), [2]shanza.atta@outlook.com, [3]auk@aumc.edu.pk*

Abstract

Existing translators like Google Translate, Bing Microsoft Translator and Collins Translator does not identify gender and tense cases in translation from English to Urdu. This paper, primarily based on, ruled based methodology to machine translation for English (source language) to Urdu (target language), handling tense identification and semantic translation of gender cases. Data was collected gathered from published resources like BBC Urdu, newspapers, magazines, novels and literature. POS (Part of Speech) tag model and POS-based reordering techniques are presented. Analysis and findings segment of the research article elucidated our testified outcomes in detail. The proposed approach achieved 79% accuracy. The developed application allows contributing towards information to comprehend writing, logical inquires and literature in Urdu language and translate it in equivalent English language.

Keywords: Online Translators, Rule based Approach, Gender and Tense Case

1. Introduction

Natural Language Processing (NLP) has been now introduced as an interdisciplinary course in the fields of artificial intelligence, linguistics and computer science that is very helpful in exploring the usage of computers in understanding and manipulating the text or speech of natural language [1]. Computer software automatically translates the text using Machine Translation (MT) methods [3]. In MT, the source natural language is converted automatically into a new-targeted language, however, keeping the meaning of the input text original and fluent in the output language, as shown in Fig 1.

There are three architectural classifications of MT systems i.e. Direct, Transfer and Interlingua architectural [8]. In addition, RBMT, EBMT, SMT and hybrid are the approaches of MT system [5].

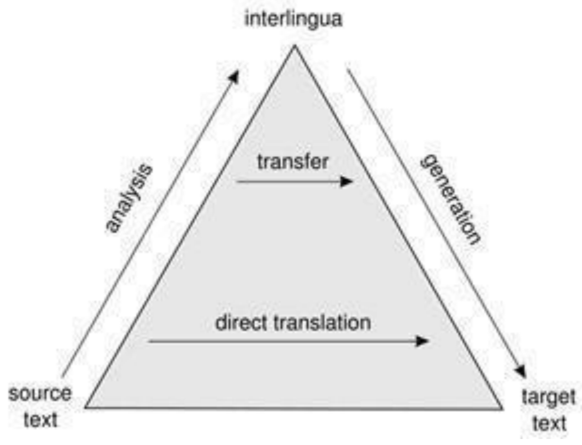


Fig 1: Machine Translation Architecture [8]

Direct Architecture is the basic type of translation substituting source language (SL) lexeme with the target language (TL) lexeme; Interlingua approach necessities a top to bottom semantic investigation and the TL is produced through this methodology. Transfer approach lies in the middle of the two boundaries; it deals with the syntactic dimension and includes semantics in a few spots. The syntactic arrangement of SL is explored to develop a processed structure, plot the rules to change it into TL arrangement, and interpretation is then produced with the use of TL definite rules.

MT provides many benefits like without the human translators a large amount of text is converted from one natural language to other language which reduce spending of money and time with less human efforts [7] Google Translate is an online translator that is a free of cost text translation system, which implies the statistical machine translation patterns and provides translations for 55 different languages or more. Besides this Microsoft Translator is also available which implies EBMT and different SMT translation systems. RBMT system is implied by Systran. English, Chinese, Arabic, Dutch, French and other languages are converted by Systran. Most of the other languages worked in pairs with English or French [18]. Another one Babel Fish is a web based translator that sutepport multilingual translation. Babel Fish translator provide services to translate webpages among 36 pairs and 13 languages including, Russian, Spanish, Korean, English, Dutch, French, Japanese, Traditional Chinese, Italian, Simplified Chinese, German, Greek and Portuguese [2]. The ImTranslator is a free web-based translator that performs translation of words, text or phrases between more than 90 languages. ImTranslator use statistical machine translation (SMT), this online translator use statistical methods that is based on multilingual texts. Statistical machine translation use the existing translation (done by human translators) of source and target language for making new rules to translate between language. The accuracy of the translator is increase by using statistical machine translation approach.

Issues and problems does exist in machine translations that make it difficult such as order of words, idioms, ambiguity in word sense, preposition, post-position, awareness of gender and context because natural languages are very complex. There are multiple meanings of most words, various readings are available for sentences and grammatical rules of one language may differ from other languages. Besides this, there are other non-linguistic aspects such as the word knowledge and language morphology is needed for carrying out translation [6] and [7].

The rest of paper is composed as follow. Section 2 explains the literature review and related work. Section 3 illustrates the strategies and methodology. Results and discussion about current work is illustrated in section 4. Section 5 closes the article with future work indicators.

2. Literature Review

Development of framework for translation from English to Urdu is considerably insufficient in comparison with the number of Urdu speakers [13]. More work is highly needed in this field. Urdu has been ranked at 19th number out of 7,105 languages spoken all over the world. It is one of the most common languages in South Asian region [11]. About 5% to 10% people only can speak or understand English in Pakistan [9], [21] and [10].

According to [12], for developing and implementing a translator all grammar of SL must be developed with bottom-up parsing algorithms. After acquiring the parse tree of SL sentences, it is then translated according to those grammatical rules in to TL sentences. In [14], corpus-based MT system was proposed to tackle problems like syntactic and structural ambiguity, anaphoric resolution and discourse analysis using data mining and text mining tools.

MT system for English to Bodo [5] uses general domain English-Bodo parallel text corpora. But the computational system containing information of Bodo language was not enough and it required more expansion.

The authors of [15], proposed a method for translating Malayalam text into English. For this purpose, rule-based machine translation system was used. The system comprises of bilingual dictionaries and conversion rules. These rules were implied for text conversion from SL to TL. In case of multiple meanings in English of a Malayalam word the proposed system generated multiple sentences.

Whereas, for translating English text into Urdu an expert system using Unicode Standards for translation was proposed in [16]. The Unicode worked with a knowledge base which contained grammatical patterns of English and Urdu, as well as a tense and gender-aware dictionary of Urdu words (with their English equivalent forms). In order to avoid the problems occurred in case of multiple meaning of a single word AGHAZ solution was implemented. A parsing-based reordering technique was presented in [17] which were used for English-to-Japanese phrase translation. The phrase-based translator is used to increase the performance of translation through reordering technique. The reordering technique was also used in the preprocessing stage for syntax base translation.

In [18], the author focused on the rule-based case transfer, as shown in Fig 2, which was a part of the transfer grammar module developed for bidirectional Tamil to Malayalam MT system. The presented study involved two typologically close and genetically related languages, Tamil and Malayalam. They considered the basic construction of sentences which was highly dependent on the case systems. The rules were written by taking into consideration the postpositions and cases in the languages. A parallel corpus was chosen and deep analyses of the case transfer patterns were drawn and rules were written to sort out the case changes that happen when translating from SL to TL. Web data was used for evaluation and the results were encouraging.

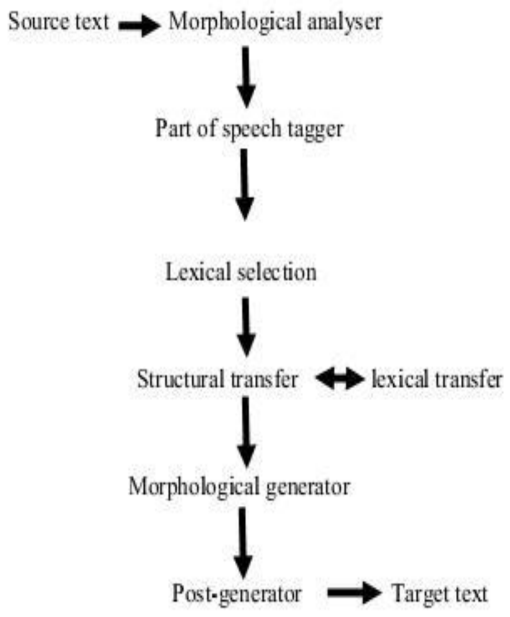


Fig 2: Architecture of RBMT [18]

As [19] presented, a system outline of English to Hindi Machine-Aided translation system named AnglaHindi. This translation system utilizing rule-based and example-based approaches, with some statistics to achieve more satisfactory and precise translation for regular verbs and nouns phrases. This approach to some degree combined hybridization of rule-based and example-based.

A translator that translate English text into Arabic by using rule based machine translation approaches was also used feed-forward back propagation of Artificial Neural Network (ANN) was proposed by [4], [20] and [25]. The proposed system translates sentences that have prepositional objects, gerunds, direct and indirect objects, infinitives, etc. Neural networks worked with bilingual dictionaries that do not have the meanings of English words into Arabic but it also store all the linguistic details of words of one language to other languages.

A translator proposed by [9] to translate English text into Urdu text through the use of example based MT. The English and Urdu translator supported homographs, idioms, and helped out other features that had the aptitude of the bilingual corpus to grow.

Some examples from traditional MT systems are discussed below. Several sentences were taken from books, magazines, and online forum to test the gender and tense cases.

She is playing. **(1)**

وہ کھیل رہا ہے-[22]

وہ چلا رہی ہے-[23]

وہ چلا رہی ہے-[24]

Amna is my friend. **(2)**

امینہ میرے دوست ہے- [22]

آمنہ میرا دوست ہے- [23]

آمنہ میرا دوست ہے- [24]

The boys wanted to help him. **(3)**

لڑکوں کو اس کی مدد کرنا چاہتا تھا-[22]

لڑکوں نے اس کی مدد کرنا چاہتا تھا-[23]

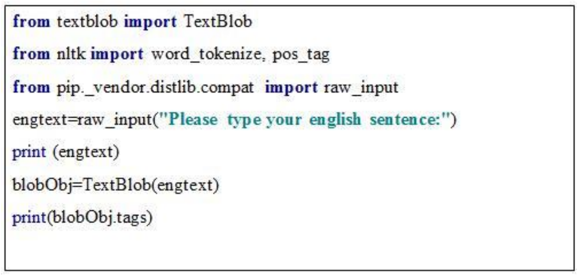
لڑکوں نے اس کی مدد کرنا چاہتا تھا-[24]

In these above statements, Google Translator [22], Microsoft Bing Translator [23] and Collins Translator [24] showed semantically incorrect translation output. A huge demand comes from users who are not familiar with English to build an automated system, which can cope with such issues. For this, the English to Urdu machine translator is designed and developed, which is a web-based system for providing correct language translation semantically.

3. Methodology

Natural Language Processing Toolkit (NLTK) is a main platform for structuring Python programs to work with language data. It gives simple interfaces to more than fifty corpora and lexical resources such as Word Net. It also provides a suite of content handling libraries for stemming, tagging, tokenization, classification and semantic reasoning.

Apart from having a discussion forum, NLTK also cover the highly develop industrial libraries using recent NLP methodologies.

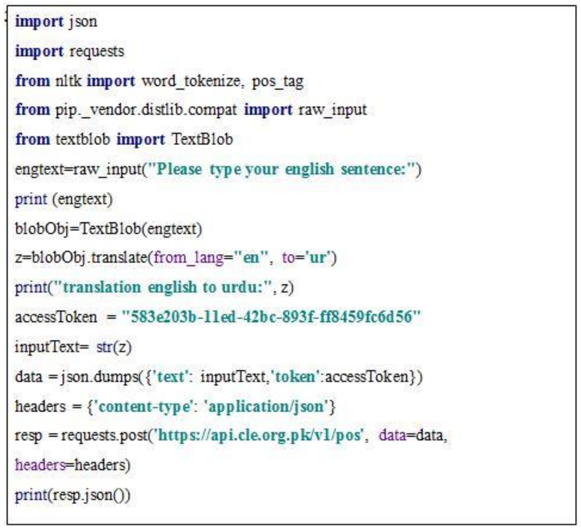


```
from textblob import TextBlob
from nltk import word_tokenize, pos_tag
from pip._vendor.distlib.compat import raw_input
engtext=raw_input("Please type your english sentence:")
print (engtext)
blobObj=TextBlob(engtext)
print(blobObj.tags)
```

Fig 3: Code for English POS tags

TextBlob is a python library utilizing NLTK. Practically, all required tasks needed in essential NLP works well as a framework with TextBlob.

Apart from it, TextBlob has some advance features. For instance, Part-of-speech tagging (see Fig 3), sentiment analysis, noun phrase extraction, classification (Naive Bayes, Decision Tree), lemmatization, language transis a leading platlation by Google, word and phrase frequencies, tokenization (splitting text into words and sentences), word inflection (singularization and pluralization) and spelling correction.



```
import json
import requests
from nltk import word_tokenize, pos_tag
from pip._vendor.distlib.compat import raw_input
from textblob import TextBlob
engtext=raw_input("Please type your english sentence:")
print (engtext)
blobObj=TextBlob(engtext)
z=blobObj.translate(from_lang="en", to='ur')
print("translation english to urdu:", z)
accessToken = "583e203b-11ed-42bc-893f-ff8459fc6d56"
inputText= str(z)
data =json.dumps({'text': inputText,'token':accessToken})
headers = {'content-type': 'application/json'}
resp =requests.post('https://api.cle.org.pk/v1/pos', data=data,
headers=headers)
print(resp.json())
```

Fig 4: Code for Urdu POS tags

TextBlob translate any English sentences in Urdu that is entered by user see Fig 4. TextBlob have the feature of POS (part of speech) tagging on English sentence. For instance:

She goes for a walk daily. **(4)**
[('She', 'PRP'), ('goes', 'VBZ'), ('for', 'IN'), ('a', 'DT'), ('walk', 'JJ'), ('daily', 'NN')]
{'response': {'status': 'ok', 'tagged_text': 'PRP| وہ RB| روزانہ VBF| چلتا NN|ہے '}}

The boys wanted to help him. **(5)**
[('The', 'DT'), ('boys', 'NNS'), ('wanted', 'VBD'), ('to', 'TO'), ('help', 'VB'), ('him', 'PRP')] {'response': {'status': 'ok', 'tagged_text': 'NN| لڑکوں PSP| کو PRP| اس PSP| کی NN| مدد VBI| کرنا AUXM| چاہتا NN|تھا . '}}

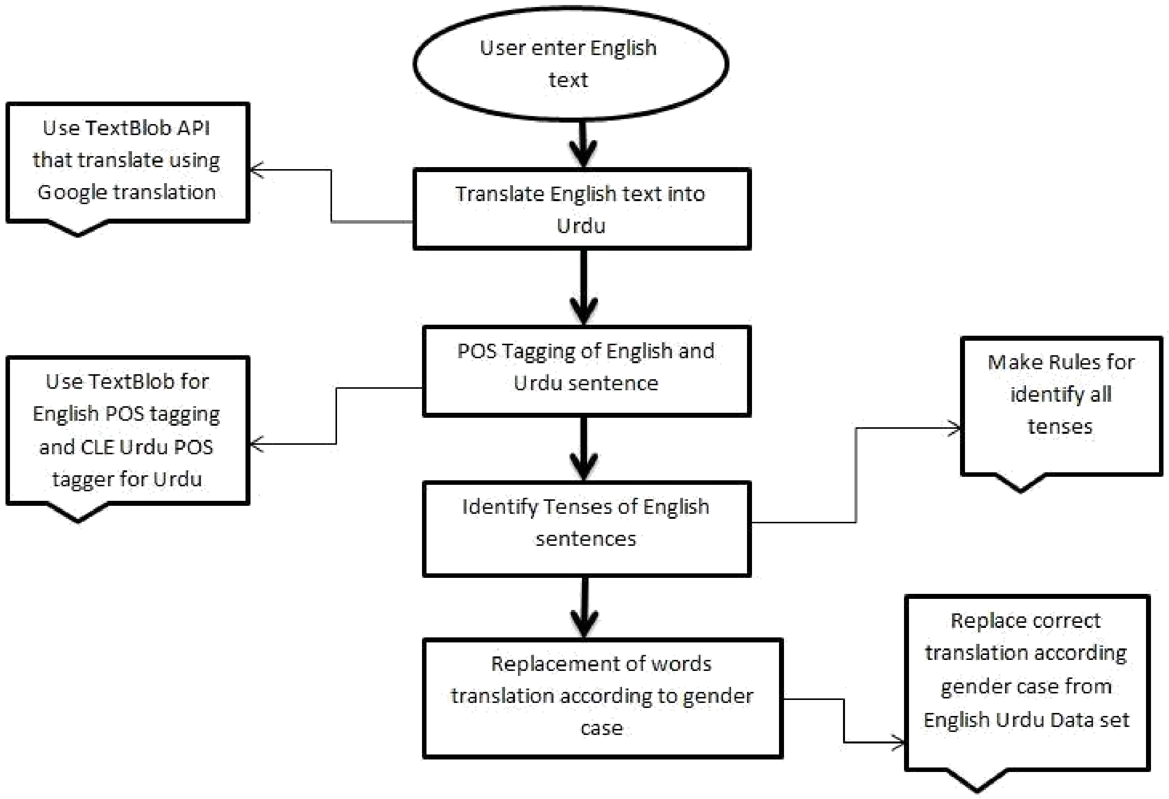

User enter English text
Translate English text into Urdu
Use TextBlob API that translate using Google translation
POS Tagging of English and Urdu sentence
Use TextBlob for English POS tagging and CLE Urdu POS tagger for Urdu
Make Rules for identify all tenses
Identify Tenses of English sentences
Replacement of words translation according to gender case
Replace correct translation according gender case from English Urdu Data set

Fig 5: Implementation of English to Urdu Translator

In these sentences, TextBlob Part of Speech tagging technique separates the POS tags. After POS tagging for Urdu sentences were applied. The proposed system, shown in Fig 5, use Urdu POS tag set of CLE (Center for Language Engineering) for Urdu part of speech tagging [26]. Urdu tag set of CLE contributions as “Urdu word sense annotation tool” is developed to run a simple interface for word sense labeling and confirming labeling stability. After identifying of tenses, the data set for English Urdu meanings was developed then the incorrect translation with correct translation was replaced.

