World Applied Sciences Journal 16 (3): 437-448, 2012
ISSN 1818-4952

Urdu Part of Speech Tagging Using Transformation Based Error Driven Learning

[1]Fareena Naz, [1]Waqas Anwar, [1]Usama Ijaz Bajwa and [2]Ehsan Ullah Munir

[1]Department of Computer Science,
COMSATS Institute of Information Technology, Abbottabad, Pakistan
[2]Department of Computer Science,
COMSATS Institute of Information Technology, Wah Cantt, Pakistan

Abstract: This paper presents a preliminary achievement of Brill's Transformation-Based Learning (TBL) approach to solve disambiguation problem of Urdu language. In the last few years lots of work has been done on European and South Asian languages but comparatively lesser efforts have been made in context to Urdu language. Keeping this aspect in mind, this study presents Part of Speech (POS) tagger for Urdu language using Data Driven Approach, called Brill's Transformation-Based Learning (TBL). This method automatically deduces rules from a training corpus with accuracy comparable to other statistical techniques as well as it possesses significant advantages over others tagging approaches. In this study, POS tagger is trained on Urdu corpus, which in contrast to English, is free word order language with inflectional characteristics and complex morphological nature. The corpus consists of 123775 tokens and 36 tag sets. The proposed POS tagger achieved a significant accuracy of around 84%. Precision, Recall and F-Measure has been calculated for complete test corpus. Error analysis (confusion matrix) for most confusing tag pairs has also been presented along with brief overview of Urdu language and tagging examples of Urdu language which elaborates the model in its best fashion. Performance of the proposed tagger has been compared with N-gram POS tagger and it is clearly evident that the proposed transformation based method outperforms the N-gram based POS tagger.

Key words: Urdu Language · Transformation-Based Learning · Statistical models

INTRODUCTION

POS tagging is the process of labeling words ofsentences according to morphological aspect of that language. Merely assignment of tags to words is not sufficient because in NLP mostly words are ambiguous (i.e. a word with multiple possibilities of tags), thus major purpose of the tagger is the disambiguitation of text. In Natural Language Processing (NLP), part of speech tagging plays an important part to resolve human language ambiguity in different analysis levels and its output (tagged data) can also be used in various applications of natural language processing such as Information Extraction, Information Retrieval, Question Answering, Speech Recognition, Text-to-speech conversion, Partial Parsing, Machine Translation and Grammar Correction etc [1]. Thus importance of tagged data can't be ignored at all. If correctly disambiguated text is provided to all of the above processing systems than they will perform much better.

The initial methods used to address part of speech tagging are Rule Based ones. After 1980's Statistical approach came into existence and gained more popularity. Later on, in Brill [2, 3] introduced Transformation Based Error Driven Learning, a method to induce constraints from tagged corpus. Nowadays, a blend of various approaches is used to get better results. As far as typical Rule Based approaches are concerned, they assign tags to words by using contextual information. Rules are developed based on contextual framework, therefore known as context frame rules.

Besides using contextual information for making rules, some systems also make use of factors like punctuation or capitalization. Importance of this information depends upon the specific language properties. Information about capitalization seems beneficial for languages like German or English for tagging an unknown word, but as every language has its own grammatical structure and requirements, so rules belonging to factors like capitalization would not help in

Corresponding Author: Fareena Naz, Department of Computer Science, COMSATS Institute of Information Technology, Abbottabad, Pakistan.

case of languages like Hindi, Urdu, Arabic and Pashto etc. Although rule based approaches use rules which require lots of human effort to write them for a specific language.

In Ailing [4] statistical approaches, large-sized corpus is distributed into two parts for analysis. In the first part, a subset of corpus is used for training a tagger to learn a statistical model. In the second part which is the testing phase, the learned statistical model will be used for the purpose of tagging untagged text in testing phase. In short, statistical taggers require complex computations and higher costs of storage, adaptation and improvement.

In the last few years, the trend has shifted towards the development of Transformation Based Learning (TBL) taggers. Statistical techniques were considered to be the most successful ones as compared to rule based techniques until the TBL tagging approach was proposed. As compared to linguistic Rule Based and Statistical approaches, TBL tagging are comparatively easier to develop. Moreover, they are language and tag-set independent, furthermore being a supervised method it automatically acquires model from annotated corpora.

In this study, we present a model for part of speech tagger based on TBL method for Urdu language, which is different from English regarding syntax and morphology. The rest of the study is organized as follows: In section 2 gives a brief overview of Urdu language; section 3 lists all the challenges that differentiate Urdu language from other languages hence making this problem much harder. Section 4 consists of an overview of few POS tagging models on various languages along with the discussion of what work has been done on Urdu language so far. Section 5 describes the methodology used in this study for POS tagging, the POS tagset and the corpus used in the experiments along with an example which elaborates the proposed model. Section 6 presents the experimental results along with discussions regarding the standardized metrics used for evaluation and the error analysis (confusion matrix). Finally, conclusion and future work has been proposed.

Brief Overview of Urdu Language: Urdu is the national language of Pakistan. It is an Indo-Aryan language and belongs to Indo-European family of languages. The name 'Urdu' itself is originated from Turkish word 'Ordu' which means 'camp/army' or 'horde'. It is also considered as the language of five Indian states and is spoken worldwide. Other than Pakistan and India, majority of its speakers live in UAE, USA and UK. Because of its rich morphology and highly inflected nature, it is also considered as the language of poets. Urdu and Hindi are closely related languages having same SOV word order. It shares its morphology, phonology and grammatical structures with Hindi. It shares its vocabulary with Arabic, Persian, Sanskrit, Turkish and Pashto language [5-7].

