# Tagger Voting for Urdu

Bushra Jawaid Ondřej Bojar
Charles University in Prague, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské nám. 25, Praha 1, CZ-118 00, Czech Republic
{jawaid,bojar}@ufal.mff.cuni.cz

Abstract
In this paper, we focus on improving part-of-speech (POS) tagging for Urdu by using existing tools and data for the language. In our experiments, we use Humayoun's morphological analyzer, the POS tagging module of an Urdu Shallow Parser and our own SVM Tool tagger trained on CRULP manually annotated data. We convert the output of the taggers to a common format and more importantly unify their tagsets. On an independent test set, our tagger outperforms the other tools by far. We gain some further improvement by implementing a voting strategy that allows us to consider not only our tagger but also include suggestions by the other tools. The final tagger reaches the accuracy of 87.98%.

Keywords: Urdu language, Parts-of-speech tagging, Tagger voting, Tagset unification.

*Proceedings of the 3rd Workshop on South and Southeast Asian Natural Language Processing (SANLP)*, pages 135–144,
COLING 2012, Mumbai, December 2012.

## 1 Introduction

Urdu belongs to the Indo-Aryan language family, a subclass of the Indo-European languages. Urdu is the official language of Pakistan and one of the 23 official languages (including English) spoken in India. It is the native language of at least 65.6 million speakers with another 40 million or more who speak it as a second language[1].

Urdu has borrowed its writing script from Persian, which is a modified form of the Arabic script, the Urdu script is thus called Perso-Arabic. Urdu is written from right to left with numbers written from left to right. The morphology of Urdu is similar to other Indo-European languages, e.g. by having concatenative inflective morphological system.

Urdu is a low-resource language with respect to even the core processing tasks like POS tagging or morphological analysis. Existing taggers for Urdu do not reach sufficient coverage and accuracy.

In this paper, we demonstrate how an ensemble of available tools and data can be joined to achieve a better performance. First, we convert the output of the existing morphological tools to a common representation and more importantly, we unify the different tagsets. Second, we train and evaluate a new tagger on the available annotated data (in our unified tagset) and finally, we implement and evaluate a "voting" scheme that combines the outputs of all available taggers.

## 2 Resources for Urdu morphology

In this section we briefly list existing morphological tools and POS tagged data for Urdu.

Apart from the available works, there are also some relevant research papers: Anwar et al. (2007a) developed POS tagger for Urdu based on Hidded Markov Models (HMM). They tried to combine several smoothing techniques with HMM model to reduce data sparseness problem. They achieved maximum of 96% accuracy using Good Turing smoothing method when trained on a 70K-token corpus by (Hardie, 2003). The size of the test data is not mentioned.

Anwar et al. (2007b) improve a simple unigram and bigram tagger for unknown and ambiguous words by considering word endings.

### 2.1 Tools

To the best of our knowledge, only two morphological analyzers for Urdu are freely available: Hussain (2004) and Humayoun (2006). The former is an implementation of a finite-state transducer whereas later is based on a functional morphology toolkit for morphology development in Haskell (Forsberg and Ranta, 2004). Because the tool published by Hussain (2004) requires Windows and its word coverage is rather low, we use only the tool by Humayoun (2006) and call it HUM analyzer in the following.

Language Technologies Research Center of IIIT Hyderabad has developed a shallow parser for Urdu[2] (called SH parser in the following) which analyses Urdu at various levels: tokenization, morphological analysis, POS tagging, chunking, etc. Our main interest is to get the morphologically disambiguated output, which can be extracted from the final output

[1] http://en.wikipedia.org/wiki/Urdu

[2] http://ltrc.iiit.ac.in/showfile.php?filename=downloads/shallow_parser.php

represented in Shakti Standard Format (SSF)[3]. Detailed description of SSF and a brief overview of tagset is available online[4].

## 2.2 Data

Besides the tools mentioned above, there is a freely available POS tagged corpus developed by CRULP (Center for Research in Urdu Language Processing)[5]. The underlying tagset is also available online[6]. The actual text of the corpus is a translation of portion of the Wall Street Journal's section of the Penn Treebank. We use this data to train our tagger, see Section 4.

Sajjad and Schmid (2009) manually tagged 110K tokens from a news corpus (www.jang.com.pk) but their data is not freely available[7]. In addition to 110K tokens, they also tagged 8K tokens[8] from BBC News (www.bbc.co.uk/urdu/) and made it freely available online[9]. Sajjad and Schmid (2009) have designed their own tagset[10]. that is purely syntactic in nature, we call this tagset *Sajjad's tagset*.