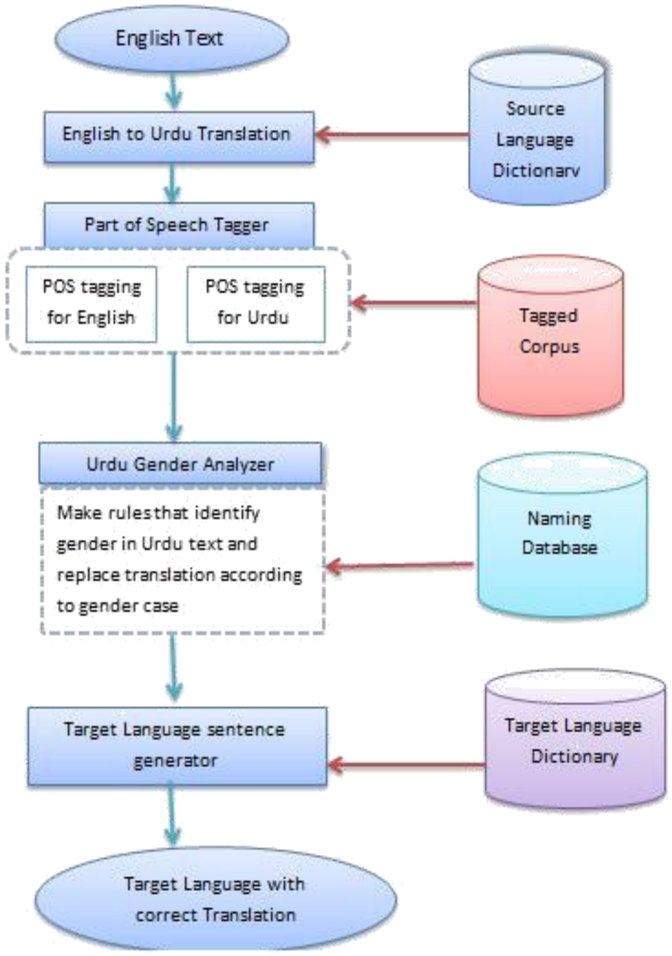

English Text
English to Urdu Translation
Source Language Dictionary
Part of Speech Tagger
POS tagging for English
POS tagging for Urdu
Tagged Corpus
Urdu Gender Analyzer
Make rules that identify gender in Urdu text and replace translation according to gender case
Naming Database
Target Language sentence generator
Target Language Dictionary
Target Language with correct Translation

Fig 6a: System Framework for Gender Analysis

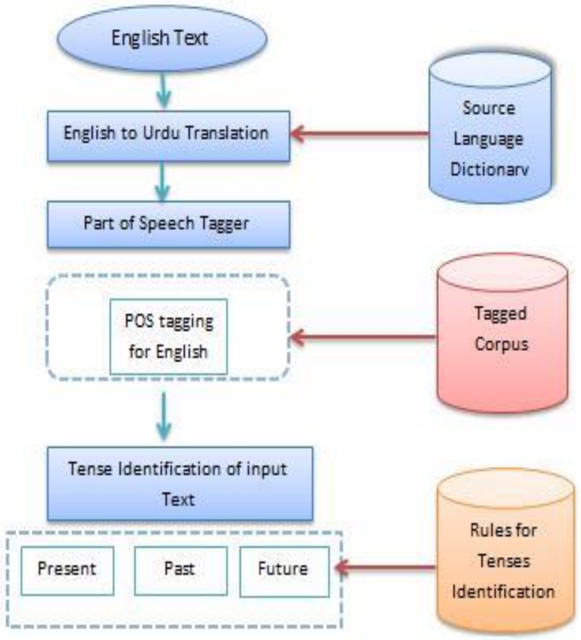

English Text
English to Urdu Translation
Source Language Dictionary
Part of Speech Tagger
POS tagging for English
Tagged Corpus
Tense Identification of input Text
Present
Past
Future
Rules for Tenses Identification

Fig 6b: System Framework for Tense Analysis

We have presented our framework in which we have explained a process of part of speech tagging of English text. Identification of gender and tense case of a given English text is shown. A clear view of the tense identification in framework that is illustrated above in Figure 6.

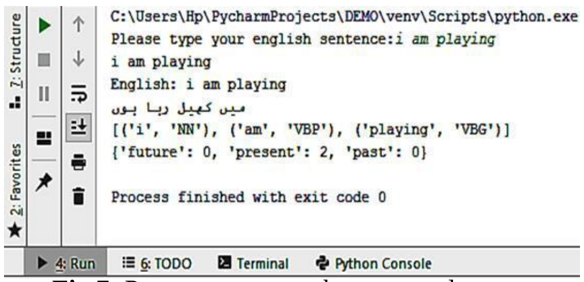
4. Result and Discussion

We checked our English to Urdu translator with various sentences of English and Urdu language.
From our proposed framework, we have corrected both tense and gender cases.

4.1 Tense Case

In tense case we have tested our system with present, past and future tense sentences. They were accurately identified by our proposed method.

I am playing. **(6)**
میں کھیل رہا ہوں۔


```
C:\Users\Hp\PycharmProjects\DEMO\venv\Scripts\python.exe
Please type your english sentence:i am playing
i am playing
English: i am playing
میں کھیل رہا ہوں
[('i', 'NN'), ('am', 'VBP'), ('playing', 'VBG')]
{'future': 0, 'present': 2, 'past': 0}

Process finished with exit code 0
```

Fig 7: Present tense case by proposed system

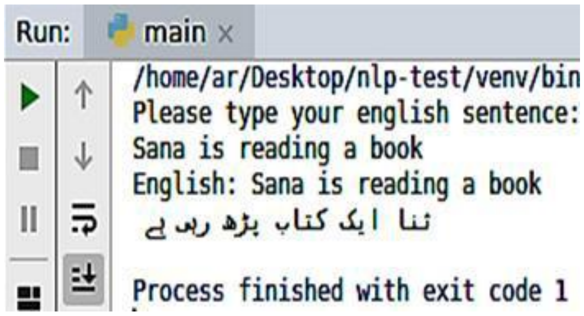
She worked in office for three years. (7)
انہوں نے دفتر میں تین سال تک کام کیا۔

We shall play together. (8)
ہم ساتھ مل کر کھیلے گے۔

4.2 Gender Case

Some examples of English sentences, whose incorrect translation by [22], [23] and [24] were corrected by our proposed system i.e.

Sana is reading a book. (10)
ثنا ایک کتاب پڑھ رہی ہے



```
Run:  main ×
/home/ar/Desktop/nlp-test/venv/bin
Please type your english sentence:
Sana is reading a book
English: Sana is reading a book
 ثنا ایک کتاب پڑھ رہی ہے

Process finished with exit code 1
```

Fig 8: Gender case by proposed system

5. Error Analysis

A sample of 190 English sentences was taken from books, magazines and online platforms given as input to the proposed system. Out of 190 sentences, the system achieved semantically accurate translation of 150 sentences.

The accuracy of the system was calculated simply using the percentage formula, i.e.

$$(150/190) * 100 = \mathbf{78.9\%}$$

Due to long and complex English sentence structure, the system could not generate semantically correct translation.

6. Conclusion

This system remove the gender discrepancies in online English to Urdu translators because the existing translators like Google Translate, Bing Microsoft Translator and Collins Translator does not identify the gender (male or female) of the person names in translation from English to Urdu. If we write the name of female in English text, Google translator gives the translation according to male gender but this research, primarily based on, ruled based approach of source language to target language, handling semantic translation of gender cases and tense identification. Our MT system supports POS (Part of Speech) tags models use for tagging of English and Urdu text and our English to

Urdu MT system gives accurate translation according to gender (Male or Female). The proposed system achieved 79% accuracy.

Although this not the first study in English to Urdu MT, however, less efforts are done to consider semantically gender cases during translation. The proposed system provide translation of simple English sentences into Urdu, we use rule based machine translation approach. The main objective was to identify these cases in English to Urdu MT system that never discussed before.

7. Future Work

The future work and extension of this work can be the extraction of accurate translation for complex long sentences. Because in long complex sentence structure a simple POS matching of one word may not work; especially if there are multiple verbs (different forms) in a sentence and one of them is incorrect. How would the authors know which one is wrong?

8. Reference

[1]. Liu, D., Y. Li, and M.A. Thomas. *A roadmap for natural language processing research in information systems*. in *Proceedings of the 50th Hawaii International Conference on System Sciences*. 2017.

[2]. Liddy, E., *Natural Language Processing, 2nd edn. Encyclopedia of Library and Information Science. Marcel Decker.* Inc., NY, 2001.

[3]. Khan, S. and R. Mishra, *Translation rules and ANN based model for English to Urdu machine translation.* INFOCOMP, 2011. 10(3): p. 36-47.

[4]. Antony, P., *Machine translation approaches and survey for Indian languages.* International Journal of Computational Linguistics & Chinese Language Processing, Volume 18, Number 1, March 2013, 2013. 18(1)

[5]. Islam,S.andB.S.Purkayastha, *Implementation of English to Bodo Machine Translation System using SMT Approach,*. International Journal of Computer Science & Applications, 2017. 14(2)

[6]. Costa-Jussa, M.R., et al., Study and comparison of rule-based and statistical catalan-spanish machine translation systems. Computing and informatics, 2012. 31(2): p. 245-270

[7]. Naeem T., Khan MA, Automatic derivation of Nouns from Adjectives, 6th International Conference on Language and Technology, 41-49, 2016

[8]. Islam, S., M. Devi, and B. Purkayastha, *A study on various applications of NLP developed for North-East languages.* International Journal on Computer Science and Engineering, 2017. 9(6): p. 386-378.

[9]. Alqudsi, A., N. Omar, and K. Shaker, *Arabic machine translation: a survey.* Artificial Intelligence Review, 2014. 42(4): p. 549-572.

[10]. Zafar, M. and A. Masood, *Interactive english to urdu machine translation using example-based approach.* International Journal on Computer Science and Engineering, 2009. 1(3): p. 275-282.

[11]. Hussain, S. *Urdu localization project: Lexicon, MT and TTS (ULP)*. in *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*. 2004, Association for Computational Linguistics

[12]. Malik, A.A. and A. Habib. *Qualitative Analysis of Contemporary Urdu Machine Translation Systems*. in *NLPAR@ LPNMR*. 2013. Citeseer.

[13]. Ahmed, T. and S. Alvi, *English to Urdu translation system.* manuscript, University of Karachi, 2002.

[14]. Revanuru, K., K. Turlapaty, and S. Rao, *Neural Machine Translation of Indian Languages.* 2017.

[15]. Tahir R., Asghar S, and Masood. *Knowledge based machine translation*. in *Information and Emerging Technologies (ICIET), 2010 International Conference on*. 2010. IEEE.

[16]. Rajan, R., et al. Rule based machine translation from english to malayalam. Advances in Computing, Control, & Telecommunication Technologies, 2009. IEEE.

[17]. Muhammad, U., et al., Aghaz: An expert system based approach for the translation of english to urdu. International Journal of Social Sciences, 2008. 3(1): p. 70-74.

[18]. Lee, Y.-S., B. Zhao, and X. Luo. Constituent reordering and syntax models for English-to-Japanese statistical machine translation. in Proceedings of the 23rd international conference on computational linguistics. 2010. Association for Computational Linguistics.

[19]. Lakshmi, S. and S.L. Devi, Rule Based Case Transfer in Tamil-Malayalam Machine Translation. Research in Computing Science, 2014. 84: p. 41-52.

[20]. Sinha, R. and A. Jain, Angla Due to long and complex English sentence structure, the system could not generate semantically correct translation.Hindi: an English to Hindi machine- aided translation system. MT Summit IX, New Orleans, USA, 2003: p. 494-497.

[21]. Akeel, M. and R. Mishra, ANN and rule based method for english to arabic machine translation. Int. Arab J. Inf. Technol., 2014. 11(4): p. 396-405.

[22]. Google Translator, 2018, [online]. Available:https://translate.google.com/, [Accessed: 16-Dec-2018]

[23]. Microsoft Bing Translator, 2018, [online]. Available:https://www.bing.com/translator, [Accessed: 16-Dec2018]

[24]. Collins Translator, 2018, [online]. Available:https://www.collinsdictionary.com/t ranslator, [Accessed: 16-Dec-2018]

[25]. Mohsin A., Asghar S., Naeem T., Intelligent Security Cycle: A rule based run time malicious code detection technique for SOAP messages, 19th IEEE International Multi-Topic Conference (INMIC), 1-10, 2016

[26]. Urooj S., Shams S. Hussain S. Adeeba F.; Sense Tagged CLE Urdu Digest Corpus, CLE KICS, UET,1-8, 2018

Bilingual Sentiment Analysis of Tweets Using Lexicon

Farwa Iqbal[1], Amber Ayoub[2], Jaweria Manzoor[3], Rida Hijab Basit[4]
Computer Science Department
Kinnaird College for Women Lahore
{farwaiqbal786, ember.ayub}@gmail.com[1,2], {jaweria.manzoor, rida.basit}@kinnaird.edu.pk[3,4]

Abstract

Sentiment Analysis determines the emotions, attitude, feelings or behavior of people towards an event, topic, or a product. The advent and growth of social media platforms have given people opportunity to voice their opinions, reviews and share experiences. User Generated Content when analyzed for its sentiments can be helpful for various reasons such as predictive analysis, summarization of reviews, measuring popularity, acceptance of products, and much more. In this regard, various researches and studies exist, but all these studies focus on resource-rich languages like English, Chinese and Arabic. In this paper, we focus on Roman Urdu language. Around 30 million people across the world speak Urdu and mostly use Roman Urdu written in Roman script to express their views, feelings or experiences over the Internet. Keeping this in view, an approach has been proposed that performs sentiment analysis of bilingual data (English and Roman Urdu), using Lexicon based approach. In order to create domain specific lexicon, political tweets related to 2018 Elections held in Pakistan have been collected and analyzed for the sentiments expressed in them.

Keywords – Sentiment Analysis, Text, Opinion Mining, User Generated Content, Pakistan Election 2018

1. Introduction

With advancements in technology and internet, the use of mobiles and laptops to access social media accounts has increased [1]. People now openly give their reviews and opinions about anything, thus making it necessary to analyze the content generated by them. Substantial amount of work related to sentiment analysis on structured languages like English, Chinese and Arabic exist, but limited work has been done on Roman Urdu or Urdu languages [2]. Majority of the people in subcontinent are not much well versed in English language and use Urdu to express their sentiments on social platforms. People tend to express their opinions in Urdu mostly using Roman script known as Roman Urdu also due to the limited availability of keyboards in Urdu; thereby emphasizing upon the need of sentiment analysis in Roman Urdu. The importance of research in Roman Urdu has also been highlighted in other researches [2], [3] and [4]**.** Many techniques and methods for sentiment analysis exist but we have focused on the development of a domain specific lexicon of 3900 words that consists of Roman Urdu and English Language. Generally, sentiment analysis has been carried out using adjectives, but sentiment analysis approach presented in this paper makes use of lexicon built using adjectives, verbs, adverbs and nouns for improved sentiment analysis. The analysis has been performed on the data collected from Twitter using several hash tags based on election, from different pages of political parties, anchors etc. For detailed experimentation and to improve results two different datasets with 5031 tweets and 4177 tweets are considered. This paper is divided into five sections. Section 1 gives an introduction, Section 2 focuses on literature survey, Section 3 discusses the methodology, Section 4 compiles the results and Section 5 concludes the paper.

2. Related Work

Various existing systems that have performed sentiment analysis using lexicon approach have been studied. Some of those papers depicting similar work are discussed below.

The concept of bilingual sentiment analysis using lexicon approach on Roman Urdu has been presented by [4]. The purpose of this approach is to analyze the bilingual data from twitter. Tweets are collected based on the keywords related to four main political parties. SentiStrength is used for extracting the sentiments for the English language but for Roman Urdu a new lexicon is created to provide sentiment strength to the Roman Urdu words. SentiStrength along with English to Roman Urdu dictionary are utilized to create bilingual sentiment repository which provides 3900 Roman Urdu words and 1673 English words. The results depict that PTI dominates other parties in general whereas in Lahore, public opinion is mostly in favor of PML-N.

In [5], analysis of the sentiments expressed in news comments has been performed using lexicon based approach to get users' opinions about a certain topic. The problem of comments oriented sentiment analysis is that the user may express his/her own opinion, which is different from the original focus of discussion. This system has used a lexicon based approach to analyze the opinions by extracting comments. The lexicon used in this system is a manually created lexicon that contains 250 news items. Objective expressions have been stored in the lexicon for identifying the focus. Objects and its features are arranged in a taxonomy-based structure. Lexicon's knowledge and the user generated content are then preprocessed by using NLP techniques. The overall sentiment of the entire document is also computed by using certain weights assigned to positive, negative and neutral comments. The experiments provide an accuracy of 0.89.

In a similar study [6], the salience for an entity in the news corpus or lexicon and the polarity of each salience as positive or negative has been calculated. The system builds the news corpus from different websites including DAWN news, ARY news, Nawai Wakt and BBC Urdu news. Corpus is further divided into chunks and POS tagging is performed to create tag list. Tag list is fed into entity finding module, which selects entities meeting a certain criterion and weight for each entity is calculated. Every salience is provided with polarity using the manual polarity tagger. After assigning the polarity the corpus is searched for intensifiers like

(1) Shadid (ʃad̪id̪, Extremely)
(2) Bohut (bɔhət̪, Lots, Intense)

Polarity of the salience with intensifier gets double for example from -5 to -10. The accuracy achieved is 84.5%.

A different approach has been discussed by [7] to find the subjectivity and polarity of the tweets using lexicon based approach. Tweets have been analyzed to predict the results of elections about a certain candidate providing a comparison on the various candidates based on sentiments expressed in them. 10,000 labeled tweets have been collected, preprocessed, imported for sentiment analysis for determining the subject to overall polarity. The sentences are classified by first assigning polarities to individual words: +1 for positive words, -1 for negative words and 0 for neutral words. Then polarity of a sentence is calculated by adding polarities of occurring words and classified as positive, neutral or negative. The subjectivity (users personal view about a candidate) of the tweets has been represented by 1 whereas the objectivity by 0. After the calculation of average polarity and subjectivity, percentage of positive, negative and neutral tweets is calculated. The experiments show that candidate Hillary has received a greater number of positive tweets whereas Trump received highest number of negative tweets.

Based on the limited research in Roman Urdu sentiment analysis, we propose a Lexicon based approach to detect sentiments depicted in tweets using Roman Urdu language as it is used by many people around the world to express their opinions on social media. In this approach, we have created a domain specific lexicon containing different parts of speech like nouns, verbs, adjective and adverbs to depict sentiments.

3. Methodology

The approach presented in this paper aims on performing bilingual sentiment analysis using lexicon. The novelty of our work lies in creation of a domain specific lexicon that contains both English and Roman Urdu words containing adjectives, nouns, verbs and adverbs. For the purpose of creating domain specific lexicon, tweets related to Pakistan Elections 2018 have been collected from Twitter using Twitter APIs. The collected data is then preprocessed, cleaned (noisy data and hashtags are removed) and tokenized (stop words removal). After formation of the tokens each word is assigned a polarity by using the lexicon ranging from -1 to + 1. Polarity of the sentence is then calculated by summing up polarities of all occurring words in the sentence. Based on the polarity, each sentence is classified as positive, negative or neutral. Furthermore, results in the form of accuracy, precision, recall and F-measure are calculated with the help of confusion matrix.

3.1. Data Preparation

3.1.1. Data Collection. The data has been collected using Twitter APIs with the help of Python. Making use of the twitter developer account, tweets from 2018 elections of Pakistan have been gathered. A total of 5031 tweets are extracted using multiple political hash tags and official pages of different anchors and politicians. A total of 2673 positive tweets, 1923 negative and 426 neutral tweets have been collected. These tweets have been stored in .json file after extraction.