Challanges for Urdu Pos Tagging: Over the past few years a lot of work has been done regarding Part of Speech tagging for English, South Asian and other European languages, but similar developments for Urdu language are still in infancy stage. The major issues hindering research on this language are:

- Lack of availability of large Urdu corpora.
- Absence of a well-defined and standardized part of speech tag-set for Urdu language.
- Unavailability of a comprehensive Urdu Part of Speech lexicon.
- Acquiring distinctive contextual information for Urdu language is a cumbersome task. (e-g) a word can have multiple possible tags as in the example given below,

اس نے کھانا کھانا ہے۔

کھانا /NN vs. کھانا /VBI

As کھانا can be taken in sense of verb as well as noun, so to find out appropriate tag according to the context in which it is used requires contextual information.

- The concept of capitalization rules doesn't exist in Urdu language which makes Part of Speech tagging a more challenging task [5].

The above mentioned points are the vital obstacles in Urdu language processing research [8].

Related Work: In this section, an overview of some famous POS tagging models for various languages as well as Urdu is given. Several approaches have been used for building POS taggers such as Linguistic, Statistical and Transformation Based Learning Approach explained below.

Linguistic Approach: In this study [9] developed a application of POS tagging by assigning tags to the words on the basis of surrounding words and character affixes, by using hand crafted rules. Corpus used was Brown corpus, tag set was composed of 30 tags and accuracy rate achieved was 90% [10].

Table 1: Results of the POS tagger [13]

No. of words in Lexicon	No. of Rules	POS tagger Accuracy
100	10	40%
1000	40	62%
10,000	70	76%
100,000	120	88%

In this study [11] developed a system TAGGIT to assign tags to words, which uses suffix and lexicon information for tagging. The main feature of TAGGIT was to deal with exceptions i.e. capitalization, apostrophes with words.77% disambiguation was done with TAGGIT and 23% was done manually. CG (Constraint Grammar) is known rule based tagger which handles both POS and grammatical functions tagging. EngCG is known as a close competitor to statistical data driven taggers. EngCG2 uses 3,600 rules and Swedish CG uses 2,100 rules for disambiguation. In this study [12] presented rule based tagger for Icelandic text named as ice tagger, which with the help of an 'unknown word guesser' named as iceMorphy gives accuracy (91.54%) better than TNT taggers which stand at an accuracy (90.44%).

In this study [13] presented a first ever Pashto rule based part of speech tagger, as it was a pioneer contribution in rule based approach for tagging Pashto language. For tagging any language, tag-set is needed to be developed and every language has its own characteristics and morphological features which differentiates it from other languages. Therefore a tag-set was also developed specifically for Pashto language, which composed of 54 tags in total. This architecture is very simple and gives reasonably good accuracy.

The Table: 1 given above clearly shows that with the increase of no. of words in lexicon and no. of rules, accuracy also improves.

Stochastic Approach: Stochastic approaches employ training models, which by using tagged/untagged corpus find parameters. When a model gets trained, it can be used for tagging raw texts. Statistical approaches mostly use HMM for part of speech tagging. In such approaches, lexical and transition probabilities are used to find a tag for a word. The TNT (Trigram HMM) and HMM tagger are considered as standard. In TNT tagger, preceding two tags are used to calculate transition probability. In this study [14] achieved an accuracy of 96%-97%, when the performance of TNT tagger was tested on the corpus (NEGRA, Penn Treebank).

Most recently in [15] conducted a comparison of various taggers on Urdu language in order to investigate whether commonly used disambiguation techniques and standard POS taggers can be used for tagging text of Urdu language. For this purpose a tag-set was also proposed and 100,000 tokens of corpus were used for training the model and better results were obtained showing accuracy (94.15%) when lexicon extracted from same corpus was used and showing accuracy of (95.66%) when separate lexicon was used. It was concluded that in case of known words SVM tool gives best accuracy and incase of unknown words CRF tagger gives best results.

Transformation Based Learning Approach (TBL): In Brill [16] introduced a transformation based error driven approach, a machine learning approach used in many different areas specifically for handling classification problems. TBL approach is a sort of hybrid approach, because it makes use of both statistical and rule based approaches. The basic idea is very simple, initially most likely tag is picked from training corpus by using any statistical approach and then later on it applies set of predefined transformation templates in order to check that whether a tag should be replaced with another one. If any new rule is found during training it saves that in list, so that it can be used for further tagging raw text. In Brill [2] introduced a new approach to POS Tagging and is called Transformation-Based Error Driven Learning or "Brill Tagger". A functional description of this learning algorithm to POS tagging is given below and a more detailed description along with example is presented in methodology section V.

In this study [17] presented a variant of Brill's implementation that uses GA to automatically produce transformations to be used in the Brill tagger and provide an adapted ranking of the rules by using a natural process of re-ordering done by the crossover operator. The GA Brill tagger performance was compared with the original Brill tagger performance, while assuming a closed vocabulary. The input data to the GA Brill tagger is the same as the one used for original Brill tagging system which is the Penn Treebank Wall Street Journal corpus with a training set of 600,000 words and a dictionary. However results show that the GA Brill tagger implementation achieved accuracy up to 89.8% on the same corpus which is much less than the 97.0% accuracy of the traditional Brill tagger.

In this study [18] designed a supervised transformation-based Khmer POS tagger. As Khmer language is syntactically and morphologically different from the English language the authors modified Brill's implementation according to the characteristics of Khmer language. For dealing with the unknown words a hybrid

approach was proposed, which combined rule based and trigram. On test data it achieved an accuracy of 91.96% which includes 9% of unknown words. For the proposed system tagset of 27 tags (excluding punctuation marks) and Corpus of 41,061 words (1,298 sentences) was defined.

In this study [19] did POS tagging for Arabic text and found that TBL gives better accuracy as compared to other techniques. The developed tagger achieved an accuracy of 98.6% on the training data and 96.9% on the test data with same templates used for other languages. To deal with the unknown words, an n-gram technique was opted to select best tag from a list of possible candidates.