Note that Sajjad's tagset is more coarse-grained than the one used by CRULP. For instance, Sajjad tags proper noun by PN whereas CRULP distinguishes between the first proper noun in a multi-word name (NNP) and the following ones (NNPC - Proper Noun Continued). For instance, "Saudia Arabia" is tagged as "Saudia|NNP Arabia|NNPC" in CRULP data.

Table 1 summarizes the data we use in our experiments. We see that 1315 of the test tokens (about 15% of the test set) are never seen in the training data.

| | Train (CRULP) | Test (Sajjad) | Total |
|---|---|---|---|
| Sentences | 4320 | 404 | 4724 |
| Tokens | 123843 | 8670 | 132513 |
| Out-of-vocabulary | – | 1315 | – |

**Table 1:** Statistics of our training and test data.

# 3 Format and tagset unification

Due to the low coverage of existing morphological tools for Urdu, it is hard to get an entire corpus tagged with a reasonable accuracy using any of the available linguistic tools alone. To join the forces of the CRULP tagged data and the two analyzers, we need to unify their formats and tagsets.

## 3.1 Conversion of CRULP's data and output of tools to a common format

The first step in the conversion is a technical mapping of the file formats to a common one.

[3]SSF is a highly readable representation for storing language analysis.
[4]http://ltrc.iiit.ac.in/analyzer/urdu/shallow-parser-urd-3.0.fc8/doc/ssf-guide-4oct07.pdf
[5]http://www.crulp.org/software/ling_resources/UrduNepaliEnglishParallelCorpus.htm
[6]http://www.crulp.org/Downloads/ling_resources/parallelcorpus/Urdu%20POS%20Tagset.pdf
[7]They have used this data for training the taggers in their experiments.
[8]Small dataset is used for taggers evaluation in their work, we also use it for same purpose.
[9]http://www.ims.uni-stuttgart.de/~sajjad/resources.html
[10]http://www.cle.org.pk/Downloads/langproc/UrduPOStagger/UrduPOStagset.pdf

Table 2 illustrates the common format of the morphological analysis of word "cloths" by both tools. Starting from right, each position in a token represents: word, lemma and tag, separated by the "|" symbol. HUM analyzer, in most cases, provides an ambiguous morphological analysis for the word. All the possible tags are joined using the "-" sign. Each predicted tag has also a simple internal structure (reading left to right): the main part of speech is followed by revelant morphological features. The tag and morphological features are delimited with the "+" symbol.

We also convert the CRULP annotated data from the original format "<CD>اکسٹھ" to "CD|اکسٹھ".

| | HUM Analyzer | SH Parser |
|---|---|---|
| Output | N+NF+Pl+Nom+Masc\|کپڑے\|کپڑا<br>N+NF+Sg+Voc+Masc-<br>N+NF+Sg+Obl+Masc- | NN+n+m+pl++\|کپڑے\|کپڑا |

**Table 2:** The output of HUM Analyzer and SH Parser converted to the common format.

## 3.2 Unification of tagsets

The second step is the tagset unification. We selected Sajjad's tagset as the target one because it fits our long-term goal of improving English-Urdu phrase-based translation. We manually map each of the tagsets (CRULP data, HUM analyzer and SH parser outputs) to the Sajjad's tagset. The mapping is shown in Table 3.

The symbol "—" indicates that there is no tag in the given tagset that would correspond to the Sajjad's one. In some cases, the Sajjad's tagset is less detailed and we map several tags to a single one, e.g. RB and I in CRULP tagset map to ADV.

When designing the mapping rules, we considered available documentation and also data tagged with the tagset.

## 3.3 Mapping HUM analyzer and SH parser outputs

Before mapping test set tagged by HUM analyzer and SH parser on Sajjad's tagset, we drop all the morphological information and preserve only the set of proposed POS tags.

A sample test sentence tagged by HUM analyzer and SH parser is shown in Table 4. Again, "|" delimits words from their tags. The mapping to Sajjad's tagset can again introduce ambiguity. We delimit ambiguous tags with "-".

# 4 Our tagger

Giménez and Màrquez (2004) introduced and made publicly available a multi-purpose tagger called SVM Tool. SVM Tool performed better than state-of-the-art taggers and Sajjad and Schmid (2009) confirmed this for Urdu.