Table 1: Hashtags and keywords used to extract data

#PTI	#PMLN	#MQM	#corruptleader	#nayapakistan
#imrankhan	#votekoizzatdo	#nawazsharif	#maryamnawaz	#absirfimrankhan
#jiyebhutto	#maryammeriawaz	#miansaab	#tabdeeli	#shehbazsharif

3.1.2. Data Preprocessing. Extracted data has been cleaned by removing all the unnecessary characters and symbols [8].

3.1.3. Translation of Urdu Tweets. At this stage, the extracted data consists of tweets in three different languages namely English, Urdu and Roman Urdu. For our proposed approach, Roman Urdu and English tweets are to be considered so Urdu tweets have been translated into Roman Urdu using online translator iJunoon.com.

3.1.4. Labeling of Tweets. After the translation, data set now contains only English and Roman Urdu tweets. As the next step, data labeling of tweets as positive, negative or neutral has been performed for each tweet by a single resource person. However, to remove partiality while labeling data for the dataset with 4177 tweets, data labeling has been performed using crowdsourcing technique. The tweets are labelled according to the sentiments in them like

(1) IK buhat acha politician hai

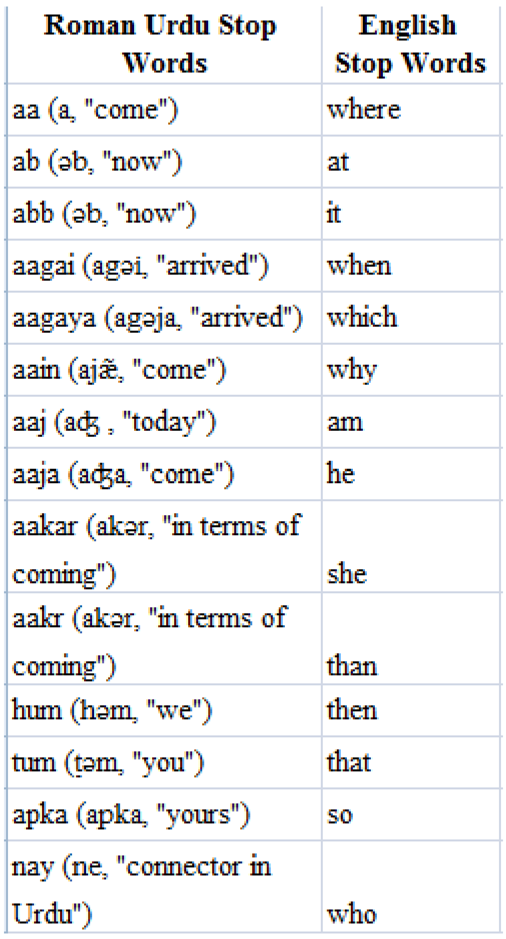
IK bəhət̪ atʃʰa politician hæ
"IK is a great politician" is labeled as positive

(2) Zardari ek corrupt insaan hai
zərd̪ari æk corrupt ɪnsan hæ
"Zardari is a corrupt person" is labeled as negative

(3) PTI or PMLN jo bhi jeete, hume vote aur elections ko izzat deni chahiye

PTI ɔr PMLN ʤo bhi ʤit̪e həme vote ɔr elections ko ɪzət̪ d̪eni ʧahie
"PTI or PMLN whoever wins, we should respect vote and elections" is labeled as neutral

3.1.5. Normalization and Tokenization of Tweets. The tweets are then normalized, where the stop words are removed from each tweet so that the meaningless words (words that play no part in sentiment analysis) like 'wo (vo, "they"), hum (həm, "we"), are, is, it' etc. are eliminated. Unlike English, Roman Urdu is not a structured language and does not have list for stop words. Therefore, we created a list translating Urdu stop words to Roman Urdu using iJunoon.com and used this with built-in Python Normalization function to remove stop words from Roman Urdu tweets.



Roman Urdu Stop Words	English Stop Words
aa (a, "come")	where
ab (əb, "now")	at
abb (əb, "now")	it
aagai (agəi, "arrived")	when
aagaya (agəja, "arrived")	which
aain (ajæ̃, "come")	why
aaj (aʤ , "today")	am
aaja (aʤa, "come")	he
aakar (akər, "in terms of coming")	she
aakr (akər, "in terms of coming")	than
hum (həm, "we")	then
tum (t̪əm, "you")	that
apka (apka, "yours")	so
nay (ne, "connector in Urdu")	who

Figure 2: List of some stop words

3.2. Lexicon based Approach

Sentiment analysis approach presented in this paper make use of lexicon of 3900 words built using adjectives, verbs, adverbs and nouns for improved sentiment analysis [4]. The reason of creating a lexicon with different parts of speech is the morphological richness of Urdu Language and Roman Urdu. Basically, Roman Urdu is Urdu language written using Roman Script where an adjective can inflect from noun or other parts of speech like

(1) daftari,

d̪əft̪əri
"official"

(2) kagazi

kaɣəzi
"Thin, scariose"
etc. [9]. Moreover, the sentiment of a complete sentence may not be depicted by adjective only like

(1) Mujhe ye phone buhat acha aur sasta lagta hai.

mʊʤʰe je phone bɔhət̪ aʧʰa ɔr səst̪a ləgt̪a hæ
"I think this phone is very good and inexpensive"

For the analysis of this sentence we also must take intensifier 'buhat (bɔhət̪, "very")' into consideration along with adjectives 'acha (aʧʰa, "good")' and 'sasta (səst̪a, "inexpensive")'.

The lexicon is created using SentiWordNet (lexical resource for Sentiment analysis and opinion mining). The sentiment scores for English words are directly taken from SentiWordNet whereas the Roman Urdu words are first translated into Urdu and then into English and then those English translations are searched in SentiWordNet for the sentiment scores. The scores assigned range from -1 to +1.

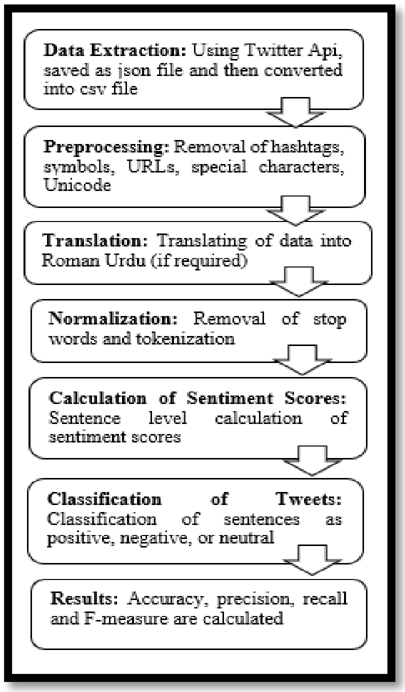

Data Extraction: Using Twitter Api, saved as json file and then converted into csv file
Preprocessing: Removal of hashtags, symbols, URLs, special characters, Unicode
Translation: Translating of data into Roman Urdu (if required)
Normalization: Removal of stop words and tokenization
Calculation of Sentiment Scores: Sentence level calculation of sentiment scores
Classification of Tweets: Classification of sentences as positive, negative, or neutral
Results: Accuracy, precision, recall and F-measure are calculated

Figure 2: Lexicon based Approach

3.2.1. Sentence Level Classification. Each tweet is normalized and converted into tokens. The java code specifically built for this purpose then assigns sentiment score (polarity) to individual tokens in each tweet using the lexicon. Then the sentiment score of the whole sentence is calculated by summing up the scores of all occurring tokens or words in the sentence similar to the approach used by [7] for sentence level sentiment classification. The classification of a sentence is based on the following conditions:

- If the resultant value is greater than 0, the sentence is classified as positive
- If the resultant value is less than 0, the sentence is classified as negative
- If the resultant value is equals to 0, the sentence is classified as neutral

4. Results and Discussion

This section discusses the results of the experiment carried out in the light of objective of this study.

Experiments are performed initially by considering dataset with 5031 tweets consisting of tweets including negation words like 'nae, ni, nahi (no)' etc. and mixed sentiments expressed in them like:

(1) Imran Khan buhat acha insaan hai, a great cricketer lakin buhat hi bura leader aur politician hai is ko kuch nahi ata pata ni umeed krni chaiye ya ni

Imran Khan bɔhət̪ aʧʰa ɪnsan hæ, a great cricket lekɪn bɔhət̪ hi bura leader ɔr politician hæ ɪs ko kuʧʰ nəhi at̪a pət̪a ni umid̪ kərni ʧahie ja ni
"Imran Khan is a good person, he is a great leader but not a good politician, he knows nothing, I don't know we should hope or not."

Better results are achieved by eliminating tweets with different sentiments and negation words. Removal of such tweets reduced the dataset to 4177 tweets only. Although there is a difference between negation words and words that depict negative sentiment like corrupt, evil, bad. We have kept the tweets like

(1) IK ek corrupt insaan hai

IK æk corrupt ɪnsan hæ
"IK is a corrupt person"

But have removed sentences like

(2) IK ek acha leader ni hai

IK æk aʧʰa leader ni hæ
"IK is not a good leader"

Furthermore, different measures have been calculated using confusion matrix for all three classes (positive, negative and neutral) separately. Confusion matrix is a form of a table that is used to represent a classification model. In the confusion matrix below, n depicts total number of tweets. Actual values of three classes depict the tweets labeled as positive, negative and neutral through human labeling. Predicted values of three classes depict the tweets labeled as positive, negative and neutral using domain specific lexicon that is created in this paper. Furthermore, accuracy, precision, recall and F-measure with respect to each class are calculated separately.

4.1. Dataset with 5031 Tweets containing Positive, Negative and Neutral Classes

4.1.1. Positive Class. Here the confusion matrix is created by taking positive class in consideration and the accuracy, recall, precision, F-measure are calculated. Our lexicon based approach of sentiment analysis predicts positive class with 81% accuracy. For positive class, TP (true positive) is the intersection of actual positive and predicted positive. FP (false positive) is the sum of values in the corresponding column, whereas FN (false negative) is the sum of values in the corresponding row excluding value of TP in both cases. TN (true negative) is the sum of all the values excluding the row and column containing positive class. The column matrix with respect to positive class is given below.

Table 2: Confusion matrix for positive class

n = 5031	**Predicted Negative**	**Predicted Positive**	**Predicted Neutral**	
Actual Negative	TN = 739	FP = 0	TN = 1193	1932
Actual Positive	FN = 0	TP =1734	FN = 939	2673
Actual Neutral	TN = 0	FP = 0	TN = 426	426
	739	1734	2558	

Measures calculated for positive class are as follow:
Accuracy = 0.81 = 81%
Precision = 1
Recall = 0.649
F Measure = 0.787

4.1.2. Negative class. Here the confusion matrix is created by taking negative class in consideration and the accuracy, recall, precision, F-measure are calculated. Our lexicon based approach predicts negative class with 76.7% accuracy. For negative class, TP (true positive) is the intersection of actual negative and predicted negative. FP (false positive) is the sum of the values in the corresponding column, whereas FN (false negative) is the sum of values in the corresponding row excluding value of TP in both cases. TN (true negative) is the sum of all the values excluding the row and column containing negative class. The column matrix with respect to negative class is depicted below.

Table 3: Confusion matrix for negative class

n = 5031	**Predicted Negative**	**Predicted Positive**	**Predicted Neutral**	
Actual Negative	TP = 739	FN = 0	FN = 1193	1932
Actual Positive	FP = 0	TN =1734	TN = 939	2673
Actual Neutral	FP = 0	TN = 0	TN = 426	426
	739	1734	2558	

Measures calculated for negative class are as follow:
Accuracy = 0.763 = 76.3%
Precision = 1
Recall = 0.383
F Measure = 0.55

4.1.3. Neutral class. Here the confusion matrix is created by taking neutral class in consideration and the accuracy, recall, precision, F-measure are computed. Our lexicon based approach of sentiment analysis predicts neutral class with 57.6% accuracy. For neutral class, TP (true positive) is the intersection of actual neutral and predicted neutral. FP (false positive) is the sum of the values in the corresponding column, whereas FN (false negative) is the sum of values in the corresponding row excluding value of TP in both cases. TN (true negative) is the sum of all the values excluding the row and column containing neutral class. The column matrix with respect to neutral class is depicted below.

Table 4: Confusion matrix for neutral class

n = 5031	**Predicted Negative**	**Predicted Positive**	**Predicted Neutral**	
Actual Negative	TN = 739	TN = 0	FP = 1193	1932
Actual Positive	TN = 0	TN =1734	FP = 939	2673
Actual Neutral	FN = 0	FN = 0	TP = 426	426
	739	1734	2558	

Measures calculated for neutral class are as follow:
Accuracy = 0.576 = 57.6%
Precision = 0.167
Recall = 1

F Measure = 0.286

4.1.4. Results with 5031 tweets. The results with this dataset show that positive class is predicted by lexicon more accurately as compared to negative and neutral classes. The accuracies of positive, negative and neutral classes are not encouraging because there is a huge difference in their actual and predicted values. The reason of bad performance is that the lexicon created in our approach does not handle negation which are words like ‘not’ in English and ‘nahi, nai, nae, nayi, ni (nəhi, nəi, nəi, nəi, ni” not/no”)’ in Roman Urdu. Therefore, lexicon based approach is unable to predict tweets containing mixed sentiments correctly and labels them as neutral. Moreover, for this dataset, labeling has been performed by only one person contributing to the poor performance to some extent.

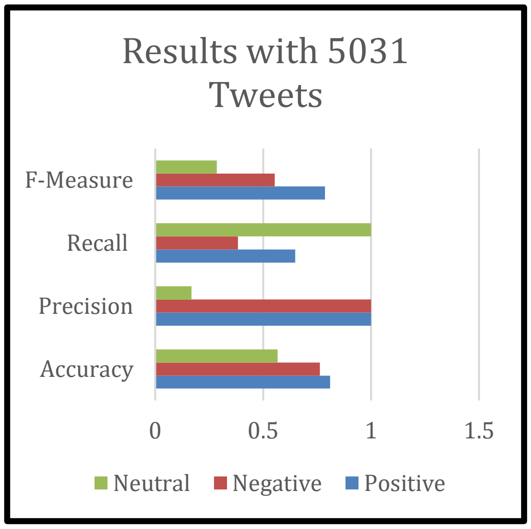


Figure 2: Results with 5031 tweets

4.2. Dataset with 4177 tweets containing positive, negative and neutral classes

To improve the performance of our proposed approach the tweets containing mixed sentiments and negation words discussed earlier are removed. Moreover, for this experiment crowdsourcing technique has been used to perform data labeling of tweets. In this technique labeling from more than one person is taken into consideration and one final labeling is deduced from those labeled tweets. The confusion matrix is then built using these manually labeled tweets along with tweets where sentiment score has been computed using domain specific lexicon.

4.2.1. Positive class. Here, the confusion matrix is created by taking positive class in consideration and the accuracy, recall, precision, F-measure are calculated. The lexicon based approach predicts positive class with 98% accuracy. Confusion matrix for this class is depicted below which is constructed like positive class with 5031 tweets given in the previous section.

Table 5: Confusion matrix for positive class

n = 4177	**Predicted Negative**	**Predicted Positive**	**Predicted Neutral**	
Actual Negative	TN = 417	FP = 0	TN = 0	417
Actual Positive	FN = 81	TP = 1411	FN = 0	1492
Actual Neutral	TN = 150	FP = 0	TN = 2118	2268
	648	1411	2118	

Measures calculated for positive class are as follow:
Accuracy = 0.98 = 98%
Precision = 1
Recall = 0.946
F Measure = 0.972

4.2.2. Negative class. Here, the confusion matrix is created by taking negative class in consideration and the accuracy, recall, precision, F-measure are calculated. Our lexicon based approach predicts negative class with 94% accuracy. Confusion matrix for this class is depicted below which is constructed like negative class with 5031 tweets given in the previous section.

Table 6: Confusion matrix for negative class

n = 4177	**Predicted Negative**	**Predicted Positive**	**Predicted Neutral**	
Actual Negative	TP = 417	FN = 0	FN = 0	417
Actual Positive	FP = 81	TN = 1411	TN = 0	1492
Actual Neutral	FP = 150	TN = 0	TN = 2118	2268
	648	1411	2118	

Measures calculated for negative class are as follow:
Accuracy = 0.94 = 94%
Precision = 0.64
Recall = 1
F Measure = 0.783

4.2.3. Neutral class. Here, the confusion matrix is created by taking neutral class in consideration and the accuracy, recall, precision, F-measure are calculated. Our lexicon based system predicts neutral class with 96% accuracy. Confusion matrix for this class is depicted below which is constructed like neutral class with 5031 tweets given in the previous section.

Table 7: Confusion matrix for neutral class

n = 4177	**Predicted Negative**	**Predicted Positive**	**Predicted Neutral**	
Actual Negative	TN = 417	TN = 0	FP = 0	417
Actual Positive	TN = 81	TN = 1411	FP = 0	1492
Actual Neutral	FN = 150	FN = 0	TP = 2118	2268
	648	1411	2118	

Measures calculated for neutral class are as follow:
Accuracy = 0.96 = 96%
Precision = 1
Recall = 0.934
F Measure = 0.966

4.2.4. Results with 4177 tweets. Results show improved accuracies in all three classes on this reduced dataset as compared to the previous dataset, with highest accuracy achieved for positive class. Accuracy for positive, negative and neutral classes is better with this dataset because there is less difference in their actual and predicted values due to elimination of tweets with negation words and mixed sentiments. Moreover, crowd sourcing has been performed for manual labeling instead of using a single resource person for labeling to remove any bias and impact the result positively.
.

4.3. Comparison of proposed approach with related work

In related work highest accuracy of 89% has been achieved by [5] that propose an approach to analyze sentiments depicted in news comments using domain specific lexicon consisting of 250 words. In the approach presented here, accuracy of 98% for positive class, 94% for negative class and 96% for neutral class is achieved with dataset containing 4177 tweets. This dataset includes positive, negative and neutral tweets and excludes tweets expressing mixed sentiments or tweets including negation. However, for the larger dataset of 5031 tweets with all kinds of sentiments expressed, the results show accuracy of 81%, 76.3% and 57.6% for positive, negative and neutral class respectively.