MATERIALS AND METHODS

This section emphasizes on the practical application of Brill tagging i.e. Transformation Based Learning Approach (TBL) to Part of Speech Tagging, which appears to be the most efficient approach with considerable accuracy. On the broad spectrum, the main theme is very simple; initially tagger assigns tag to each word by guessing and then reducing the mistakes. Like n-gram tagging, it is supervised approach because we need correctly tagged data in training. But unlike n-gram tagging, rather than adding up observations, it collects a list of transformation rules for fixing the details of errors in [20].

The process of Brill tagging can be understood from an example of map generation. Normally the surveyors of geological departments make the maps by reading the ground from the heights or vantage points or through aerial reccoinessanse and drawing these minor details on the plain sheets, i.e. drawing the grid intersections, contours of ground, relief of terrain, tracks, roads, valley walks built up areas, jungles, marshes, rivers, lakes etc and that all the detail has to be according to a set scale to make it easy for the reader. One of the strengths of this method is that it makes use of more context than Bigram or Trigram, Thus wider coverage of lexical and syntactic regularities are taken in account [21].

Transformation Based Error Driven Part-of-Speech Tagging: This study presents a Part of Speech Tagging for Urdu language using Brill's Transformation-Based Error Driven Tagger. It is a Machine Learning approach and draws inspiration from both Rule Based and Stochastic taggers. It can be called a hybrid approach, because tagger first uses statistical technique to tag data at initial level and then fix all mistakes that were introduced by the statistical techniques [21].

For fair evaluation of the system, the corpus is split into testing and training sets. For training, 90% of the whole corpus was chosen and the rest 10% for the purpose of testing the tagger. In this study a cutoff was used to distribute the whole corpus into training and testing sets, which is given below:

$$Cutoff = int(num\_sents*train)$$

Where num_sents represents that how many sentences of training and testing data to use. The train represents the fraction of the corpus to be used for training. So basically value of the corpus distributes the whole corpus into training and testing which is as follows:

Training = : Cutoff (the value of cutoff represents the size of training data)
Testing = Cutoff : num_sents(value from cutoff to onward till num_sents represents the testing data)

Training Phase: First we create an un-annotated version of the training corpus and then pass it to the initial state annotator as shown in Figure 1 given below:

Now when un-annotated data comes to initial state annotator, any approach can be used for the purpose of initially annotating the data with the Part of Speech tags. The following different approaches can be used:

- We can simply label all the words with their most likely tags (i.e. the way it is annotated in the training corpus). OR
- Simple statistical techniques (n-gram taggers) can also be used for the initial state annotation. OR
- All words can be straight away labeled with the noun tag.

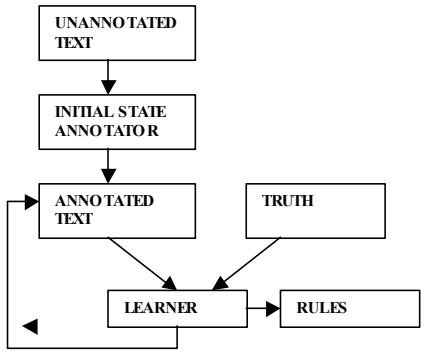

Fig. 1: Transformation-Based Error-Driven Learning [2]

From the above mentioned choices, the one used for the purpose of initial state annotation is the statistical n-gram tagger (i.e. Unigram tagger, Bi-gram tagger and Regexp tagger). Initially the un-annotated data is passed to unigram tagger. The unigram tagger by using the maximum probability P (t_i | w_i), finds the best tag t_i^* (according to eq 1) and assignes it to the targeted word. It does it by maintaining the dictionary listing out all the words $w_1, w_2, w_3, \ldots w_n$ and their corresponding tags t_1, t_2, t_3, ….. t_n. Tags and the words are represented by t_i and w_i respectively.

$$t = \arg\max P(t \mid w) \quad (1)$$

Where P(t_i | w_i) is computed accordingly as in eq (2)

$$P(t \mid w) \approx \frac{C(t,w)}{C(w)} \quad (2)$$

Here C (w) is the number of times the word w has appeared in the training data. And C (t_i, w_i) is the number of the times; tag t appears with word w.

As the Unigram tagger assigns a 'None' tag to all unknown tokens. The sparse data problem is handled by using unigram tagger with a backoff. Here the Regular Expression Tagger (Regexp) tagger is used as a backoff , so that when unigram tagger fails to determine a tag, it consults backoff. The Regexp tagger can be used as a fall-back tagger for handling unknown words and as it gives better score than the Default tagger, so that's why it is employed as backoff tagger, which uses simplest theme of word suffixes (i.e. specific suffix or prefix strings) for determining part of speech tag. By using Regexp Tagger, one can determine the part of speech tags by defining his own word patterns.

Keeping in mind, the morphological structure of Urdu language, only two regular expressions are taken in accounts which are given below:

```
>>> Patterns = [(r'^-?[0-9]+(.[0-9]+)?$', 'CD'), # Cardinal
                Numbers (r'.*', 'NN') # Nouns (Default)]
```

After being annotated by the unigram tagger, accuracy of the unigram tagger is also computed with or without the backoff tagger. And then the output i.e. the data which has been annotated by the Unigram tagger is passed to Bigram tagger for tagging. The Bigram tagger is similar to the Unigram tagger, but it uses dual information to find the most likely tag for each word (i.e.) current tag for a current word and the preceding tag. The context for the token is calculated by collecting tags for the current word and its preceding tag. On the basis of this context,

Table 2: Structure of the Bigram Dictionary [7]

t_{i-1} , w_i	t_i
NNP, چند	Q
Q , صنعتی	JJ
JJ, قوموں	NN
NN, میں	CM

the Bigram tagger by using Maximum likelihood estimation P (t_i | t_{i-1}, w_i) assigns the most probable tag t_i^* (according to eq 3) to the targeted word.

$$t = \arg\max P(t \mid t\, w) \quad (3)$$

Where P (t_i | t_i - w_i) is computed accordingly eq (4)

$$P(t \mid t\ , w) \approx \frac{C(t\, t, w)}{C(t\ , w)} \quad (4)$$

Here C (t_{i-1}, t_i, w_i) is the number of times the tag t_{i-1} appears with the tag t_i and with the word w_i in the training data.