We follow up on this work and train SVM Tool on CRULP manually tagged data (Section 2.2). SVM Tool offers five different kind of models for training a learner. We use 'model 4' with tagging direction from right-to-left. Model 4 boosts identification of unkown words during the learning time by artificially marking some of the words as unknown

| Sajjad's Tagset | CRULP | HUM Analyzer | SH Parser |
|---|---|---|---|
| A | JJRP | PostP, Part | PSP |
| AA | AUXA | — | VAUX |
| AD | — | RelPron2 | DEM |
| ADJ | JJ | Adj, Adj1, Adj2, Adj3, AdjD | JJ, XC |
| ADV | RB, I | Adv | RB, INTF, NST |
| AKP | — | InterPron1, InterPron2, InterPron3 | — |
| AP | — | RelPron2 | NST |
| CA | CD | Num | QC, ECH |
| CC | CC | Conj | CC |
| DATE | DATE | — | — |
| EXP | SYM | — | SYM |
| FR | FR | — | — |
| G | PRP$ | PossPron | PRP |
| GR | PRRFP$ | — | PRP |
| I | ITRP | Part | RP |
| INT | INJ | Intjunc | JJ |
| KD | — | InterPron | WQ |
| KER | KER | PossPostPos | PSP |
| KP | — | InterPron | WQ |
| MUL | MUL | Verb, Verb1 | VM |
| NEG | NEG | Neg | NEG |
| NN | NN, NNCM, NNC, NNCR, MOPE, MOPO, NNL | N | NN, XC |
| OR | OD | RelPron2, N | QO |
| P | CM | PossPostPos | — |
| PD | DM | DemPron | PRP |
| PM | PM | — | SYM |
| PN | NNP, NNPC | PN | NNP, XC |
| PP | PR | PersPron, RelPron1 | DEM |
| Q | Q | IndefPron1, IndefPron2, RelPron2, IndefPron, RelPron3 | QF |
| QW | QW | Quest | WQ |
| RD | DMRL | RelPron | — |
| REP | PRRL | RelPron | PRP |
| RP | PRRF | RefPron | PRP |
| SC | SC | Conj | CC |
| SE | SE, RBRP | PostP | PSP |
| SM | SM | — | SYM |
| TA | AUXT | — | VAUX |
| U | U | — | — |
| UNK | UNK | UNK, Verb3, Verb_Aux | UNK |
| VB | VB, VBL, VBI, VBLI, VBT | Verb, Verb1, Verb2 | VM |
| WALA | WALA | — | — |

**Table 3:** Tagset mapping of Humayoun Morphological Analyzer, Urdu Shallow Parser and CRULP tagset to a common Sajjad's tagset.

words. This feature enhances the capabilities of the learning model and makes it more realistic and refine. Sajjad and Schmid (2009) used the similar model in their work.

Remember that we want to evaluate our tagger using Sajjad's tagset and the mapping described in Section 3.2. This gives rise to two approaches: either we could train the tagger on CRULP manually tagged data and map its output to Sajjad's tagset afterwards, or we could map the training data from CRULP to Sajjad's tagset and train the tagger on this modified training data. We opted for the latter approach, because a deterministic mapping after a statistical classifier in general increases the risk of error cumulation.

We measure the accuracy of the tagger as the percentage of correctly tagged tokens of all the tokens in the Sajjad's test set, see Section 2.2.

With default settings, our SVM Tool tagger achieves the accuracy of only 63.71%. After

| | |
|---|---|
| HUM Before Mapping | UNK\|۔ Verb_Aux\|تھا Verb\|ہوا Verb1\|پڑھا N\|جماعت Num\|N\|سات UNK\|چھ |
| HUM After Mapping | UNK\|۔ UNK\|تھا VB-MUL\|ہوا VB-MUL\|پڑھا NN-OR\|جماعت NN-OR-CA\|سات UNK\|چھ |
| SH Before Mapping | SYM\|۔ VAux\|تھا VAUX\|ہوا VM\|پڑھا NN\|جماعت ECH\|سات QC\|چھ |
| SH After Mapping | SM-PM-EXP\|۔ AA-TA\|تھا AA-TA\|ہوا VB-MUL\|پڑھا NN\|جماعت CA\|سات CA\|چھ |