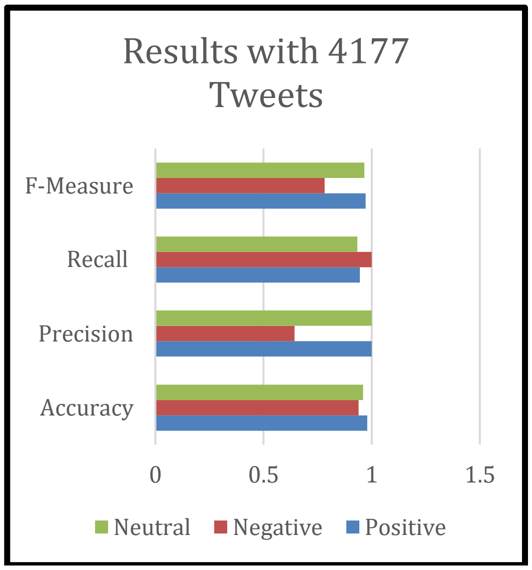


Figure 3: Results with 4177 Tweets

5. Conclusion and Future Work

People in the subcontinent mostly use Urdu language but due to the unavailability of Urdu keyboards, Roman script is used to write Urdu language which is known as Roman Urdu. In this study, bilingual sentiments expressed in tweets including English and Roman Urdu are analyzed. The analysis has been performed by building domain specific bilingual lexicon to assign sentiment scores. Extensive experimentation has been carried out by considering different kinds of tweets. Better results are achieved with tweets dataset containing specific sentiments i.e. positive, negative or neutral as compared to mixed sentiments or sentiments making use of negation. We now aim to further build our lexicon making it more generic and applying the proposed approach to a larger dataset for further validation. We are already in the process of using hybrid technique combining Machine Learning and lexicon for identifying sentiments expressed in bilingual user generated content. Work can also be done to handle negation and to analyze those sentences which include multiple kinds of sentiments.

6. References

[1] M. Kamran and K. Moin , "Sentiment Classification of Customer's Reviews About Automobiles in Roman

Urdu," in Future of Information and Communication Conference, 2018, pp. 630--640.

[2] G. Saqib Muhammad and S. Tariq Rahim, "Twitter and Urdu," in 2018 International Conference on Computing, Mathematics and Engineering Technologies – iCoMET 2018, 2018.

[3] M. Khawar, E. Daryl and S. Kamran, "Sentiment Analysis System for Roman Urdu," in Intelligent Computing, vol. AISC 858, 2018, pp. 29--42.

[4] Javed and A. Hammad, "Opinion Analysis of Bi-Lingual Event Data from Social Networks," in ESSEM@ AI * IA, 2013, pp. 164--172.

[5] M. Alejandro, R. M, C. JL and Z. Jose Manuel, "Lexicon-based comments-oriented news sentiment analyzer system," Expert Systems with Applications, vol. 39, pp. 9166--9180, 2012.

[6] S. A. ALi , M. D. Noor, M. A. Javed , M. M. Aslam and O. A. Khan, "Salience Analysis of NEWS Corpus using Heuristic Appoach in Urdu Language," International Journal of Computer Science and Network Security, vol. 16(4), p. 28, 2016.

[7] N. Farah and H. B. Sayyada, "Sentiment Analysis to predict election results using Python," in 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018, pp. 1259--1262.

[8] B. Muhammad, I. Huma, S. Muhammad and K. Amin, "Sentiment classification of Roman-Urdu opinions using Naive Bayesian, Decision Tree and KNN classification techniques," Journal of King Saud University-Computer and Information Sciences, vol. 28(3), pp. 330--344, 2016.

[9] S. Afraz Z., A. Muhammad and M.-E. Ana Maria, "Lexicon Based Sentiment Analysis of Urdu Text Using SentiUnits," in Mexican International Conference on Artificial Intelligence, springer, 2010, pp. 32--43.

[10] H. Ahmed , K. Hoda and M. Walaa, "Sentiment analysis algorithms and applications: A survey," Ain Shams engineering journal, vol. 5(4), pp. 1093--1113, 2014.

NCL-Crawl: A Large Scale Language-specific Web Crawling System

Hafiz Muhammad Shafiq, Muhammad Amir Mehmood
Al-Khawarizmi Institute of Computer Science, UET, Lahore, Pakistan
{hafiz.shafiq, amir.mehmood}@kics.edu.pk

Abstract

There exist many cases that require language-specific web crawling, e.g., text corpus building in Natural Language Processing (NLP) domain and regional language search engine content crawling. In NLP, linguistics use text corpus for statistical analysis, checking occurrences or validation of linguistic rules within a specific language territory. Similarly, regional search engines use focused crawling to serve better quality results to the users. In this study, we build a system "NCL-Crawl" for large scale language specific web crawling using Apache Nutch crawler. For this purpose, we have customized Apache Nutch and added Compact Language Detector 2 (CLD2) module for language identification at runtime. The system also provides an option to specify minimum language bytes to avoid garbage collection in configured language. For evaluation, we have chosen the Urdu language and crawled 25,723 documents from the given seed and got very good quality pages with better accuracy. Our work is an effort towards building large scale text corpus for the NLP community especially for the low resource languages. In addition, regional search engines can effectively use NCL-Crawl for language specific web crawling.

1. Introduction

With the passage of time as Internet users are increasing, many regional search engines have appeared in the search engine market, e.g., Baidu, Yandex etc. In China, Baidu has 76.69% market share [1] and in Russia, Yandex has 45.16% market share [2]. These types of regional search engines require to crawl the WWW for a specific language at large scale. This approach not only provides better quality web documents of given language, but also helps to use minimal resources in terms of storage, bandwidth, and time [3]. Further, language-specific crawl is also required to build text corpus for linguistic researches and NLP applications. The advancement in NLP and Information Retrieval (IR) domain, e.g., summarization, cross-language information retrieval, etc., requires to build corpus in single or multiple languages at large scale [4][5].

There exists many open-source solutions to crawl World Wide Web (WWW) from small to large scale, e.g., curl, Apache Nutch, Scrapy, Heritrix etc. [6]. But for language specific crawl, none of these provides a concrete solution. In most cases, a new job is executed to find language information of crawled documents, which is both time as well as storage consuming. Moreover, language threshold based crawling, e.g., crawling documents with more than 50% Urdu content, is even more complex than former case. Some crawlers provide language filters but in most cases, it is based on web server response header and hence, it requires customization to find language information from crawled content. For instance, Apache Nutch provides "language-identifier" plugin to find language information but it is also based on web server response header.

This work is an effort to build Apache Nutch with CLD2 Language Crawl (NCL-Crawl) - A system using Apache Nutch Crawler and Compact Language Detector (CLD2) for language specific web crawling. NCL-Crawl aims to filter web-documents based on the content size in bytes of a particular language. Apache Nutch is an open-source large scale web-crawler and is developed in Java language that can be extended very easily. It has two major development branches, i.e., 1.x and 2.x [7]. We have used latter one for our experimentation. For language detection, we have integrated CLD2 with Nutch that can detect a maximum of three languages in a single document with percentage information [8]. Further, we do our customization in the fetcher module of Nutch to remove irrelevant documents at run time which also minimizes time and storage resources. For this purpose, we also added many new configuration parameters to set the language label and minimum bytes threshold.

To test our work, we collect Urdu language seed of 50 URLs from different domains and run the crawler for 40 iterations. We configure Urdu minimum threshold to 256 bytes and disable out-links. For politeness, a maximum of 50 URLs are selected in each iteration from a single domain. Our main findings in this study, are given below:

- **Yield Rate Statistics:** NCL-Crawl runs for 40 iterations and crawls 25,723 documents. From total fetched documents, 24,172 documents have Urdu content more than configured threshold. Crawling rate varies from 50 documents to 1300 during the experimentation.

- **Accuracy Measurements:** Overall 93.99% of the fetched documents have Urdu content greater than the configured threshold. For each iteration, accuracy varies from 86% to 99%.

The rest of our paper is organized as follows: In Section 2, we discuss existing work for language specific web crawling. Section 3 presents our methodology for Nutch customization and experimentation. In Section 4, experimental results are presented. Finally, we conclude our work in Section 5.

2. Related Work

For language-specific crawl for text corpus building and regional search engines, researchers have suggested various solutions. For instance, [9] has proposed a heuristics-based approach for focused web crawler. This approach uses pattern-based recognition algorithm to match the topic of crawled text. It requires a lot of space to save fetched data in each iteration and later analysis for pattern recognition. In [10], the authors have used Dictionary and Breadth-First algorithm for focused crawling to build Javanese and Sundanese Corpus. Their study shows that these two algorithms deliver the highest performance as compared to others in a focused crawl.

Further, [11] has used Semantic Similarity Vector Space Model for focused crawler improvement. Their results show better performance of focused crawler as compared to the Breadth-First model and VSM model. Similarly, [12] and [13] have used topic-based approach for focused crawling. In former, the authors have built a classifier that evaluates the relevance of a given document with respect to the topics and in latter work, a weight table is constructed with topic frequency to check the similarly of a web page.

For language-specific crawl, [14] has used linguistic graph analysis approach for crawling. The authors have analyzed web data from large crawl with specific language attributes for selection strategies. Moreover, [15] has used language locality in selecting the crawl paths from a large space of Thai weblogs for specific web crawl. Their work achieve higher performance than a naive Breadth-First crawling strategy.

Apache Nutch is one of the most matured web crawlers and has been used extensively in the research area for web crawling [16]. In [17], the authors have optimized Apache Nutch for domain-specific crawling at large scale. During experimentation, they got a success rate of only 0.0015% due to sparse data distribution and duplicate content on the Web. In our approach, we have also used Apache Nutch to build language-specific web crawler.

3. Formatting instructions

In this section, we discuss our proposed approach for language-specific crawling. First, we briefly describe Apache Nutch crawler with different phases. Next, we discuss the existing challenges in Apache Nutch for language-specific crawling. After this, we describe our proposed approach for Nutch customization in this regard. Finally, we discuss our testing environment for customized Nutch.

3.1 Apache Nutch Crawler Overview

Apache Nutch is an open-source distributed crawler to crawl the web at large scale. There exist two major versions of Nutch namely 1.x and 2.x. The latter one differs from the former with the addition of Apache Gora as a storage abstraction layer that allows to use different NoSQL databases, e.g., Hbase, Cassandra, etc., [18]. We have used Apache Nutch 2.x branch in this study. Further, each cycle of Nutch consists of many phases to complete a job as shown in Figure 1. Each of these phases have been described below:

Inject & Generate: The *inject* phase is the first phase where selected seed URLs are provided and crawler starts crawling by introducing some default score to URLs. This step is very important because the crawler will grow and fetch new web-pages based on the initial seed. The next phase in Nutch is *generate* phase where top URLs are marked for fetching based on the assigned score to URLs. Note that this score is the default for the first iteration but later on, it is calculated in *updatedb* phase of Nutch for each next iteration.

Fetching: In this phase, the crawler requests the marked URLs (in the generator phase) and fetches HTML of these pages from the World Wide Web (WWW). This job is multi-threaded and one can control the number of threads via Nutch configuration. There exist many controls in Nutch for various purposes in this phase, e.g., age filter (*filter.age.timestamp*), size filter (http:content:limit), fetcher threads per host (*fetcher.threads.per.queue*) etc. At the end of this phase, complete downloaded HTML with headers is stored in configured storage back-end, e.g., Hbase etc.

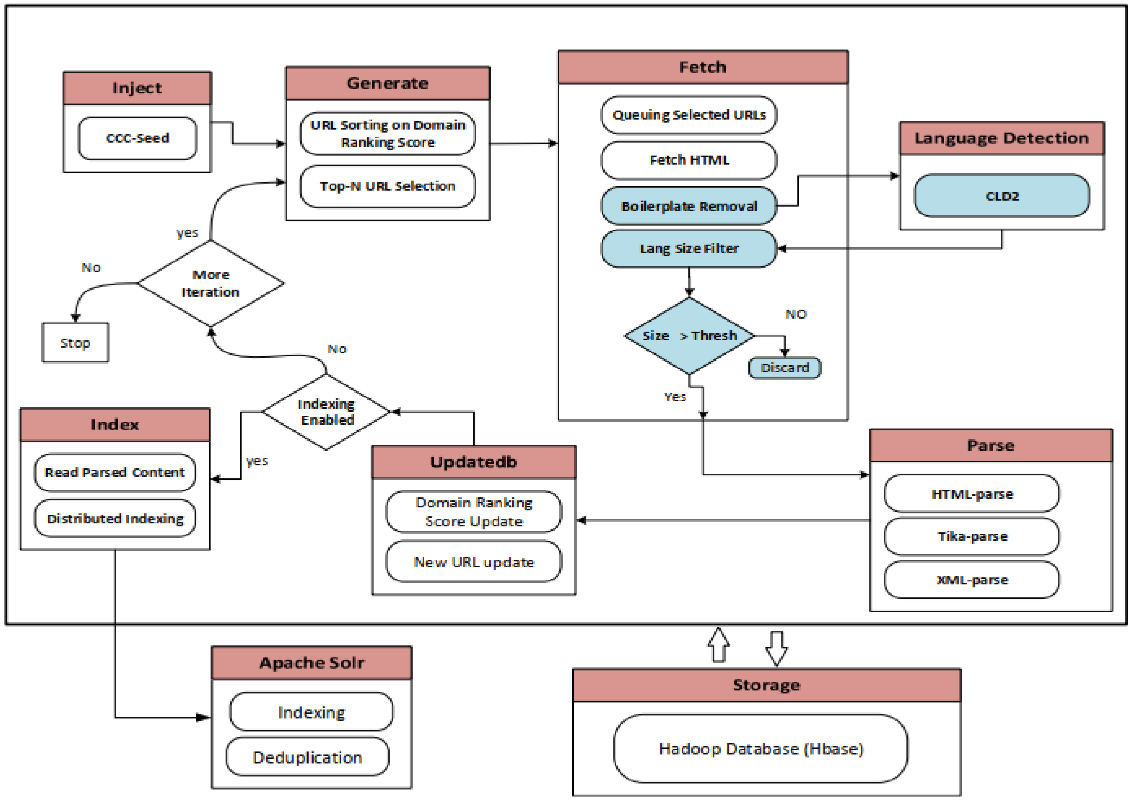
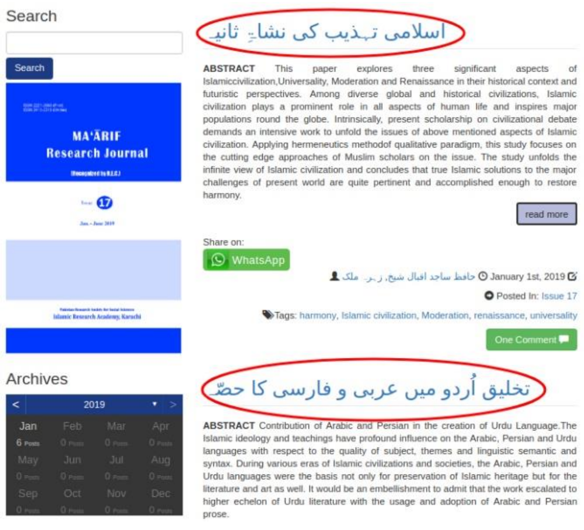


Figure 1: NCL-Crawl Execution Pipeline



اسلامی تہذیب کی نشاۃِ ثانیہ

ABSTRACT This paper explores three significant aspects of Islamiccivilization,Universality, Moderation and Renaissance in their historical context and futuristic perspectives. Among diverse global and historical civilizations, Islamic civilization plays a prominent role in all aspects of human life and inspires major populations round the globe. Intrinsically, present scholarship on civilizational debate demands an intensive work to unfold the issues of above mentioned aspects of Islamic civilization. Applying hermeneutics methodof qualitative paradigm, this study focuses on the cutting edge approaches of Muslim scholars on the issue. The study unfolds the infinite view of Islamic civilization and concludes that true Islamic solutions to the major challenges of present world are quite pertinent and accomplished enough to restore harmony.

حافظ ساجد اقبال شیخ, ز.ر. ملک January 1st, 2019

Posted In: Issue 17

Tags: harmony, Islamic civilization, Moderation, renaissance, universality

تخلیق اُردو میں عربی و فارسی کا حصّہ

ABSTRACT Contribution of Arabic and Persian in the creation of Urdu Language.The Islamic ideology and teachings have profound influence on the Arabic, Persian and Urdu languages with respect to the quality of subject, themes and linguistic semantic and syntax. During various eras of Islamic civilizations and societies, the Arabic, Persian and Urdu languages were the basis not only for preservation of Islamic heritage but for the literature and art as well. It would be an embellishment to admit that the work escalated to higher echelon of Urdu literature with the usage and adoption of Arabic and Persian prose.

Figure 2: A sample webpage with very small Urdu

Parsing: As discussed earlier, each crawled document consists of various levels of information, e.g., raw content, i.e., HTML source code, request/response headers, etc. The parse phase of Nutch parses each of this information of crawled data and saves them separately in the configured database. There are different parser plugins available in Nutch, e.g., html–parser, tika–parser, xml–parser that can be configured via Nutch configuration file.

UpdateDB & Indexing: After parsing, the next phase of crawling is to update the database using parsed documents. Many types of activities are performed in this phase, e.g., addition of in-links/outlinks, page score calculation, extra markers removal etc. There exist many configuration options for each of these actions to enable/disable these controls or to add some scoring plugin etc. For instance, db.ignore.external.links configuration parameter is used to allow the addition of external links, i.e., outlinks, in the database. Later on, these URLs get a mark possibility for fetch in *generate* phase based on their score. This is the last major phase of Nutch in the crawling cycle and the crawler can jump back to *generate* phase from here for the next cycle. Nutch also provides an indexing phase to index and search crawled content via some text search platform, e.g., Apache Solr or Elastic Search, etc. This phase should be executed after *updatedb* to include current crawled documents in index.

3.2. Language Specific Crawl and Challenges in Nutch

Although, Nutch provides language identification plugin ”language-identifier” to find language details of crawled documents, however, this plugin is based on a web-server response header and is not reliable. In most websites, this header is not properly set and in some cases, it is not even available. Further, Nutch also does not provide language information for multilingual pages with their percentage distribution.

To crawl specific websites only for text corpus building and regional search engine content, Nutch provides an option to disable out-links and crawl inlinks completely. For this purpose, one has to configure *db.ignore.external.links* property to true in Nutch configurations. Despite the fact that this approach will crawl all documents of given seed, but it cannot filter low quality documents w.r.t. given language, and hence, will cause garbage collection. For instance, Figure 2 shows such a sample page that has English as a primarily language and Urdu as a secondary language. Although, this document has Urdu content but it is of very small size, i.e., bytes. Such documents should be filtered for applications that requires very rich content in Urdu language. Unfortunately, Nutch does not provide any such option to avoid garbage collection for language specific crawling.

3.3. Nutch Customization for Language specific Crawl

In this section, we have discussed the customization of Apache Nutch for language specific crawling. First, we discuss the addition of language detection tool in Nutch and later, we discuss about the implementation of minimum language size filter to avoid garbage collection.

3.3.1. Language Detection Tool. For language specific crawling, the first and most important step is tool selection for language identification of a given language. For this purpose, there exist many open-source tools e.g., *langid, langdetect, ldig* and CLD2 [19][20][8]. Each of these tools has its own limitations and requirements. In our case, we have selected CLD2 for language identification of crawled content. CLD2 accepts only UTF-8 encoded strings and can identify 161 different languages. For a given text, it can detect upto maximum of three languages along with their percentage and total bytes. The percentage information can help to apply a minimum size filter for configured language as discussed later.