C (t_{i-1}, w_i) is the number of the times; word w_i appears with the tag t_{i-1} .

The above Table 2 shows structure of a dictionary to be followed by Bigram model. In this table 2 w_i represents the current word with the preceding tag t_{i-1}, against the current tag t_i. It's a trivial task for the Bigram tagger to tag all those words that appear in the training, but performs poorly when the sparse data problem occurs. Thus the Bigram tagger is backed off to the Unigram tagger incase if any word pair doesn't appear in the training corpus holding the specific context. The Unigram tagger backs off to Regexp tagger, if the occurrence of the word is too sparse. Therefore it becomes a Sequential Backoff Tagging. Finally annotation done by the Bigram is also evaluated by computing its accuracy. The data tagged through Initial State Annotator is the Annotated Text (temporary corpus) which is tagged according to some fashion. This Temporary corpus is then compared with Truth (i.e. manually annotated corpus used to check the correctness of the output of tagged data by making its comparison with the output of the tagger) to check the correctness of the initial annotation done by the initial state annotator. Annotated Text (temporary corpus) then becomes input to the Learner. Where an ordered list of transformation is learned, which is applied on the output of initial state annotator to make it close to the truth.

- Transformation is comprised of the two components:
- Rewrite rule
- Triggering environment

Therefore an example of the rewrite rule is as follows:

Change the tag from NN to NNP

And an example of the Triggering environment is as follows:

The following word is tagged NNPC.

Taken together, the transformation with this rewrite rule and triggering environment when applied to the word صدر will correct the wrong tag.

صدر /NN->NNP پش/NNPC نے/CM خاص/……

Transformations are basically instantiated from a set of predefined transformation templates. Pre defined templates contain un-instantiated variables and are of the form;

If Trigger, then change the tag X to the tag Y

Where X and Y are variables. The interpretation of the transformation template is that if the rule triggers on a word with current tag X then the rule replaces current tag with resulting tag Y. During training phase, tagger guesses the values for X and Y and the variable mentioned in the context, to create thousands of candidate rules. In each iteration of learning, we pick the transformation, which when applied on the corpus gives best score, where transformation is scored according to its net benefit as follows:

Score = Fixed - Broken

Where Fixed = num tags changed incorrect -> correct
and Broken = num tags changed correct -> incorrect

After adding the best scored transformation in the final ordered transformation list, the whole corpus is updated by applying this high scored transformation. In this way, learning continues until there is no further improvement.

Predefined Transformation Templates Are Divided into Two Categories:

- Non Lexicalized Templates
- Lexicalized Templates

1. The preceding (following) word is tagged z.
2. The word two before (after) is tagged z. One of the two preceding (following) words is tagged z.
3. One of the three preceding (following) words is tagged z.
4. The preceding word is tagged z and the following word is tagged w.
5. The preceding (following) word is tagged z and the word two before (after) is tagged w.

where a, b, z and w are variables over the set of parts of speech.

Fig. 2: List of Non Lexicalized Templates

Table 3: List of non lexicalized transformations learned from Urdu training corpus

	Change tag		
#	From	To	Condition
1.	PR	DM	if the tag of the following word is 'NN'
2.	SM	PM	if the tag of the preceding word is 'PM'
3.	DM	PR	if the tag of words i+1...i+3 is 'AUXT'
4.	CM	VB	if the tag of the preceding word is 'NN' and the tag of the following word is 'AUXT'
5.	DM	PR	if the tag of the preceding word is 'SC' and the tag of the following word is 'NN'
6.	NN	NNP	if the tag of the following word is 'NNPC'
7.	DM	PR	PR if the tag of words i-2...i-1 is 'VB'

Non-Lexicalized Version of the Tagger: In this case, transformation templates depend only on the surrounding tagging information to change the tag of the current word and do not make reference to specific words. In the non-lexicalized tagger, the transformation templates we added are listed below in Figure 2: Change tag a to tag b when:

For learning a transformation, the learner will apply each transformation and add up the number of errors generated. Once all of the transformations have been applied, learner picks one transformation in each iteration that result in maximum error reduction. As it is an iterative process, so learning will continue until stopping criterion is met (no rule could be found whose application results in further improvement).

In Table 3, we list the non lexicalized transformations learned from training on the Urdu Corpus:

In Table 3, The first transformation states that a PR should be changed to a DM if the tag of the following word is 'NN'.

Lexicalize Version of Tagger: Statistical n-gram taggers have not yet directly encoded the relationships between words. As there are lots of beneficial relationships, such as the relationship between current word and the previous word, between a current tag and the next word, that even can't be directly captured by N-gram taggers or Markov-model based taggers. Even the non-lexicalized version of transformation-based tagger also lies within the same category of N-gram taggers or Markov-model based taggers, because its transformation templates also do not

1. The preceding (following) word is w.
2. The word two before (after) is w.
3. One of the two preceding (following) words is w.
4. Current word is w and the preceding (following) word is x.
5. Current word is w and the preceding (following) word is tagged z.
6. The current word is w, the preceding (following) word is w and the preceding (following) tag is t.

Where w and x represent variables for all words in the training corpus, and z and t represent variables for all parts of speech.

Fig. 3: List of Lexicalized Templates

Table 4: List of lexicalized transformations learned from training

#	Change tag From	To	Condition
1.	SM	PM	if the text of words i+1...i+3 is ','
2.	VB	VBL	if the text of the preceding word is 'شروع'
3.	VB	VB	if the text of the following word is 'گئی'
4.	CM	VBL	if the text of words i-2...i-1 is 'ادا'
5.	Q	PR	PR if the text of words i+1...i+2 is 'زیادہ'

Table 5: List of transformations learned from training data

SCORE	FIXED	BROKEN	OTHER	RULES
89	347	258	17	PR -> DM if the tag of the following word is 'NN'
76	77	1	25	SM -> PM if the tag of words i+1...i+3 is 'SM'
67	67	0	2	SM -> PM if the tag of the preceding word is 'PM'
50	50	0	0	VB -> VBL if the text of the preceding word is شروع
31	33	2	2	VB -> VBL if the text of words i-2...i-1 is ادا
17	48	31	3	DM -> PR if the tag of the preceding word is 'SC', and the tag of the following word is 'NN'
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Table 6: List of errors generated from testing data

LEFT CONTEXT	WORD/TEST->GOLD	RIGHT CONTEXT
CM/کو NNP/جمعرات	NN->NNP/بینکنگ	CM/میں ابتدائی
AUXT/تھی AUXA/سکتی	NNCM->NN/پہاں	P/وہ SC/کہ CM/تک
JJ/پختہ PR/وہ	VB->VBL/ہوئے	JJ/مکمل PM/-
CM/پر NN/اقدر JJ/مل	PM->NN/،	CM/تک NN/جب
SC/کہ CM/تک	NN->JJ/جاری	VBI/کرنے
NN/پابندی CM/پر	VB->VBI/لگانا	AU/میں AUXA/جانیے
NNCM/طلب NN/کیے	NN->RB/بنا	NN/رکاوٹ Q/کسی
NN/قیمتیں CM/میں	NN->JJ/پسپا	VBL/ہوئی AUXT/ہیں
JJ/تھوڑی	JJRP->RBRP/سی	NN/بدنصیبی

Table 7: Tag free version of Training Data

Tagged version	Tag-Free Version
اس<PR>	اس
نے<CM>	نے
کھانا<NN>	کھانا
کھانا<VBI>	کھانا
ہے<VBT>	ہے
- <SM>	-

make reference to words. To overcome this problem, the transformation-based tagger was used along with contextual transformations that can make reference to words as well as part-of-speech tags. The transformation templates added are listed in Figure 3: Change tag a to tag b when:

In Table 4, we displayed a list of lexicalized transformations that were learned during training on the Urdu Corpus:

Testing Phase: In testing phase, an Un-annotated text is passed to an Initial State Annotator, which will tag all tokens according to their respective mechanism mentioned in detail in training phase. The output of the Initial State Annotator is then compared with the Truth (i.e. goal corpus which is the manually annotated corpus used as our reference) to check the correctness of the initial annotation done by the initial state annotator. Annotated Text (temporary corpus) then becomes input to the Learner, where the ordered list of learned transformation is applied on Initial tagged data (temporary corpus), from which few are given in Table 5.

Few contents of the list of Errors files which is generated as a result of testing are given in Table 6

Part of Speech Data, Tagset and Tagging Example: For Urdu, A Part of Speech tagged corpus of size 4323 sentences having 123775 tokens from CRULP [22] has been used in different experiments along with a tag-set of 36 tags.

Here is an example to illustrate the proposed methodology: First, an untagged version of training corpus is created and passed to the initial state annotator as shown in table 7 given below:

For Initial State Annotation, Sequential Back-off tagging is incorporated, which by using Unigram, Bigram and Regexp tagger initially annotates the data, as shown in Table 8 given below:

The data annotated through Initial State Annotator then becomes our temporary corpus, which is compared with correctly annotated data (Truth) for pointing out positions and list of errors, that were introduced by initial state annotator. As shown in Table 9:

After comparing temporary corpus with truth, it was found that کھانا was incorrectly initially tagged as <NN>, where as it should be assigned VBI tag.

Table 9: Comparison of temporary corpus with Truth (i.e. correctly annotated data)

Untagged data	Bigram	Unigram	Regexp	Temporary Corpus	Truth
	Initial State Annotator				
اس	اس<PR>			اس<PR>	اس<PR>
نے	نے<CM>			نے<CM>	نے<CM>
کھانا	کھانا<NN>			کھانا<NN>	کھانا<NN>
کھانا	کھانا<NN>			کھانا<NN>	کھانا<VBI>
ہے	ہے <VBT>			ہے <VBT>	ہے <VBT>
-	- <SM>			- <SM>	- <SM>

Table 10: Transformation generated through learner

Untagged data	Bigram	Unigram	Regexp	Temporary Corpus	Truth	Learner
	Initial State Annotator					
اس	اس<PR>			اس<PR>	اس<PR>	
نے	نے<CM>			نے<CM>	نے<CM>	
کھانا	کھانا<NN>			کھانا<NN>	کھانا<NN>	
کھانا	کھانا<NN>			کھانا<NN>	کھانا<VBI>	کھانا /NN->VBI
ہے	ہے<VBT>			ہے<VBT>	ہے<VBT>	
-	- <SM>			- <SM>	- <SM>	

Table 11 : Final list of rules

Score	Fixed	Broken	"Score= Fixed – Broken" Fixed=num tags changed incorrect -> Correct Broken=num tags changed correct -> Incorrect
			Rules
-> 89	347	258	NN -> VBI if the tag of the following word is 'VBT' اس/PR نے/CM کھانا/NN->VBI ہے/VBT ./SM
76	77	1	JJ -> NNP if the tag of the following word is 'NNPC' ایک/CD نئے/JJ چیف/JJ->NNP عملکاری/NNPC افسر/NNPC
.	.	.	.
.	.	.	.
.	.	.	.