To add CLD2 module in Nutch, we decided to detect document language at runtime, and if a document is irrelevant, we truncate it at the spot. This strategy not only helps to remove documents that are not in the required language but also helps to save storage. For this purpose, we have customized Apache Nutch fetcher module that actually crawls the documents from WWW. In this module, fetched content is parsed via *Boilerpipe* library to get the main article of document. *Boilerpipe* is an open-source library developed for boilerplate removal from HTML documents [21]. Later, this extracted text is sent to CLD2 module that returns language information of the document.

3.3.2. Minimum Size Filter. In order to implement minimum size filter to avoid garbage collection, as already discussed, we cannot directly use CLD2 percentage value as a minimum threshold without language bytes information. For example, if there are two documents with a content of 1 MB and 1KB respectively and CLD2 returns 10% Urdu in both cases, then in first case, Urdu bytes are 100 KB while in second, these are just 100 bytes. Thus, first document is more rich with Urdu as compared to the second one. To cater this problem, we find language bytes from the CLD2 output using following equation:

$$LangBytes = \frac{(total\ bytes) * (lang.percentage)}{100}$$

Bytes information for above discussed documents will be as follows w.r.t. this equation:

$$Doc1\ LangBytes = \frac{(1{,}024{,}000)\ *\ (10)}{100} = 100KB$$

$$Doc2\ LangBytes = \frac{(1{,}024)\ *\ (10)}{100} = 100B$$

In order to configure language and minimum language threshold, we introduce few new configuration parameters, e.g., *filter.lang.label* and *filter.lang.minSize.bytes* etc. Complete details of all new configuration parameters are given in second part of Table 1 with default values and description. Lastly, Figure 1 shows complete workflow of Apache Nutch with language filter and minimum size filter. Our main contributions are highlighted with light blue color in the diagram.

3.4. Testing Environment

To test our customized Nutch crawler, we set up a small size Hadoop/Hbase cluster with 3 worker nodes and run it for 5, 10, 20, 30 and 40 iterations. We select Urdu as a test case language, and for seed, we collect 50 number of URLs from different Urdu domains. To avoid garbage collection, minimum threshold for language

Table 1: Configuration changes for testing environment with new language specific parameters

Type	Property	Value Options	Test value	Description
Default	db.ignore.internal.links	true, false	false	Enable/ disable internal links
Default	db.ignore.external.links	true, false	true	Enable/ disable external links
Default	generate.max.count	numeric	50	Maximum links from single domain in each iteration
New	filter.lang.enable	true, false	true	Enable/disable language filter
New	filter.lang.label	language label	Urdu	Language name to be filtered
New	filter.lang.minSize.enable	true, false	true	Enable/ disable minimum size filter
New	filter.lang.minSize.bytes	numeric (bytes)	1	Minimum language bytes, i.e., threshold
New	filter.lang.maxSize.enable	true, false	false	Enable/ disable maximum size filter
New	filter.lang.maxSize.bytes	numeric (bytes)	-	Maximum language bytes
New	filter.lang.minPercentage.enable	true, false	false	Enable/ disable language percentage filter
New	filter.lang.minPercentage.limit	Numeric (%)	-	Minimum language percentage

size filter is set to 256 bytes. In each iteration, the crawler selects a maximum 2,500 URLs (topN) from all domains, and from single domain, a maximum of 50 URLs are marked for politeness via *generate.max.count*. In addition, instead of manually checking crawl documents for accuracy measurement, we index all crawled documents in Apache Solr that is an open source full text search engine [22]. Relevant documents are retrieved using query filter present in Apache Solr. Important configuration parameters with their test-case values are given in Table 1.

4. Results

In this section, we present experimental results from language-specific crawler. First, we discuss yield rate statistics and later, we discuss the accuracy measures of our proposed crawler.

4.1. Crawler Yield Rate

Yield rate statistics help to know the crawling rate at different intervals. After the crawling job completion, a total of 25,723 documents are successfully fetched, out of which, 24,174 documents have content in Urdu language with a percentage more than threshold. Figure 3 presents yield plot in our experimentation of customized crawler for overall successfully crawled documents and Urdu language documents vs the number of iterations. In each iteration, the number of crawled documents is very close to the total number of documents fetched in that iteration. It shows better accuracy of crawler in context of Urdu documents. The crawling rate varies from 50 to 1300 in this analysis. The

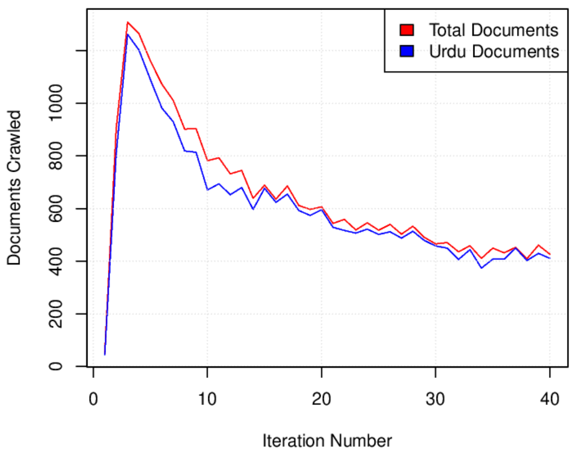


Figure 3: Crawler Yield Rate

decaying behavior in the figure shows that over time, available URL space, i.e., the list, is reducing as more and more URLs are crawled with time. It is due to the reason that we have disabled out-links as already discussed in Section 3.

4.2. Accuracy Measurements

In order to measure effectiveness of NCL-Crawl system, we use the percentage of relevant pages from total downloaded pages in each iteration as the accuracy

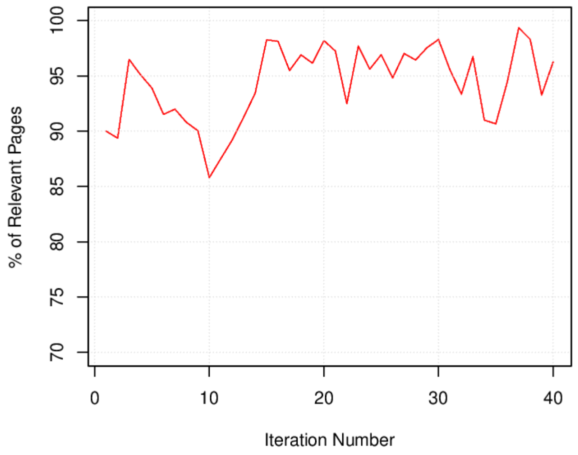


Figure 4: Accuracy Measure of Proposed Language

measurement. The relevant pages are those pages where Urdu language is found and where the percentage of Urdu content is greater than the configured threshold. Figure 4 shows the customized crawler accuracy measurement for each iteration, and it varies from 86% to 99% during the experimentation. Overall, the crawler accuracy is 93.99% which is a very good score.

In general, the system accuracy depends upon the seed collection and language threshold parameter. If the seed is refined to the given language and threshold is not very large, then the accuracy will be very high as observed in our current experimentation, and if seed is not very well refined in context of the given language and the threshold is set very high, then the accuracy will increase or decrease immoderately. As already discussed, this is a new feature added in the default Nutch crawler, hence we cannot compare our results with some existing feature in Nutch default version.

5. Conclusion

In this work, we endeavor to build a language specific web crawling system, i.e., NCL-Crawl, to assist the NLP community for textual corpus building and regional language web crawling at a large scale. For this purpose, we have customized the Apache Nutch fetcher class and added CLD2 language detection module to identify the language of crawled content at run time. For experimentation and evaluation of our work, we collect 50 seed URLs in Urdu language from different domains and run the crawler for 40 iterations. To avoid garbage collection, we set minimum size threshold to 256 bytes of Urdu. Total crawled documents are 25,174 that include 24,174 documents with Urdu language content more than threshold. The crawling rate varies from 50 to 1300 during the job execution. Overall accuracy is 93.99% and varies from 86% to 99%. In general, this accuracy is dependent on the given seed URLs and language threshold parameter and will vary with these two parameters. Lastly, this solution can be used for any language and threshold value; one has to just change configuration parameters only. In future, we plan to open source this system for research community.

6. Acknowledgement

This research work was funded by Higher Education Commission (HEC) Pakistan and Ministry of Planning Development and Reforms under National Center in Big Data and Cloud Computing.

7. References

[1] Search engine market share china — statcounter global stats. https://gs.statcounter.com/search-engine-marketshare/all/china/monthly-201808-201908, 2019.
[2] Search engine market share russian-federation — statcounter global stats. https://gs.statcounter.com/searchengine-market-share/all/russian-federation/monthly-201808-201908, 2019.
[3] Focused web crawler - wikipedia. https://en.wikipedia.org/wiki/Focused crawler, 2019.
[4] Natural Language Processing - Wikipedia. https://en.wikipedia.org/wiki/Natural language processing, 2019.
[5] Text Corpus - Wikipedia. https://en.wikipedia.org/wiki/Text corpus, 2019.
[6] Top 50 open source web crawlers for web mining. https://bigdata-madesimple.com/top-50-open-sourceweb-crawlers-for-data-mining/, 2019.
[7] WIKI Apache Nutch Web Crawler. https://wiki.apache.org/nutch/NutchTutorial, 2019.
[8] Compact Language Detector 2. CLD2owners/cld2. https://github.com/CLD2Owners/cld2, 2019.
[9] Joy Dewanjee. Heuristic approach for designing a focused web crawler using cuckoo search. International Journal of Computer Science and Engineering, 4(09):59–63, 2016.
[10] William Eka Putra and Saiful Akbar. Focused crawling using dictionary algorithm with breadth first and by page length methods for javanese and sundanese corpus construction. International Journal of Procedia Technology, 11:870–876, 2013.
[11] Yajun Du, Wenjun Liu, Xianjing Lv, and Guoli Peng. An improved focused crawler based on semantic similarity vector space model. Applied Soft Computing, 36:392–407, 2015.
[12] Ayar Pranav and Sandip Chauhan. Efficient focused web crawling approach for search engine. International Journal of Computer Science and Mobile Computing, 4(5), 2015.

[13] Soumen Chakrabarti, Martin Van den Berg, and Byron Dom. Focused crawling: a new approach to topic specific web resource discovery. International Journal of Computer networks, 31(11-16):1623–1640, 1999.
[14] Takayuki Tamura, Kulwadee Somboonviwat, and Masaru Kitsuregawa. A method for language-specific web crawling and its evaluation. International Journal of Systems and Computers in Japan, 38(2):10–20, 2007.
[15] Kulwadee Somboonviwat, Masaru Kitsuregawa, and Takayuki Tamura. Simulation study of language specific web crawling. In Proceedings of International Conference on Data Engineering Workshops (ICDEW'05), pages 1254–1254. IEEE, 2005.
[16] Apache nutch web crawler. http://https://nutch.apache.org/, 2019.
[17] Luis A Lopez, Ruth Duerr, and Siri Jodha Singh Khalsa. Optimizing apache nutch for domain specific crawling at large scale. In Proceedings of International Conference on Big Data (Big Data), pages 1967–1971. IEEE, 2015.
[18] Digitalpebble's blog: Nutch fight! 1.7 vs 2.2.1. https://gs.statcounter.com/search-engine-market-share, 2019.
[19] saffsd. Python's standalone language identification tool. https://github.com/saffsd/langid.py, 2017.
[20] Michal Danilak. langdetect: language-detection library to python. https://github.com/Mimino666/langdetect, 2017.
[21] Boilerpipe. https://code.google.com/archive/p/boilerpipe, 2019.
[22] Apache solr. https://lucene.apache.org/solr/, 2019.

Sentiment and Emotion Analysis of Text: A Survey on Approaches and Resources

Nazish Azam, Bilal Tahir, Muhammad Amir Mehmood
Al-Khawarizmi Institute of Computer Science, UET, Lahore, Pakistan
{nazish.azam, bilal.tahir, amir.mehmood}@kics.edu.pk

Abstract

The evolution of internet has given the ability to the users to give their reviews, ratings, and opinions on social media or commercial websites. Sentiment and emotion analysis is an ongoing field of research in text processing, which aims to classify these reviews automatically. This paper presents the survey regarding approaches and resources used for sentiment and emotion analysis of text. We summarize the techniques, datasets, and resources available for text analysis. Additionally, we focus on summarizing literature and resources available for Urdu, a low resource language, along with some open problems for Urdu text analysis. The presented survey can be used effectively to understand challenges and to take future direction for research in sentiment and emotion analysis field, especially for Urdu.

Keywords– Sentiment analysis, Emotion analysis, Lexicon, Urdu natural language processing

1. Introduction

In recent years, technology has been so much enhanced that internet is now an irreplaceable part of our lives. According to Human Computer Interaction (HCI) studies, people are now so addicted and connected to computers and busy in using internet. The accelerated evolution of internet has attracted people from all over the world to social media platforms, micro-blogging websites, and online discussion forums. The sentimental content in form of reviews, opinions, recommendations, ratings, and feedback is generated by users on these platforms. Analysis of these sentiments has spread across many fields such as consumer information, marketing, and social analysis. Sentiment analysis is performed to enhance the quality of products or to understand the public opinion towards different topics [1], [2], [3].

In the field of sentiment analysis, subjectivity analysis of text is done to determine the attitude or polarity of the writer. Such analysis helps in decision making and it is an important human aspect because it tells us “What other people think”. In general, the polarity or sentiment of text is classified into three main classes, i.e., positive, negative, and neutral. Similar to sentiment, emotions can be analyzed computationally. Despite the fact that sentiments and emotions are synonyms and equivalent words, but they don’t express something very similar. Looking into the dictionary shows that sentiment is just an opinion or view while emotion alludes to feeling according to the mood [4]. However, the goal of emotion analysis is a difficult task as differences between emotions are subtler than those between positive and negative class. Additionally, the emotion itself is a universal feeling however different people concerning social context, values, interests, and experience have a different interpretation of text [5].

The approaches for sentiment and emotion analysis for text are categorized into three classes: 1) Lexicon Based, 2) Machine Learning, 3) Hybrid approach. Lexicon based approach classifies textual content utilizing a list of manually labelled words. Machine learning methods use machine learning algorithms along with textual features of content for classification of text. Hybrid method combines lexicon and machine learning approaches to enhance the performance of classifiers. However, the performance of the approach is highly dependent on data quality, size, and content language.

The sentiment and emotion analysis are extensively applied to understand social, political and business behaviours. The sentiment analysis of reviews is done in [1], [2], [6], and [7] to automatically rate the product using user opinion. Similarly, tweets are analyzed to understand the political biases of news channels [8] and to calculate sentiment towards Syrian refugees [9]. The emotion analysis of tweets is done in [10] to predict the outcome of US election 2016 by analyzing public perception towards candidates. Previous studies show that much work has been done on sentiment and emotion analysis for English text. A large research gap is still present in case of Urdu - a resource poor language.

In this paper, we aim to do a survey on sentiment and emotion analysis research efforts, datasets, lexical resources and classification techniques. Besides discussing approaches used for English language, we focus on approaches and challenges for sentiment and emotion analysis of Urdu and Roman Urdu text as well. Additionally, a discussion of available resources and datasets for the Urdu language is provided to facilitate future work.

The rest of the paper is organized as follows: We discuss available lexicons, datasets, and existing approaches for sentiment analysis in Section 2. Next, we present our survey on emotion analysis in Section 3. Before the conclusion, we discuss some open problems related to Urdu text analysis in Section 4. Finally, we conclude our paper in Section 5.

2. Sentiment Analysis

In this section, first, we provide details of publicly available lexicons and datasets to perform sentiment analysis. Next, we discuss different approaches used to perform sentiment analysis.

2.1. Datasets and Lexicons

Table 1 provides a summary of a few online available sentiment analysis lexicons and datasets. AFFIN [11] lexicon consists of 3300+ English words labelled from -5 to +5 scale with an integer variance. Similarly, SentiWordNet [12], and Sentiment lexicon [13] provide English words with sentiment score. However, the Urdu language lacks in lexical resources. An Urdu sentiment lexicon1[3] provides sentiment labels of Urdu words by translating English language lexicon [13] into Urdu using a dictionary lookup. Additionally, all synonyms of translated words are also included in the lexicon. The lexicon consists of 2,607 positive and 4,728 negative sentiment words.

The number of datasets from microblogs, blogs, user comments and review sites are constructed because these platforms provide a good understanding of public opinions. The IMDB review dataset [14] provides 50k movie reviews with even split of 25,000 reviews from positive and negative class. Similarly, Twitter US airline sentiment[4] labelled 14.5k tweets related to six US airlines into three classes of positive, negative, and neutral. Multi-language sentiment analysis[5] provides labelled public opinion from chat logs of WhatsApp, Messenger, and SMS data in English, Mandarin, and Malay language. Additionally, the efforts are made to understand public opinion in Urdu by building Roman Urdu[6] and Urdu language sentiment [15] analysis dataset. Roman Urdu dataset consists of 20,000 roman Urdu sentences labelled into three classes. Urdu language sentiment dataset consists of 999 Urdu language political tweets manually labelled by three judges.

2.2. Approaches

For sentiment detection in textual data, various methods are introduced in the literature. Table 2 provides the summary of research done for sentiment analysis. The work is distributed into three sections of English, Roman Urdu, and Urdu according to language focused in work. The methods used for sentiment analysis of these languages can be categorized into three classes 1) Lexicon-based Method, 2) Machine learning Method, and 3) Hybrid Method.

2.2.1. Lexicon Based Method: Lexicon based recognition approach classifies textual content utilizing a list of labelled positive, negative, and neutral words. In study [9], lexicon is developed for Turkish language. Additionally, sentiment analysis of English and Turkish tweets related to Syrian refugees is performed. The classification of tweets into 5 categories of very negative, negative, neutral, positive, and very positive shows the positive sentiment of Turkish tweets compared to English.

In [18], the Roman Urdu Opinion Mining System (RUOMIS) is built for analysis of comments on mobile review website. The lexicon is built by labelling adjectives in content. The results show 100% recall but the precision value is 27.1% only. The reason for poor precision is noise and failure of POS tagger in identifying adjectives correctly due to the unstructured nature of Roman text.

Study in [20] use English translated Urdu lexicon [15] for analysis of 124 Urdu comments. The experiment shows the accuracy of 66% in sentiment classification. Similarly, study in [21] use Urdu content from blogs to for sentiment analysis. Lexicon is built by identifying and labelling nouns and adjectives using POS tagger. The accuracy of 66% in classification of text shows good results. However, remaining parts of POS-tagged text need to be analyzed and included in lexicon to improve the accuracy. The research is done for identification and labelling of words with sentiment (SentiUnits) in [22]. The authors use POS tagger with grammatical and semantic rules to identify and label SentiUnits in Urdu text. The labelled lexicon is used for sentiment analysis of Urdu corpus containing reviews about movies and products. The results show 72% and 78% accuracy for movies and reviews, respectively. The similar study is done in [1] by using SentiUnits for sentiment classification of Urdu text. The study also shows the improvement in results by applying negation handling during sentiment classification of Urdu text.