This Temporary tagged data then become input to Learner, where the learner through learning will generate list of candidate rules from predefined list of templates (both lexicalized and non-lexicalized version of templates are considered) and will pick only one transformation which results in maximum error reduction (i.e. whose score is highest) among the list of thousands of candidate rules. As in table 10 given below, the following transformation is picked among the list of candidate rules, because it maximally reduces the error rate (scores best). Now after adding this transformation (i.e. rule) to list of final rules, we update the temporary corpus using the same transformation or rule and this iterative process continues until no further improvement is achieved.

As discussed earlier, transformations are generated from the predefined templates. So the transformation or rule which is displayed in table 10 given below is basically generated from the template given below:

- Using above template following Transformation is generated:

The above transformation emphasizes to change the tag from NN to VBI, if the tag is VBT. Using same mechanism the above transformation changes کھانا /NN->VBI, because it found that the next word is tagged with VBT.

The Table 11 given below shows the structure of list of best rules (final outcome of training phase) that is generated as a result of training. This is applied in testing phase after initially annotating the data to get tagged data.

In Table 11, only two rules are shown to give an idea of how list of rules actually looks like but in reality this list contains thousands of rules. Thus in each iteration of learning only one rule is picked among the thousands of candidate rules which scores the best. In this way final list of rules is maintained, which is the final output of our training phase. This list of rules is then further used to enhance the initially tagged data after initial state annotation in testing phase.

RESULTS AND DISCUSSION

Performance of the tagger is evaluated by considering different aspects. Depending upon the parameters; lots of experiments were conducted for training and testing of system. In Figure 4, Learning bars illustrate the performance (over all Accuracy) of the simple Unigram, Unigram with Backoff, Bigram with Backoff and Brill tagger, by keeping corpus size constant (number of. Tokens i.e. 32133) and varying training fraction from 0.5 to 0.9. Using simple Unigram, accuracy rates achieved are 72.276 %, 74.991 %, 75.604 %, 76.623 %, 77.638 % for 0.5, 0.6, 0.7, 0.8, 0.9 training fractions respectively with corpus size (32133 no. of tokens). Using Unigram with Backoff, accuracy rates achieved are 79.377 %, 80.701 %, 80.825 %, 81.589 %, 82.344 % for 0.5, 0.6, 0.7, 0.8 and 0.9 training fractions respectively with corpus size (32133 no. of tokens). Using Bigram with Backoff, accuracy rates achieved are 80.733 %, 82.015 %, 82.31 %, 82.798 %, 83.514 % for 0.5, 0.6, 0.7, 0.8 and 0.9 training fraction respectively with corpus size (32133 no. of tokens). Finally using Brill approach, accuracy rates achieved are 81.504%, 82.801%, 83.05 %, 83.388% and 84.740% for 0.5, 0.6, 0.7, 0.8 and 0.9 Training Fractions respectively with corpus size (32133 no. of tokens).

It is concluded from the results that accuracy of tagger not only depends on the fraction of the corpus used for training the tagger but also on the size of the corpus. Figure 5 shows a clear picture of the impact of training fraction and size of the corpus on the useful rules generation mechanism.

The learning bar illustrates the performance when using different sizes of corpus and it concludes that with the increase in the size of the corpus, the performance of the tagger also increases. The fraction of corpus used for

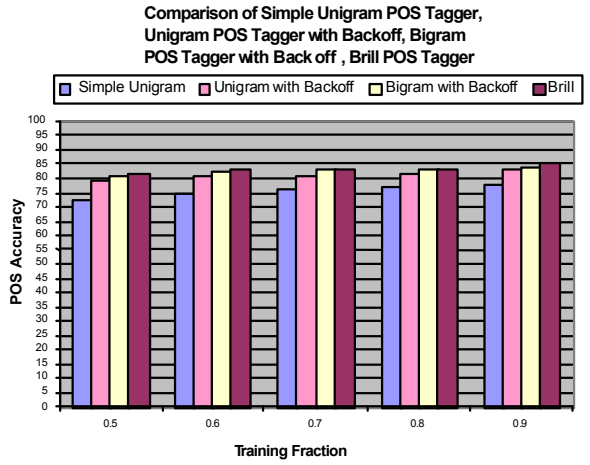

Fig 4: Comparison of Simple Unigram POS Tagger, Unigram POS Tagger with Backoff, Bigram POS Tagger with Backoff, Brill POS Tagger

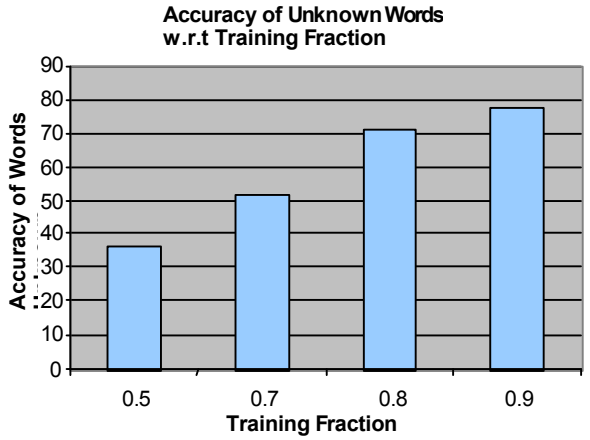

Fig. 5: Learning Bars for the impact of size of the corpus & training fraction on the generation of the useful rules mechanism

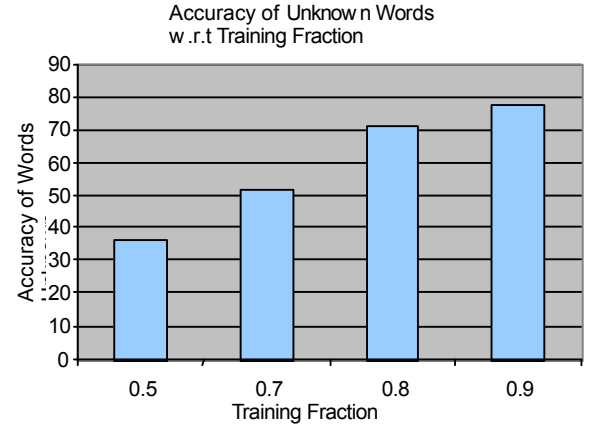

Fig 6: Accuracy of Unknown Words w.r.t Training Fraction (32133 No. of Tokens.)

training (training fraction i.e. 0.9 means ninety percent of the whole corpus will be used for training and ten percent for testing) also shows significant impact on the performance of the tagger .Large training fraction leads to better accuracy results.