The authors use Urdu tweets to determine the political biases of Pakistani news channels in [8]. They

[3]http://chaoticity.com/urdusentimentlexicon/
[4]https://www.kaggle.com/crowdflower/twitter-airline sentiment
[5]https://www.kaggle.com/weywenn/sentiment-analysis-multilanguage
[6]https://archive.ics.uci.edu/ml/datasets/Roman+Urdu+Data+Set

build an Urdu language sentiment lexicon by labelling nouns and adjectives in Urdu tweets. In addition to sentiment analysis, the aspect analysis of sentiment is done to determine the biases of three news channels towards the Pakistani Government. The paper [23] compares the performance of three machine learning algorithms i.e., Support Vector Machine (SVM), Decision Tree, K-Nearest Neighbor (KNN), and lexicon approaches for sentiment classification of Urdu text. The results show that the lexicon approach improves accuracy from 73.88% to 89.03% compared to machine learning algorithms. In addition, the lexicon approach performs better in terms of precision, recall, time-cost. The reason for better performance of lexical approach is that wide coverage of lexicon and an efficient Urdu Sentiment Analyzer is developed that can efficiently handle data from multiple domains.

2.2.2. Machine Learning Method: The study in [16] uses Multinomial Naive Bayes (MNB) model with n-gram and POS tags as features for classification of English tweets into positive, negative, and neutral class. The results describe the best performance of model with bi-gram features. Additionally, the evaluation of model on different size datasets shows the improvement in accuracy of classification on large dataset. However,

Table 1: Summary of publicly available lexicons & datasets for sentiment analysis

	Name	Data Size	Language	Classes
Lexicon	AFINN lexicon [11]	3,300+ words	English	Integer between -5 (negative) and +5 (positive)
	SentiWordNet [12]	-	English	Positive, Negative, and Objectivity
	Sentiment Lexicon [13]	6,800 words	English	Positive and Negative.
	Urdu Sentiment Lexicon[1]	7,335 words	Urdu	Positive and Negative.
Datasets	IMDB reviews [14]	50,000 reviews	English	Positive and Negative
	Twitter US Airline Sentiment [2]	14,500 Tweets	English	Positive, Negative, and Neutral
	Sentiment Analysis Multi-Language[3]	1,531 samples	Multi Language	Very Satisfied, Satisfied, Neutral, Unsatisfied, Very Unsatisfied.
	Roman Urdu Dataset[4]	20,000 records	Roman Urdu	Positive, Negative, and Neutral
	Urdu-Sentiment- Dataset [15]	999 Tweets	Urdu	Positive, Negative, and Objective

Table 2: Summary of sentiment analysis research

Language	Author	Methodology	Data
English	Öztürk and Ayvaz [9]	Lexicon based	1,353,367 English & 1,027,930 Turkish Tweets
	Pak and Paroubek [16]	Machine learning	300,000 Tweets
	Shoeb and Ahmed [17]	Machine learning	489 Tweets
	Mukwazvure and Supreethi [3]	Hybrid Method	333,686 News Comments
	Govindarajan [6]	Hybrid Method	2,000 movie reviews
Roman Urdu	Daud et al. [18]	Lexicon based	1,620 Roman Urdu comments
	Arif et al. [7]	Machine learning	Roman 1,600 Urdu/Hindi hotel reviews
	Noor et al. [2]	Machine learning	20,286 Roman Urdu reviews from Ecommerce site
	Ghulam et al. [19]	Machine learning	Roman Urdu text
Urdu	Rehman and Bajwa [20]	Lexicon Based	Urdu news
	Hashim and Khan [21]	Lexicon Based	Public opinion on news headlines
	Syed et al. [22]	Lexicon Based	1,000 Reviews on Urdu Websites
	Syed et al. [1]	Lexicon Based	Urdu corpus of movie reviews
	Amjad et al. [8]	Lexicon Based	26,614 Urdu news Tweets
	Mukhtar et al. [23]	Lexicon Based	Urdu text from blogs

when the dataset is large enough, the improvement cannot be achieved by only increasing the size of the training data. Similarly, in [17], a study is done to classify English tweets using K Nearest Neighbor (KNN), Naive Bayes (NB), and Decision Tree (DT) classifier. The results describe the Decision Tree as outperforming classifier with 84.66% accuracy and 95.96% precision.

The sentiment analysis of hotel reviews in Roman Urdu text is done in [7]. The corpus is built by translating English text into Roman Urdu. The translation of English text is done with one translation tool to avoid irregularities in spelling. The SVM classifier shows significant performance with 95% accuracy with Term Frequency (TF)-Inverse Document Frequency (IDF) features. However, the performance is with no spelling inconsistencies which is one of the challenging issues in the analysis of Roman Urdu text. Similarly, [2] uses SVM model with Bag of Word (BoW) features to classify Roman Urdu reviews from an e-commerce site into positive, negative, and neutral classes. In [19] the comparison of baseline machine learning models (NB, Random Forest (RF), and SVM) and deep learning model Long Short Term Memory (LSTM) is done for sentiment classification of Roman Urdu text. The comparison shows the better performance of LSTM model with word embedding.

2.2.3. Hybrid Method: The study in [3] combines lexicon and machine learning approach to classify English news comments from technology, business, and political domain. The lexicon is used for polarity detection of text. The output of lexicon is used to train SVM and KNN models. The results describe the negative impact of small training data size and neutral class on the performance of classifiers. In [6] NB and Genetic Algorithms are combined as an ensemble technique for analysis of English documents. Their comparative experiments show the effectiveness of hybrid technique for sentiment classification. The comparison of both models with hybrid approach shows the hybrid approach as the best performing model with 93% accuracy.

Table 3: Summary of emotion models

Model	Proposed Emotions	Approach	Structure
Ekman [24]	Anger, disgust, fear, joy, sadness, surprise	Categorical	-
Shaver et al. [25]	Anger, fear, joy, love, sadness, surprise	Categorical	Tree
Oatley and Johnson-Laird [26]	Anger, anxiety, disgust, happiness, sadness	Categorical	-
Plutchik [27]	Acceptance, admiration, aggressiveness, amazement, anger, annoyance, anticipation, apprehension, awe, boredom, contempt, disapproval, disgust, distraction, ecstasy, fear, grief, interest, joy, loathing, love, optimism, pensiveness, rage, remorse, sadness, serenity, submission, surprise, terror, trust, vigilance	Dimensional	Wheel
Circumplex Russell [28]	Afraid, alarmed, angry, annoyed, aroused, astonished, at ease, bored, calm, content, delighted, depressed, distressed, droopy, excited, frustrated, glad, gloomy, happy, miserable, pleased, relaxed, sad, satisfied, serene, sleepy, tense, tired	Dimensional	Valence, Arousal
OCC Ortony et al. [29]	Admiration, anger, appreciation, disappointment, disliking, fear, fears confirmed, gloating, gratification, gratitude, happy-for, hope, liking, pity, pride, sorry-for, relief, remorse, reproach, resentment, self-reproach, shame	Dimensional	Tree
Lovheim [30]	Anger/rage, contempt/disgust, distress/anguish, enjoyment/ joy, fear/terror, interest/excitement, shame/humiliation, surprise/startle	Dimensional	Cube

3. Emotion Analysis

In this section, first, we discuss existing emotion models for categorization of emotions. Next, we present available datasets and lexicons for analysis. Finally, we discuss the approaches which exist in the literature for emotion classification.

2.1. Emotion Models

To detect and analyze the emotions, they are categorized with standard emotion models. Table 3 summarizes the existing emotion models. According to psychology, there are 6 types of basic emotions expressed by human beings [37]. These emotions are 1) Happiness, 2) Sadness, 3) Fear, 4) Disgust, 5) Anger, and 6) Surprise. Variety of emotion models are presented to further classify these basic emotions. Ekman [24] presents six discrete emotions as mentioned in Table 3. Shaver et al. [25] and Oatley and Johnson-Laird [26] define 6 categories of emotions including love, anxiety and happiness. Plutchik [27] and Circumplex Russell [28] provides two dimensional emotion categorization model. Additionally, OCC Ortony et al. [29] and Lovheim [30] distribute emotions in three dimensions to create categories.

2.2. Datasets and Lexicons

Table 4 provides summary of available labelled emotion datasets and lexicons. EmoSenticNet [31] is online available emotion lexicon containing 13,171 words categorized into joy, sadness, disgust, anger, surprise, and fear. Similarly, National Research Council (NRC) Word-Emotion Association Lexicon (EmoLex) [32] categorizes 14,000+ words into eight classes of emotion. Additionally, NRC dataset is translated into 40+ languages including Urdu. Our manual inspection of Urdu translated lexicon reveals that few English terms are not translated to Urdu. Additionally, we note that ATM translates multiple English terms to one Urdu word, i.e., accused, accuser, and accusing are translated to one word "*Alzaam laga*'. In English, these terms are provided different labels, however, translated to one Urdu term creates ambiguity of assigned label. Therefore, manual cleaning of the lexicon is required to use it for Urdu text.

EmoBank [33] dataset consists of 10k English sentences labelled with six Ekman emotions using Valence-Arousal-Dominance scheme. Similarly, International Survey on Emotion Antecedents and Reactions (ISEAR) [34] dataset provides 7,666 labelled English sentences with seven emotions of anger, disgust, fear, sadness, shame, joy, and guilt. Furthermore, the emotion in text dataset5[7] consists of manually label 40k tweets into 13 classes of emotion. Additionally, 2,892 Facebook posts are categorized into six classes using Valence-Arousal-Dominance scheme [35]. Similarly, Affective text [36] classifies 1,200 news headlines into Ekman model categories. Up to our knowledge, there is no publicly available Urdu language emotion labelled dataset. This shows the scarcity of resource and huge research gap in emotion analysis of Urdu content.

Table 4: Summary of publicly available lexicons & datasets for emotion analysis

	Name	**Data Size**	**Language**	**Classes**
Lexicon	EmoSenticNet [31]	13,171 words	English	Joy, Sadness, Disgust, Anger, Surprise, Fear
	NRC-EmoLex [32]	14,000+ words	40+ languages including Urdu	Anger, Fear, Anticipation, Trust, Surprise, Sadness, Joy, and Disgust.
Datasets	EmoBank [33]	10,000 sentences	English	Double annotation with valence, arousal and dominance
	ISEAR [34]	7,666 sentences	English	Joy, Fear, Anger, Sadness, Disgust, Shame, and Guilt
	Emotion in Text[5]	40,000 Tweets	English	Anger, Boredom, Empty, Enthusiasm, Fun, Happiness, Hate, Love, Relief, Sadness, Surprise, Worry, Neutral
	The valence and arousal Facebook posts [35]	2,895 Facebook posts	English	Double annotation with valence and arousal values
	Affective Text [36]	1,200 News Headlines	English	Annotated with 6 basic emotions from Ekman's model

2.3. Approaches

The brief overview of papers on emotion analysis of text is given in Table 5. Keyword based method is used in [39] for classification of English email content into three categories of happy, sad, and angry. Besides keyword spotting, semantic emotions are calculated using a semantic network method for classification of text. The proposed methodology is limited to certain emotion-related texts, e.g., couple's breakup. Therefore, emotion detection from technical writings or scientific

[7] https://www.kaggle.com/c/sa-emotions

papers is not possible because these texts simply do not contain emotions.

Authors in [10] use lexicon based approach for emotion analysis of 25 million English tweets related to Donald Trump and Hillary Clinton. Based on emotion analysis using NRC lexicon, they predicted the outcome of US state election 2016. The proposed approach has extensive applications as it is not only limited to the political domain.

The study in [40] uses machine learning models of SVM and KNN for emotion classification of tweets according to Circumplex Model of Affect. In addition, the authors manually label hashtags to use it as an emotion label of tweet. The comparison of manually labelling of complete tweet and automatic labelling of a tweet using hashtags shows that hashtags can be used as an emotion label of a tweet with 87% accuracy. Additionally, in [41] online health communities (OHCs) comments regarding cancer are classified using a combination of lexicon approach and deep learning models of CNN and LSTM. The analysis shows that hybrid model performance improves because it captures the hidden semantics in OHCs messages. Similarly, Lexicon-based, Keyword based, and machine learning-based emotion classification methods are combined for classification of 15 emotions in suicide notes [42]. The combination of Affective text lexicon with machine learning models of SVM, KNN, and maximum entropy shows that hybrid techniques exhibit a high robust discriminative capability in emotion classification especially when a large number of emotion instances are available.

The little amount of literature is available in emotion analysis of Roman Urdu content. The study in [43] presents knowledge based approach to label Roman Urdu text by developing an emotion ontology of happiness, anger, disgust, surprise, and fear emotions. The approach analyses syntax and the semantic relationship of text to detect the emotion. The testing of the proposed approach on manually labelled data shows the average recall and precision of 85.40% and 92.87%, respectively. The study presented in [44] made an effort to detect joy, fear, anger, sadness, disgust, and shame from Urdu language tweets about smartphones and sports products. The SVM, RF, NB and KNN models are tested on 1,000 sports and 1,200 smartphone tweets.

4. Open Problems in Urdu Text Analysis

The core approaches (lexicon based, machine learning, and hybrid method), as mentioned in Section 2.2, used to perform text analysis of English can also be used for Urdu but research and development effort is required because of vast differences between English & Urdu grammar, orthography, and morphology. For Urdu language text analysis, first of all, we need a dataset and there are rare datasets & corpora available for sentiment as well as emotion analysis of Urdu text. To use the lexicon based approach either for sentiment or emotion analysis, we need annotated lexicon of Urdu. Few Urdu lexicons have been created so far and most of them are not publicly available while in the case of English, we have a variety of lexicons available. Due to different sentence structure, i.e., Subject-Object-Verb (SOV) and position of preposition, as compared to English, we face difficulty in Urdu text classification [45]. Due to complex morphology and unstructured format of Urdu, English morphological analyzers and POS taggers cannot be exactly used [46].

Table 5: Summary of emotion analysis research

Language	Author	Methodology	Data	
English	Sailunaz [38]	Survey	-	-
	Ling et al. [39]	Keyword based	Emails	Happy, Sad, Angry
	Srinivasan et al. [10]	Lexicon based	English Tweets	Plutchik's Wheel of Emotions [27]
	Hasan et al. [40]	Machine learning	English Tweets	Circumplex Model of Affect
	Khanpour [41]	Hybrid method	Online health Communities' Posts	Anger, Disgust, Fear, Joy, Sadness, Surprise
	Yang et al. [42]	Hybrid method	English Suicide Notes	15 emotion categories
Roman Urdu	Nargis and Jamil [43]	Knowledge based	Roman Urdu Text or Blogs	Happiness, Anger, Hurt, Caring, Fear
Urdu	Rehman and Bajwa [20]	Machine learning	Smartphone and Sports Tweets	غصہ, خوف, خوشی ندامت, نفرت, اداس

5. Conclusion

In this paper, we perform a survey on techniques used for sentiment and emotion analysis of text. We explore the literature related to analysis of Urdu text deeply to understand the challenges in analyzing low resource language. We observe that field of sentiment analysis for Urdu is growing despite lack of resources. Additionally, emotion analysis task has more scarcity of literature and resources for Urdu text. Building resources, used in sentiment and emotion analysis tasks, is still needed for the Urdu language. Furthermore, due to the complex nature and challenges of Urdu text, a lot of work is required to understand Urdu text context.

In the future, we will deploy and compare the performance of existing models and resources for sentiment and emotion analysis for Urdu text to provide the benchmark for future research.

6. Acknowledgement

This research work was funded by Higher Education Commission (HEC) Pakistan and Ministry of Planning Development and Reforms under National Center in Big Data and Cloud Computing.

7. References

[1] A. Z. Syed, M. Aslam, and A. M. Martinez-Enriquez, “Sentiment analysis of urdu language: handling phraselevel negation,” in *Proceedings of* Mexican International Conference on Artificial Intelligence. Springer, 2011

[2] F. Noor, M. Bakhtyar, and J. Baber, “Sentiment analysis in e-commerce using svm on roman urdu text,” in *Proceedings of icetic* Springer, 2019, pp. 213–222.

[3] A. Mukwazvure and K. P. Supreethi, “A hybrid approach to sentiment analysis of news comments,” in Proceedings of ICRITO (Trends and Future Directions), Sep. 2015, pp. 1–6.

[4] N. Allouch, “Sentiment and emotion analysis: The absolute difference” May 2018.[online]. Available: http://blog.emojics.com/emotional-analysis-vs-sentiment-analysis/

[5] A. Balahur and A. Montoyo, “Applying a culture dependent emotion triggers database for text valence and emotion classification,” Journal of Procesamiento dellenguaje natural, no. 40, pp. 107–114, 2008.

[6] M. Govindarajan, “Sentiment analysis of movie reviews using hybrid method of naive bayes and genetic algorithm,” International Journal of Advanced Computer Research, vol. 3, no. 4, p. 139, 2013.

[7] H. Arif, K. Munir, A. S. Danyal, A. Salman, and M. M. Fraz, “Sentiment analysis of roman urdu/hindi using supervised methods,” in Proceedings of ICICC, 2016.

[8] K. Amjad, M. Ishtiaq, S. Firdous, and M. A. Mehmood, “Exploring twitter news biases using urdu-based sentiment lexicon,” in Proceedings of ICOSST. IEEE, 2017, pp. 48–53.

[9] N. Öztürk and S. Ayvaz, “Sentiment analysis on twit- ¨ ter: A text mining approach to the syrian refugee crisis,” Journal of Telematics and Informatics, vol. 35, no. 1, 2018.

[10] S. M. Srinivasan, R. S. Sangwan, C. J. Neill, and T. Zu, “Twitter data for predicting election results: Insights from emotion classification,” IEEE Technology and Society Magazine, vol. 38, no. 1, pp. 58–63, 2019.

[11] F. A. Nielsen, “A new ANEW: Evaluation of a word list ˚ for sentiment analysis in microblogs,” arXiv preprint arXiv:1103.2903, 2011.

[12] S. Baccianella, A. Esuli, and F. Sebastiani, “Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining.” in **Proceedings of LREc**, vol. 10, no. 2010, 2010, pp. 2200–2204.

[13] M. Hu and B. Liu, “Mining and summarizing customer reviews,” in Proceedings of SIGKDD, ACM, 2004

[14] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in Proceedings of the 49th annual meeting of the ACLWEB: Human language technologies,” *ACLWEB*, 2011

[15] M. Y. Khan, S. M. Emaduddin, and K. N. Junejo, “Harnessing english sentiment lexicons for polarity detection in urdu tweets: A baseline approach,” in Proceedings of ICSC, IEEE, 2017, pp. 242–249.

[16] A. Pak and P. Paroubek, “Twitter as a corpus for sentiment analysis and opinion mining.” in Proceedings of *LREc*, vol. 10, 2010, pp. 1320–1326.

[17] M. Shoeb and J. Ahmed, “Sentiment analysis and classification of tweets using data mining,” *International Research Journal of Engineering and Technology*, 2017.

[18] M. Daud, R. Khan, A. Daud *et al.*, “Roman urdu opinion mining system (ruomis),” *arXiv preprint arXiv:1501.01386*, 2015.

[19] H. Ghulam, F. Zeng, W. Li, and Y. Xiao, “Deep learning based sentiment analysis for roman urdu text,” *Procedia computer science*, vol. 147, pp. 131–135, 2019.

[20] Z. U. Rehman and I. S. Bajwa, “Lexicon-based sentiment analysis for urdu language,” in *Proceedings of INTECH*. IEEE, 2016, pp. 497–501.

[21] F. Hashim and M. Khan, “Sentence level sentiment analysis using urdu nouns,” *Department of Computer Science, University of Peshawar, Pakistan*, 2016.

[22] A. Z. Syed, M. Aslam, and A. M. Martinez-Enriquez, “Lexicon based sentiment analysis of urdu text using sentiunits,” in *Proceedings of micai*. Springer, 2010

[23] N. Mukhtar, M. A. Khan, and N. Chiragh, “Lexiconbased approach outperforms supervised machine learning approach for urdu sentiment analysis in multiple domains,” *Telematics and Informatics*, vol. 35, no. 8, 2018.

[24] P. Ekman, “An argument for basic emotions,” *Cognition and Emotion*, vol. 6, no.3-4, pp. 169–200, 1992.

[25] P. Shaver, J. Schwartz, D. Kirson, and C. O’connor, “Emotion knowledge: further exploration of a prototype approach.” *Journal of personality and social psychology*, vol. 52, no. 6, p. 1061, 1987.

[26] K. Oatley and P. N. Johnson-laird, “Towards a cognitive theory of emotions,” *Cognition and Emotion*, vol. 1, 1987

[27] R. Plutchik, “A psychoevolutionary theory of emotions,” *Social Science Information*, vol. 21, no. 4-5, 1982

[28] J. Russell, “A circumplex model of affect,” *Journal of Personality and Social Psychology*, vol. 39, 12 1980.

[29] A. Ortony, G. L. Clore, and A. Collins, “The cognitive structure of emotions,” *Cambridge Uni*, 1988.

[30] H. Lovheim, “A new three-dimensional model for emo- ¨ tions and monoamine neurotransmitters,” *Medical hypotheses*, vol. 78, no. 2, pp. 341–348, 2012.
[31] S. Poria, A. Gelbukh, A. Hussain, N. Howard, D. Das, and S. Bandyopadhyay, “Enhanced senticnet with affective labels for concept-based opinion mining,” *Journal of IEEE Intelligent Systems*, vol. 28, no. 2, pp. 31–38, 2013.
[32] S. M. Mohammad and P. D. Turney, “Crowdsourcing a word–emotion association lexicon,” *Journal of Computational Intelligence*, vol. 29, no. 3, 2013.
[33] S. Buechel and U. Hahn, “Emobank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis,” in Proceedings of the European Chapter of the ACLWEB.” Volume 2, 2017

[34] K. R. Scherer and H. G. Wallbott, “Evidence for universality and cultural variation of differential emotion response patterning.” Journal of personality and social psychology, vol. 66, no. 2, p. 310, 1994.
[35] D. Preot¸iuc-Pietro, H. A. Schwartz, G. Park, J. Eichstaedt, M. Kern, L. Ungar, and E. Shulman, “Modelling valence and arousal in facebook posts,” in Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, 2016, pp. 9–15.
[36] C. Strapparava and R. Mihalcea, “Semeval-2007 task 14: Affective text,” in Proceedings of SemEval, 2007
[37] K. Cherry, “The 6 types of basic emotions and their effect on human behavior,” Jun 2019, (visited on 23 September, 2019). [online]. Available: https://www.verywellmind.com/an-overview-of-the-types-ofemotions-4163976
[38] K. Sailunaz, M. Dhaliwal, J. Rokne, and R. Alhajj, “Emotion detection from text and speech: a survey,” Journal of Social Network Analysis and Mining, vol. 8, no. 1, 2018.
[39] H. S. Ling, R. Bali, and R. A. Salam, “Emotion detection using keywords spotting and semantic network,” in Proceedings of ICOCI. IEEE, 2006, pp. 1–5.
[40] M. Hasan, E. Agu, and E. Rundensteiner, “Using hashtags as labels for supervised learning of emotions in twitter messages,” in proceedings of SIGKDD, USA, 2014.
[41] H. Khanpour and C. Caragea, “Fine-grained emotion detection in health-related online posts,” in Proceedings of the *EMNLP* 2018
[42] H. Yang, A. Willis, A. De Roeck, and B. Nuseibeh, “A hybrid model for automatic emotion recognition in suicide notes,” Journal of Biomedical informatics insights, vol. 5, pp. BII–S8948, 2012.
[43] G. Z. Nargis and N. Jamil, “Generating an emotion ontology for roman urdu text.” International Journal of Computational Linguistics Research, vol. 7, 2016.
[44] L. Sana, K. Nasir, A. Urooj, Z. Ishaq, and I. A. Hameed, “Bers: Bussiness-related emotion recognition system in urdu language using machine learning,” in *proceedings of* BESC. IEEE, 2018, pp. 238–242.
[45] S. M. J. Rizvi, “Development of algorithms and computational grammar for urdu,” Ph.D. dissertation, Ph. D. thesis, Pakistan Institute of Engineering and Applied Sciences (PIEAS), 2007.
[46] S. A. Khan, W. Anwar, and U. I. Bajwa, “Challenges in developing a rule based urdu stemmer,” in Proceedings of WSSANLP, 2011

Understanding User Search Behavior of Humkinar Urdu Search Engine

Nazish Azam, Hafiz Muhammad Shafiq, Muhammad Amir Mehmood
Al-Khawarizmi Institute of Computer Science, UET, Lahore, Pakistan
University of Engineering and Technology, Lahore, Pakistan
{nazish.azam, hafiz.shafiq, amir.mehmood}@kics.edu.pk

Abstract

Search engines have become inevitable in the current digital information age. Different search engines such as Google, Bing, Yahoo, etc., provide access to the most relevant information present on the World Wide Web to users. These search engines not only require the infrastructure to crawl the World-Wide-Web regularly but also need a framework to gather user metadata to understand user search behavior for improving user experience. In addition, user metadata is required to perform business analytics and digital forensics. User information like IP address, location, type of device, response time, user website activity, etc., help us to know about user navigational pattern. In this paper, we present a user search behavior study of regional search engine called Humkinar Urdu Search Engine (USE) by integrating an open source web analytics application "Matomo". We collect metadata of Humkinar users for about 35 months. Summary reports generated by the tool show different analyses which can help to effectively monitor the search engine. Furthermore, we present subjective test results and feedback to highlight the preferences of USE users. The analysis and survey can be used to improve the overall performance of Humkinar Urdu search engine in terms of ranking and personalization.

1. Introduction

In recent years, search engines have turned into a significant source of multi-domain data. Our knowledge source has moved from books and papers to web, predominantly because of the way that search engines give a wide variety of relevant data in a couple of seconds [1]. About 98.8% Internet users utilize search engines to get required information [2]. There are many search engines available in different languages for public, e.g., Google [3], Yahoo [4], Bing [5], Baidu [6], DuckDuckGo [7], and many others [8]. "Baidu" is specifically designed for Chinese region, "Yandex" is a well-known search engine in Russia and similarly, there are many other search engines available in different languages.

38% of all Americans use a search engine, 31% read news online, and 30% peruse the Internet just for entertainment. During this online activity, users leave "digital footprints" with their internet service provider (ISP) or search engine, disclosing their interests [9]. Collection of user information is necessary as government agencies and parties in civil litigation regularly ask technology and communication companies to turn over user data. In Pakistan, out of around 205 million population, about 76% have mobile phone subscriptions, 37 million people are active social media users, and an estimated 22% of the population uses Internet [10]. Other than this, user information helps to improve user experience of website visitors.

Urdu Search Engine (USE) [11], named as "Humkinar", is a practical step to encourage research in Urdu and facilitates such community who prefers to search and get information in Urdu. On the basis of above discussion, we have used a monitoring tool to make USE better with respect to design, development, content, and ranking. USE team needs to know what their visitors are doing on site, where do they click, what content they read and which links they follow. To attract more people on USE, it is required to make it perform efficiently by giving as much minimum delay as possible.

For website performance improvement, user behavior analysis is an important factor. It shows the interests of the user, and its engagement can be increased by upgrading most visited sections. For this purpose, a large variety of solutions are available as products or services, e.g., Matomo, AWStats, Elogic, Google Analytics, and many more [12]. In most cases, one has to append a small snippet of JavaScript in web pages where user monitoring is required [13]. Also, user activity analysis on a website helps to check the security of a website indirectly. Another important fact to keep in mind is that no one can find out about what your clients need except the clients themselves. So why not ask them? Our aim is to improve the user experience of incoming visitors, that is why we are analyzing user activities and their interests regarding USE.

In this paper, we describe the design, integration, and usage of our user tracking framework. Our main objective includes collecting user tracking details for performance betterment, ranking, and personalization of USE. We use an open source web analytics tool known as "Matomo" (formerly Piwik) [14]. For user survey, we made a questionnaire and got the feedback from 87 users. Our key findings in this study, for last 35 months are mentioned below:

- 23,022 people visit USE and total viewed pages are 117,439.
- Total searched queries are 54,710 out of which 15,694 are single word queries and the most searched query is “Pakistan”.
- About 84.06% of visitors belong to Pakistan and 24.4% used GNU/Linux OS.
- Average page load time of USE is 1.6403s, average network latency is 0.5116s, and average server serve time is only 0.0064 seconds.
- From user survey, we found that 70.1% of users know how to type in Urdu.
- From design point of view, 23% of users gave us 8 points showing a positive impression.
- 59.8% of users said that the design and features of USE are easy to use for searching and reading Urdu content.

The remainder of the paper is organized as follows: Section 2 describes the related work. In Section 3, some tools are discussed which are applied to USE. Section 4 presents design and implementation. In Section 5, we discuss the results obtained from the tool. Next, we present a user survey of USE showing the feedback of users in Section 6. Finally, Section 7 summarizes the whole discussion.

2. Related Work

Famous search engine Google has developed a web analytics application named as Google Analytics. In article [15], a case study has been done using Google Analytics showing prominent features, literature review, real life application of the software and guidelines for the first time users of Google Analytics. Another article states that all search engines track user behavior and recent development shows that search engines try to integrate results from different collections into their results to guide their users for relevant results [16]. This is how users can be guided to quality content based on personalization functionality. In another paper [17], the authors have proposed a new ranking algorithm for user-oriented web page ranking. They did it by tracking the user’s time spent on web page and compare it with Google’s PageRank algorithm. The study made in [18], shows that the authors used AWStats and Google Trends to visualize the statistics comprising of number of unique visitors, page views, keywords, origin of search, and geographic trends.

Eye-tracking analysis of user behavior in WWW search engine has been done which investigates how user interacts with result pages, browsing pattern and views [19]. A quantitative study has been made to explore that how the behavior of the Google users can help web masters to improve their techniques to be in top results on Google [20]. Search engines capture users’ activities in the search log, which is stored at the search engine server. An interface is proposed and developed by [21] which acts as a layer between Google and the searcher. This framework captures users’ queries before redirecting them to Google.

For large volume of user data, an intelligent system is required to analyze the user behavior and show trend prediction. Discovery of user information allows web based organizations to predict user access pattern and helps in future developments [22]. A methodological framework was proposed in the study [23], which predicts purchase behavior of websites audiences. Instead of targeting individual user interests and activities, they profile websites audiences.

Web server logs provide information like traversal from one page to another, storing user IP address and all the related information. In [24], a study has been done in which authors have found different statistics such as most visited web-pages, user IPs with most visits, and type of errors users have to face, etc., using *WebLogExper*t tool. Similarly in [25], authors have used both web client data as well as web server logs to build an automated data mining and recommendation system for web usage via KNN classification method. User click stream data was obtained via web client and other information such as IP address, user name, server name, etc., were obtained from web server logs.

The analysis of user behavior also helps in building a better recommendation system for users while searching on website. For this purpose, [26] has proposed a new method through semantic enhancement by analyzing web access logs. The

Table 1: Comparison of Google Analytics and Matomo

Feature	**Google Analytics**	**Matomo**
Vendor	Google	Matomo
Edition	Single	Self/Cloud hosted
Installation	No	Easy to install
User interface	Easy	Easy
Link to website	Addition of tracking ID	Addition of JavaScript
Addition of plugin	Not allowed	Allowed
Number of users	Limited	Unlimited
Re-marketing integration	Google Ads	None
Data freshness	Not guaranteed	All time
Data	Limited	Unlimited

authors have built three models for this purpose, two of them are for domain knowledge of website and third one is an ontology based model. They have shown that their proposed method enhances the web-page recommendation system and performs better than the most advanced web mining methods, i.e., PLWAP-Mine. Furthermore, [27] has examined web-server logs to find the number of visitors and their behavior to enhance the usability of an educational website. For this analysis, the authors have used *logExpertLite* tool and found different statistics such as total hits, users, bandwidth usage, unique IPs, etc., for 5 days of the week. In this study, they have discussed how to increase the accessibility and usability of a website from these metrics.

3. Tools

There is a large variety of web monitoring tools available on the Internet like AWStats, eLogic, Google Analytics, ShinyStats, Webalizer, and many others. Here, first, we provide a brief description of Google Analytics and Matomo. Next, we discuss the rationale behind our choice of analytical platform for studying user behavior of USE.

3.1. Google Analytics

It is a service based solution which is provided by Google to track traffic of a website. Free version is perfect for small companies and provides multiple data collection options across websites. Enterprise version is required for integration with Google BigQuery, Salesforce, advanced analysis, and access to raw data. A maximum of 200 number of views per property can be utilized while enterprise solution gives limit to 400 numbers. In order to use it, one just needs a Google account and has to append a small JavaScript code provided by Google Analytics in the footer of web pages. Google Analytics Spreadsheet add-on is available to access and manipulate data using Google spreadsheet. Native re-marketing is done with Google Ads. Google Ads, AdSense, and Search Console are used for native data on-boarding [28].

3.2. Matomo

Matomo (formerly Piwik) is an open source web analytics platform which provides detailed insights about user activities and their engagement on a website. Real-time data updates can be received containing detailed view of visitors and their activities. It also provides row evolution feature which allows to compare current and past metric data for various reports. Page transitions can be seen through it which help to view what visitors did before, and after viewing a specific page. The dashboard of this platform is customizable and can be extended by adding a wide variety of widgets and plugins. Major advantage of this tool is that one has complete control over it as this can be installed on web server side. Using Matomo APIs, data accessibility is easy. Advance reports can be collected by adding manual queries in the database. Adding custom dimensions and settings is another feature provided by Matomo. It gives privacy protection by not sharing user data with advertising companies. It uses database for archival and storage. Data formatter is used to format the data in presentable format [14]. Many other features of this tool are discussed later in this paper.

3.3. Comparison

Table 1 provides a brief comparison between Matomo and Google Analytics. Although Google Analytics is easy to use and there is no need for any type of installation, but being a search engine website, USE should own the complete user data, privacy and web hosting. Also, there are bandwidth and user limitations while using Google Analytics services. Moreover, it is not allowed to customize available plugins. Due to such restrictions, we have to use Matomo that is an open-source solution and easily customizable

4. Design and Implementation

In this section, first, we briefly discuss USE, its major components, and features. After that, we provide brief description about hosting and dashboard customization of Matomo. Finally, in the end, we discuss integration of tool with USE along with data acquisition and rendering.

4.1. Urdu Search Engine

USE is an Urdu language search engine which can be accessed at *www.humkinar.com.pk.* USE is comprised of three major components: Cloud Infrastructure (CI), Information Retrieval (IR), and Search Management (SM). CI is responsible for incremental web crawling services, development, testing and deployment of the work. On the other hand, IR performs linguistic and textual analysis on raw content while SM deals with building of indexes for available documents and apply ranking algorithms to present meaningful results to the user. Figure 1 presents a workflow diagram for USE. It has a distributed crawler that crawls and indexes web documents continuously. Customized ranking algorithms are being used to display most relevant and trending results to the user. An adaptable web interface is developed to serve results according to the query of user. For indexing and search solutions, “Apache Solr” is used by USE. Primary source of information storage and retrieval is Apache Hadoop framework. USE has developed their own filters for checking language, age, size and profanity of the documents. It has its own developed summary module to present summarized result according to the query of the user. Another major achievement of USE is that it has given SMS facility to users so that they can get latest and updated news by using SMS facility through their smartphones.
To keep all the above mentioned functionalities safe and updated, there is a dire need to monitor all the activities on USE. Unique requirements of USE include self-

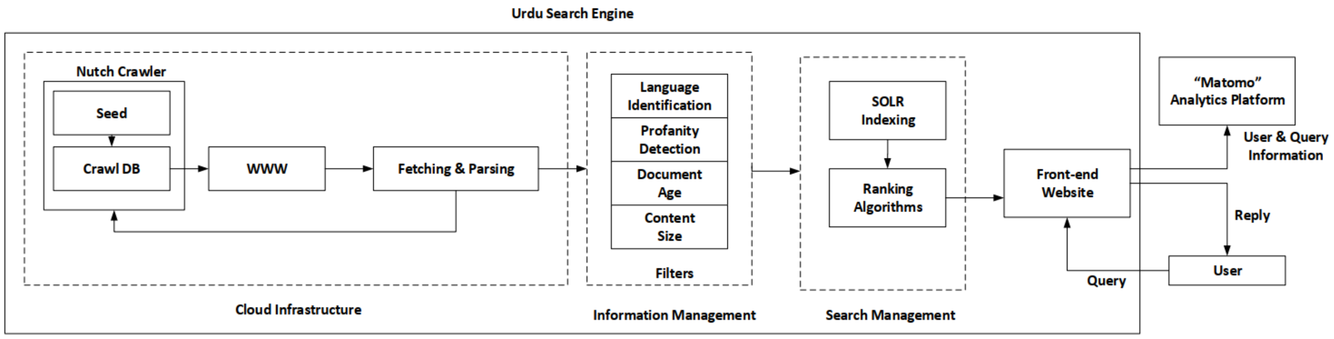


Figure 1: Architectural diagram of Humkinar USE

hosted tool so that it can have total control. Based on these requirements, a monitoring tool is designed for debugging, user behavior analysis, trends, ranking, personalization, and security checking. The next section briefly describes the design and implementation of the tool developed for USE.