In Figure-6, we show that accuracy rate of unknown words is related to fraction of the training corpus used to train the tagger. If we train the tagger on large amount of data we get accurate tagging results for unknown results, where the reason behind this is when we increase the corpus size we get more combination of tags, which leads to variety of rules that covers various aspects of multiple tags. Thus it concludes that results are dependent on fraction of training data used to train the system.

In Figure 6, learning bar illustrates that with (0.5, 0.7, 0.8 and 0.9) fraction of training, the tagger achieved an accuracy rate (36.27%, 51.785%, 70.73% and 77.77%) respectively for unknown words

Table 12 shows the precision, recall and f-measure of Brill Part of Speech Tagger, when it uses 90% and 60% of the corpus as training data and 10% and 40% as testing data. The above table 12 indicates that following results are achieved by conducting experiments on two different sizes of the corpus (16006, 32133 no. of tokens). With 60% training fraction and 40% testing data of 16006 no. of tokens (i.e. corpus size), 95.45 % precision, 85.71 % recall and 90.32 % f-measure was achieved. With the 90% training fraction and keeping corpus size same i.e. for 16006 no. of tokens, 97.560% precision, 95.238% recall and 96.385% f-measure was achieved. Thus with the increase in training fraction, we achieved better results. When the corpus size was increased from 16006 to 32133, it was observed that with the 60% training fraction and 40% testing data, 97.727% precision, 93.478% recall and 95.55% f-measure was achieved. With the 90% training fraction and keeping corpus size same i.e. for 32133 no. of tokens, 100% precision, 95% recall and 97.435% f-measure was achieved.

All the above results show that when the training data is increased, the results are improved. It means that the amount of training fraction used for training the tagger and the corpus size have significant impact on the performance of Brill Part of Speech Tagger. Therefore considering the sizes of corpus used for the experiments, our tagger achieved remarkable accuracy with a limited corpus.

Evaluation: Resulting disambiguated texts of our Brill Part of Speech Tagger have been evaluated using standardized metrics: i.e. Accuracy, Precision, Recall and F-Measure.

Table 12: Performance of Brill Model with Training Data [60% and 90%] and Test Data [40% and 10%] on corpus size (16006 and 32133) respectively

Training Fraction	No. Of Tokens	Precision	Recall	F-measure
.6	16006	95.45	85.71	90.32
.9	16006	97.560	95.238	96.385
.6	32133	97.727	93.478	95.55
.9	32133	100	95	97.435

Table 13: Confusion Matrix for most confusing Pairs overall

Correct Tags	NN	VB	JJ	VBL	PR	AUXA	NNP	AUXT	VBT	CM	Q	Total
NN	<2406>	4	19	0	0	0	16	0	0	13	3	2461
VB	40	<345>	0	94	0	8	0	0	0	0	0	487
JJ	242	3	<836>	0	1	0	20	0	0	0	12	1114
VBL	1	69	0	<102>	0	2	0	3	1	0	0	178
PR	2	0	3	0	<158>	0	0	0	0	0	10	173
AUXA	5	24	0	23	0	<207>	0	2	1	1	0	263
NNP	191	2	7	0	0	0	<329>	0	0	0	0	529
AUXT	1	3	0	1	0	12	0	<358>	0	0	0	375
VBT	0	0	0	0	0	0	0	7	<62>	0	0	69
CM	6	3	0	5	24	0	0	0	0	<1843>	0	1881
Q	1	0	20	0	10	0	0	0	0	0	<116>	147
Total	2895	453	885	225	193	229	365	370	64	1857	141	7677

- Accuracy = |pos(R)| / |pos(R)| + |neg(R)|
- Precision = Number of Correctly Tagged Tokens returned by Tagger / Total Number of Tagged Tokens returned by Tagger.
- Recall = Number of Correctly Tagged Tokens returned by Tagger / Total Number of Tagged Tokens in Correctly Tagged Data.
- F-measure = 2*Precision* Recall / (Precision + Recall)

The accurate and distinctive disambiguation of each token leads to an ideal case, where precision and recall will appear to be 1.0, which is our main motive i.e. to decrease the ambiguity and achieve the precision and recall as possibly closer to 1.0.

Error Analysis: The goal of Part of Speech Tagger is to assign an accurate Syntactic tag (Noun, Verb and Adjective) to each word in the sentence or in other words to resolve ambiguities of syntactic categories. Thus most taggers resolve such ambiguities effectively, but sometimes depending upon the nature of the language (such as if it comes to Urdu, it is of complex morphology and highly inflected nature), it becomes difficult to resolve such ambiguities. So far, the significance of our experimental results with the evaluation method has been shown but still these usual evaluations don't explain everything in-depth and merely give the percentage of correct words. If we come to know that what the actual misclassified words are, then such useful information can play a very important role in resolving the problem of unknown words and ambiguity by adding those features to taggers. Keeping this aspect in mind Confusion matrix shown as table 13 is composed, which extracts useful information based on major problems faced by the taggers during tagging. Thus through confusion matrix an error analysis is presented which basically portrays a clear picture of ambiguity between different part of speech in form of matrix.