4.2. Self-Hosting of Matomo

In our case, we use "self-hosted" approach to install Matomo on our web-server. Before its installation, it is required to make sure that you have a web server, shared hosting or dedicated server. If web server is not available then "Cloud Hosted" Matomo can be used for user analytics. By fulfilling all requirements, we successfully integrated version 3.7.0 of Matomo with

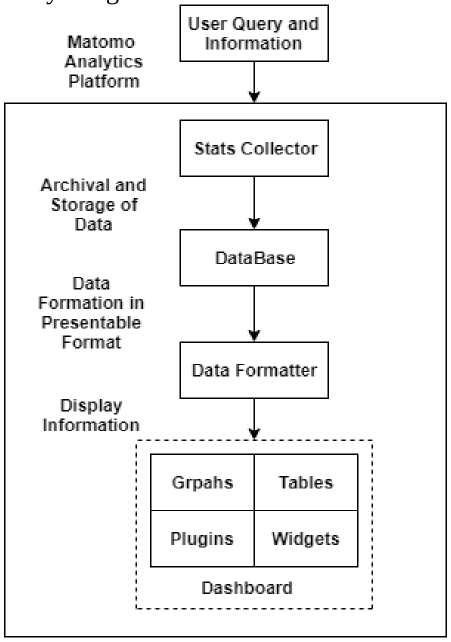

Matomo Analytics Platform
User Query and Information
Stats Collector
Archival and Storage of Data
DataBase
Data Formation in Presentable Format
Data Formatter
Display Information
Grpahs
Tables
Plugins
Widgets
Dashboard

Figure 2: Matomo structure

USE. It has a user-friendly graphical interface which is also customizable. We customized different plugins according to the requirements.

4.3. Dashboard Customization of Matomo

After providing login credentials, dashboard of Matomo can be accessed and there we have quick links to various sections of the analytics tools. The real-time section shows two subsections namely "real-time IP" and "searches". This is a custom plugin that shows only the summary of currently active IP addresses and searches made. Dashboard is the main analytics section of Matomo which can be customized according to the requirements. Different metrics can be used to track user behavior like evolution over the period, reports, device type, operating system, top searches, best performing pages, visitor logs, out-links etc. Default analytics features of Matomo are somehow limited in their usage. For example, default location provider of Matomo identifies the location of a user based on the language they use which is not very accurate. To tackle this problem, we added GeoIP2

Table 2: Yearly based analytics of Humkinar (October 24, 2016 - October 01, 2019

Attributes	**2016**	**2017**	**2018**	**2019**
Total visits	859	4,560	9,661	7,942
Unique visits	244	2,104	3,640	5,390
Total page views	6,948	22,267	71,896	16,328
Total search keywords	3,916	11,101	33,926	5,767
Bounce rate	23%	46%	42%	63%
Total outlinks	227	1,379	7,642	5,747

Table 3: General statistics

Attributes	**Values**
Total visits	23,022
Unique visits	11,378
Average page load Time	1.6403s
Average time spent by visitor	14 min 21s
Total page views	117,439
Total searches	54,710
Total outlinks	14,995

(PHP version) which uses GeoIP2 database and MaxMind's PHP API to find accurate location of the user. Another custom analytics feature was added in Matomo which helps us to record the document position. This position is then used for ranking of search results in Humkinar. Similarly, instead of using default reports, we have used custom reporting APIs, not limited in usage, to get our desired information in JSON or other formats.

4.4. Integration and Data Acquisition

After installation of Matomo on USE platform, a script is provided by Matomo that we append at the footer of those web pages that should be monitored. It logs all activities being carried out on the frontend and sends to back-end monitoring server. For USE, it includes information such as entered queries, click events, number of new and recurring users, IP, browsers information etc. Figure 2 shows a high level view of work-flow diagram for user monitoring at USE. Client enters a query on search engine and information about user and his query is stored in Matomo stats collector. This data is then sent to database for archival and storage. Data formatter converts the received data into presentable format and passes it to web dashboard. User is not disturbed at all in the whole process and he sees only search results on frontend of USE as a reply. Furthermore, in this study, we have analyzed data of October 24, 2016 to October 01, 2019.

5. Results

In this section, we present our findings for user behavior monitoring on USE with Matomo. First section describes yearly based statistics of Humkinar. Then we discuss other metrics like visitor browser, device type, event logs etc. After that, we discuss about the metrics that are very important for search engine websites such as searched keywords, clicks, user Geo-location, and website performance for different sections etc. Table 3 shows general statistics of USE.

5.1. Yearly Based Analytics

Table 2 shows statistics for year 2016 (start from 24 October), 2017, 2018, and 2019 (up till 01 October). For each year, we are presenting attributes and their respective values. Attributes include total visits, unique visits, total page views, total search keywords, and total out-links. The statistics show that total number of visits is increasing every year, i.e., in 2016 total visits were counted 859 and in 2019 total visits count is 7,942. It can be seen that bounce rate is increasing every year as the users are increasing. The reason is that as USE is not only a search engine but a portal as well and provides latest content on its home page. Hence, it is quite obvious that some users just visit USE to read the latest content and leave the page after reading. Overall, these statistics show that USE is getting more attention year by year.

5.2. Visitor Browser

Information about the visitor browser is really supportive for solving the browser inconsistencies. Designers need to keep in mind that cross browser testing is necessary to avoid the most common problems [29]. Hence on the basis of this point, we obtained the information about it to avoid any cross-browser inconsistency. We found that 55.89% of visits are from Chrome browser, so USE developers should pay more attention to this for display of USE. Other browsers include Firefox, Opera, Safari, and others. More than 15 different types of browsers and their types are found in our record while tracking the users of USE, e.g., Mobile Safari, Chrome Mobile etc.

5.3. Device Type

We observe that more than 80% of the users use desktop/laptop to visit USE. Other devices include smartphone, tablet and phablet. This information is really helpful as it suggests to improve the site visibility with respect to desktop devices. Device type information helps to make the website responsive with respect to different screen sizes. It is also possible to show more on large screens and less on small screens.

5.4. Event Logs

These type of logs provide two levels of information, user queries and corresponding clicks on search results. It can be used to know user interest on the website e.g., most clicked results and corresponding queries, images, tabs visit etc. Keeping this information in mind, further changes can be made in these sections of website to attract more users. In event logs section, a sample shows that 0.1% of visits contain search term “Pakistan” and clicks on Urdu Wikipedia outlink.

Table 4: Number of unique searches for different tabs

Tab Name	**Number of unique searches**
Web	3,892
Books	872
Islam	1,148
News	1,035
Poetry	966
Sports	270
Videos	559
Wikipedia	266
Famous websites	187

5.5. Site Search Keywords

Matomo also provides searched keywords information for each user. We observe that a total of

54,710 queries are searched and "Pakistan" keyword is at top. We also analyzed the length of searched keywords i.e., how many are single word, two words and so on. Most users search single word query on USE and their total count is 15,694. Similarly, for two-word, three-word, and four-word queries, we have frequency values of 2,036, 1,069 and 548 respectively.

5.6. Website Tabs Usage & Search Statistics

As USE has many sections (tabs) e.g., web, news, poetry, books, etc., here we present the usage distribution of each section. Obtained statistics show that most people visit the home page of USE with about 24% share. Other top visited sections are web, poetry, Islam, news and videos tabs with a share of 13.5%, 11%, 3.2%, 1.9% and 1.1% respectively. These statistics also indicate the interest of users on USE at section level. It also suggests which section should be further improved to increase user engagement. Similarly, we also collect information about number of search queries for different tabs. Table 4 shows unique search statistics in different tabs of USE. We have mentioned the number of unique search keywords for each tab. Out of total searches, 9,195 searches are unique keywords.

5.7. Visitor Log

To analyze the user behavior, we made a visitor log displaying its profile and details as each and every minor information is important to be logged. Table 5 shows the user-level details of a sample visitor. It has

Table 5: Visitor profile attributes

Attributes	**Values**
IP address	66.249.93.88
Visitor profile ID	1362e2e13b0b8819
Browser type	Chrome mobile
OS type	Android 6.0
Device type	Smartphone, Motorola
Location	United States
Total time spent	3min 34s
Number of actions	5
page views	1

different attributes about the visitor like IP address, user ID, browser type, Geo-location etc. A sample taken from record shows that a user from the United States with IP address 66.249.93.88 visits USE through Android 6.0 using chrome mobile browser in Motorola Smart-phone. He spends 3min 34s on USE and performs 5 different actions. He finds 1 item of his choice and redirects to the respective link. His actions include www.humkinar.com.pk/Poetry, www.punjnud.com and some other outlinks.

5.8. Website Performance Statistics

The performance monitoring of our website with respect to page load time, network latency, and server serve time is also calculated. As it is not affordable to overlook the significance of website load speed because clients who are baffled by a slow page speed are probably going to leave the site. This is why it is important to improve the website load time to enable clients to get where they're speeding up. We found that average page load time of USE is 1.6403s, average network latency is 0.5116s, and average server serve time is only 0.0064s.

5.9. Others

We find that 40 different versions of operating systems like Windows, Linux, Ubuntu, Android, iOS, etc., are used to visit the USE. By analyzing these statistics, we observe that Linux is the most used Operating System (OS) with 24.4% of the total users. We also observe that USE visitors belong to more than 50 different countries with Pakistan at the top position with 84.06% share. Other countries include United States, Australia, India, Saudi Arabia etc. These properties may seem less important but they actually guide the developers to avoid any limitations in their website. Another important information about the user is to find the channel type from where he/she is accessing the site. In our case, we found three channels, i.e., search engine, websites, and social network. It means that users are visiting USE through other search

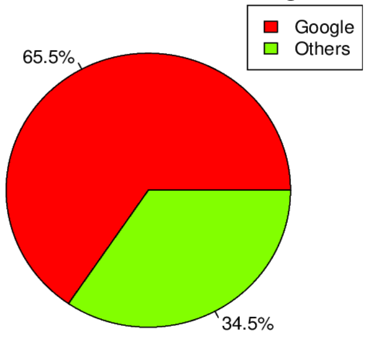


Figure 3: Search platform preference for Urdu content

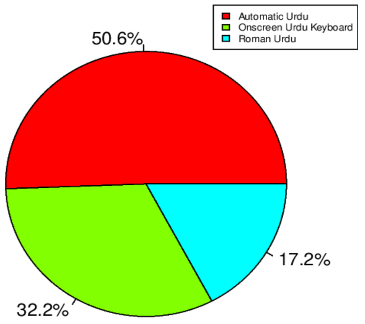


Figure 5: Urdu typing methods

engines, from some website redirection, or from any social network like Twitter, Facebook etc.

6. User Survey for Humkinar Urdu Search Engine

In this section, we discuss the user survey results and feedbacks regarding USE. To observe the user behavior and interest on Humkinar, we conducted a survey in which different questions regarding the features and search results of Humkinar were asked. We got a total of 87 responses from both males and females subjects. Out of the 87 users, 69% were males and 31% were females. Most of them belong to the age of 20-30 as majority of the subjects were students. We asked them to fill the questionnaire by visiting Humkinar and checking the features and functionalities step by step and answer the questions accordingly. It was necessary to ask them about Urdu typing experience as Urdu typing is the key functionality for our search feature. Most of them answered Yes, i.e., 70.1%, while 29.9% answered in No, which shows that majority of users already know how to type Urdu. Figure 3 shows that 65.5% of the users said that they use Google to find Urdu documents while remaining 34.5% use other platforms to search Urdu content.

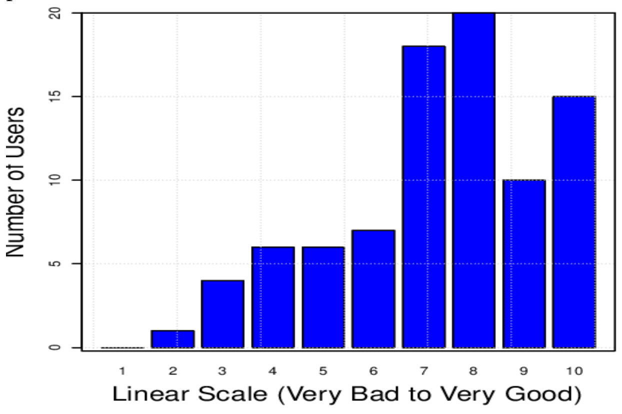


Figure 4: Subjective test results – Humkinar design

From the design and features point of view, we prepared a separate section containing questions related to design view only. To get the overall feedback about design from a user, we used 1-10 linear scale range, i.e., 1 shows very bad and the number goes on to 10 showing very good. Figure 4 shows the chosen values by users regarding design of Humkinar. Majority users, i.e., 23% chose scale value 8. 71.3% users voted that they like the color scheme and presentation of Humkinar frontend. Humkinar uses Nafees Nastaleeq Urdu font and 97.7% users liked its rendering style and readability. For Urdu typing, Humkinar provides three typing methods: 1) Automatic Urdu Typing 2) On-screen Urdu Keyboard 3) Roman Urdu Typing. Figure 5 provides division of users based on the Urdu typing methods. For search results, an individual section was made to ask search result questions for different tabs of Humkinar. 59.8% users said that it is easy to find their required results using this platform, 26.4% selected the option of ”Very Easy”, and 13.8% of the users found it difficult to search Urdu content using Humkinar.

Overall, the feedback was satisfying as majority of the responses were positive. We also got comments from each and every user at the end of questionnaire and many useful suggestions were given by them, e.g., add more sections like cooking, health, horoscope, currency rates, biography page for famous personalities etc. Some of them proposed that we should also add voice search option to find query results. We can conclude that the overall survey feedback was good enough to implement new functionalities in Humkinar for the ease of users and to make it more adaptable.

7. Conclusion

In this study, we analyzed Urdu Search Engine (USE) user behavior and obtained different statistics. For this purpose, we have used open-source solution “Matomo” and customized it according to our requirements. With this tool, we have analyzed last 35 months user search behavior on USE. For this interval, our findings show that USE is visited 23,022 times and total page views are 117,439. Total searched queries are 54,710, top query is “Pakistan” and most search queries are single word query (15,694). About 84.06% visitors belong to a single country, i.e., Pakistan and most of them used Chrome browser (55.89%) with Linux (24.4%) OS. While loading the USE website, total load time is only 1.6403 seconds. By incorporating click information of visitor for search query, we updated ranking algorithm of search results. Further, we presented user survey results, total 87 participants, regarding USE design, content, and features. It was found that 65.5% users use Google to search Urdu content. 71.3% users liked the interface of USE. Overall feedback is agreeable and it is helpful for us to improve the quality of USE with respect to design, features, and content. In future, we plan to use “Matomo" stack personalization” to implement personalization feature in Humkinar for enriched user experience.

8. Acknowledgement

This research work was funded by Higher Education Commission (HEC) Pakistan and Ministry of Planning Development and Reforms under National Center in Big Data and Cloud Computing.

9. References

[1] Mike Cafarella and Doug Cutting. Building nutch: Open source search. Queue, 2(2):54, Jan 2004.
[2] Daniel C. Fain and Jan O. Pedersen. Sponsored search: A brief history. Bulletin of the American Society for Information Science and Technology, 32(2):12–13, 2006.
[3] Google. www.google.com/ (visited on 30 Sep, 2019).
[4] Yahoo. www.yahoo.com/(visited on 30 Sep, 2019).
[5] Bing. https://www.bing.com/(visited on 30 Sep, 2019).
[6] Baidu. https://www.baidu.com/(visited on 30 Sep, 2019).
[7] Duckduckgo. https://duckduckgo.com/(visited on 30 September, 2019).
[8] Alex Chris. Top 10 search engines in the world, 2018. https://www.reliablesoft.net/top-10-search-engines-in-the-world/(visited on 30 September, 2019).
[9] Jayni Foley. Are google searches private-an originalist interpretation of the fourth amendment in online communication cases. Berkeley Tech. Law Journal, page 447,2007.
[10] DataReportal Follow. Digital 2019 pakistan (january 2019) v02, Feb 2019.
[11] Humkinar urdu search engine. https://www.humkinar.com.pk/(visited on 30 September, 2019).
[12] Rick Tansun. 10 web analytics tools for tracking your visitors, Mar 2009.
[13] Web analytics: Why they matter — siteimprove (en).
[14] Free web and mobile analytics software. https://matomo.org/(visited on 30 September, 2019).
[15] Suraj Chande. Google analytics -case study. 01 2015.
[16] Dirk Lewandowski. Search engine user behaviour: How can users be guided to quality content? 28:261–268, 01 2008.
[17] Songhua Xu, Yi Zhu, Hao Jiang, and Francis C. M. Lau. A user-oriented webpage ranking algorithm based on user attention time. In AAAI, 2008.
[18] Francesco Brigo, Simona Lattanzi, Michael O Kinney, Nicola Luigi Bragazzi, Laura Tassi, Raffaele Nardone, and Oriano Mecarelli. Online behavior of people visiting a scientific website on epilepsy. Epilepsy & Behavior, 90:79–83, 2019.
[19] L. A. Granka, T. Joachims, and Geri Gay. Eye-tracking analysis of user behavior in www search. In Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’04, pages 478–479, New York, NY, USA, 2004. ACM.
[20] Bartomeu Riutord Fe. User behaviour on google search engine. International Journal of Learning, Teaching and Educational Research, 28:104–113, 2014.
[21] Fadhilah Mat Yamin. Interfacing google search engine to capture user web search behavior. International Journal of Electronic Commerce Studies, 4(1):47–62, 2013.
[22] Xiaozhe Wang, Ajith Abraham, and Kate A. Smith. Intelligent web traffic mining and analysis. Journal of Network and Computer Applications, 28(2):147–165, 2005.
[23] Saar Kagan and Ron Bekkerman. Predicting purchase behavior of website audiences. International Journal of Electronic Commerce, 22(4):510–539, 2018.
[24] N. Goel and C. K. Jha. Analyzing users behavior from web access logs using automated log analyzer tool, 2013
[25] D.A. Adeniyi, Z. Wei, and Y. Yongquan. Automated web usage data mining and recommendation system using k-nearest neighbor (knn) classification method. Applied Computing and Informatics, 12(1):90 – 108, 2016.
[26] T. T. S. Nguyen, H. Y. Lu, and J. Lu. Web-page recommendation based on web usage and domain knowledge. IEEE Transactions on Knowledge and Data Engineering, 26(10):2574–2587, Oct 2014.
[27] N. Kaur and H. Aggarwal. Web log analysis for identifying the number of visitors and their behavior to enhance the accessibility and usability of website. International Journal of Computer Applications, 110, 2015.
[28] Google Analytics Solutions - Marketing Analytics & Measurement. https://www.google.com/analytics/(visited on 01 October, 2019).
[29] Nepal Barskar and C.p. Patidar. A survey on cross browser inconsistencies in web application. International Journal of Computer Applications, 137(4):37–41, 2016.