- Matrix is based on (n tags * n tags) entries, where the rows indicate the correct tags and columns host the tagger output.
- Each Cell (i, j) of the matrix represents the count, that how many times tag i was categorized as tag j.
- The misclassification between different parts of speech is represented through Off Diagonal elements.
- The representation of correct classifications is done through leading diagonal entries.

For example, if we consider (JJ, NN) cell, there value is 242 .Thus the result of the cell represents that there were 242 no. of words that belong to JJ's , But were incorrectly tagged as NN group. After analyzing the confusion matrix it was found that mostly incorrect cases were based on noun, verb and adjective and so on.

With a thorough analysis of the confusion matrix it was concluded that accuracy of the tagger can be improved by resolving unknown and ambiguous words and by adding these features to tagger.

Conclusion and Future Work: Approach presented in this paper is an initial implementation of Brill's Transformation-Based Learning (TBL) based on Machine Learning Model and uses a supervised tagging technique. It is Hybrid Approach, because at the initial phase we used statistical techniques i.e. Unigram model and Bigram model to initially tag the data, along with the Regexp Tagger as a Backoff to handle the sparse data problem, which enhances the performance of the statistical models and then uses the learner over this initially tagged data to automatically acquire rules to resolve ambiguity, which are linguistically understandable as opposed to the large, complex and detailed contextual and lexical probabilities. With the corpus size of 123775 tokens and 36 tag sized tag-set, our POS tagger achieved significant result of 84^{+}%. By varying size of the corpus and the fraction of training data used for training the tagger, it was concluded that as the corpus size is increased, the performance of the tagger also increases. In this way overall accuracy and the unknown word accuracy is significantly increased (as we increase the training data, the ratio of unknown words decreases). Size of training fraction also impacts on the rules generation mechanism. With a large size of training fraction, large no. of rules are generated and vice versa. So it can be safely stated that if we extend the corpus size of Urdu, then we will be able to get similar performance for Urdu as in case of English.

Encouraged by the performance of the tagger attained from research work, in future, we would like to investigate how Brill's tagger performs using unsupervised approach for POS tagging of Urdu language because unsupervised POS tagging is a very good choice for languages with limited corpus.

REFERENCES

1. Christopher Manning, Hinrich Schuetze, 2000. "Foundations of Statistical Natural Language Processing", MIT Press, Cambridge, MA,
2. Brill, E., 1995. "Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging" Computational Linguistics, 21(4): 543-565.
3. Brill, E., 1994. A Report of Recent Progress in Transformation-Based Error-Driven Learning. ARPA-94.
4. Ailing Fleming, "Probabilistic Part of Speech Tagger", CSLL Final Year Project, pp: 1-138, 2002.
5. Javed, I., 1985. New Urdu Grammar. Advance Urdu Burew. New Dehli.
6. Sajad, H., 2007. Statistical Part of Speech Tagger for Urdu. MS Thesis. pp: 1-68.
7. Waqas Anwar, Xuan Wang, Lu Li and Xiao-long Wang, 2007. "A Statistical Based Part-of-Speech tagger for Urdu Language", International Conference on Machine Learning and Cybernetics, Hong Kong, China.
8. Samuels son, Christer and Atro Voutilainen, 1997. "Comparing a linguistic and a stochastic tagger", The 8th Conference on the European Chapter of the ACL, pp: 246-253.
9. Klein, Sheldon and Robert Simmons, 1963. "A computational approach to grammatical coding of English words", J. The ACM. 10: 334-347.
10. Cherry, Lorinda L., 1980. "PARTS - A system for assigning word classes to English text.Technical Report", Computing Science, Bell Laboratories,
11. Greene, G. Rubin, 1971. Automatic Grammatical Tagging of English, Technical Report, Department of Linguistics, Brown University, Providence, Rhode Island,
12. Hrafn Loftsson, 2008. "Tagging Icelandic text: A linguistic rule-based approach", Nordic Journal of Linguistics, 31: 47-72.
13. Ihsan Rabbi, Mohammad Abid Khan and Rahman Ali, 2009. "Rule-Based Part of Speech Tagging for Pashto Language", Proceedings of the Conference on Language and Technol.,
14. Brants, Thorsten, "TnT - a statistical part-of-speech tagger", In Proceedings of the Sixth Applied Natural Language Processing Conference ANLP, Seattle, WA, 2000.
15. Hassan Sajjad and Helmut Schmid, 2009. "Tagging Urdu Text with Parts of Speech: A Tagger Comparison", Proceedings of the 12th Conference of the European Chapter of the ACL, pp: 692-700.
16. Eric Brill, 1992. "A Simple Rule-Based Part of Speech Tagger", Proceedings of the DARPA and Natural Language Workshop, pp: 112-116.
17. Garnett Wilson and Malcolm Heywood, 2005. "Use of a genetic algorithm in brill's transformation-based part-of-speech tagger", Proceedings of the 2005 conference on Genetic and Evolutionary Computation, pp: 2067-2073.
18. Chenda Nou and Wataru Kameyama, 2007. "Khmer POS Tagger: A Transformation-based Approach with Hybrid Unknown Word Handling", Proceedings of the International Conference on Semantic Computing, pp: 482-489.

19. Shabib AlGahtani, William Black and John McNaught, 2009. “Arabic Part-Of-Speech Tagging Using Transformation-Based Learning”, University of Manchester, pp: 66-70.
20. Aslı Gülen and Esin Saka, 2001. “Part of speech tagging”, A Term Paper Submitted To Ceng463 Introduction To Natural Language Processing Course Of The Department Of Computer Engineering Of Middle East Technical University, pp: 1-23.
21. Beata Megyesi, 1998. "Brill’s Rule-Based Part of Speech Tagger for Hungarian" D-level thesis (Master’s thesis) in Computational Linguistics, Department of Linguistics, Stockholm University, Sweden,
22. http://www.crulp.org/software/ling_resources/UrduGrammar.htm.