**Table 4:** Mapping HUM Analyzer and SH Parser outputs to the Sajjad's tagset

analyzing the output, we found a few types of errors caused by ambiguous mappings, i.e. when a tag X in CRULP tagset maps to tags Y and Z in Sajjad's tagset. To reduce the risk of wrong or ambiguous mappings, we modified training data prior to training the tagger as follows:

1. We initially mapped PR (Pronoun) to PP (Personal Pronoun) and KP (Kaf Pronoun) and DM (Demonstratives) to PD (Personal Demonstratives) and KD (Kaf Demonstratives). This mapping introduced substantial ambiguities in the training data apparently due to the large number of occurrences of pronouns and demonstratives. To overcome this problem, PM and DM mappings to the Sajjad's tags KP and KD, respectively are removed from the mapping table and we introduce them directly to the training data. All words starting with the the character "ک" (Kaf) get the tag KP (Kaf pronoun) if they were annotated as PR in the CRULP original annotation and the tag KD (Kaf demonstrative) if the original annotation was DM.

2. In the training data, the word "سے" is annotated with the tag "CM" if it is used as a semantic marker. However, in Sajjad's tagset, a special tag "SE" should be used for the word, if it has the characteristics of a semantic marker. To avoid the ambiguous mapping of CM to SE and P (the tag used for marking other semantic markers in Sajjad's tagset) for this word, we remove the mapping CM→SE from the table and apply it to the training data directly: if "سے" is annotated with CM, it gets the Sajjad's tag SE.

3. Negation markers "نہیں" and "نہ" are annotated using RB (Adverb) in the training data. However, in Sajjad's tagset, "NEG" is used to annotate negation markers. Again, we remove the general mapping of RB to NEG and mark these negation markers as NEG explicitly.

Table 5 lists our gradual improvement of accuracy. The column "MOD" shows the accuracy after the above refinement of ambiguous tags, 20% absolute higher than the baseline. On top of that, we created a list of some closed-class words with a fixed tag. The words in the list receive the tag from the list, not from the mapping. Similarly hard-coded list is later added for cardinals, reaching the the accuracy to 85.75%. The "*Best*" tagger uses all the previous modifications and also adds new features for SVM tool: the prefixes of 1 and 2 characters for the current token, the suffixes of 1, 2, 3 and 4 characters of the current token and also the bigram and trigram of preceding word forms and preceding tags. We use only the "Best" tagger in the following experiments.

| | Baseline | MOD | CCW | CD | Best |
|---|---|---|---|---|---|
| Accuracy | 63.71% | 84.48% | 85.40% | 85.75% | **86.18%** |

**Table 5:** Gradual improvements of the baseline tagger: modified training data (MOD), closed-class words (CCW), cardinals (CD), and two new features for the SVM Tool (Best)

In Table 6 we have shown the output of our best accuracy tagger with the reference sentence from the test set.

| | |
|---|---|
| Tagger output | SM\|۔ VB\|ہے NN\|ذیل ADJ\|درج NN\|تفصیل P\|کی REP\|جس |
| Reference | SM\|۔ VB\|ہے NN\|ذیل NN\|درج NN\|تفصیل P\|کی REP\|جس |

**Table 6:** Test set sentence tagged by the best accuracy SVM Tool tagger.

## 5 Tagger voting

In this section, we evaluate the individual performance of each of the taggers[11]. Then we describe our voting strategy, the resolution of cases where more than one tag get the same score in the voting, and also evaluate the examined configurations of tagger voting.

### 5.1 Performance of individual taggers

Due to the unavailability of a hand-tagged data using original tagset of each tagger, we originally wanted to measure the accuracy after mapping the output of the taggers on Sajjad's tagset. The mapping can however lead to ambiguity, e.g. the tag PRP by SH parser corresponds to G, GR, PD, REP or RP in Sajjad's tagset, see Table 3. To avoid the need of manual disambiguation of these cases, we collapse all such ambiguities to a common tag for the purposes of this section. The resulting 'coarse tagset' is much less informative than all other ones and would not be very useful in practice. It has only 15 tags instead of the 40 in Sajjad's tagset and lumps e.g. nouns, proper nouns, adjectives and adverbs into one tag, AD-PP-AP-Q-OR-NN-PN-ADJ-INT-ADV.

We map Sajjad's test set and the output of the taggers to the coarse tagset and calculate the accuracy of each of the taggers, see Table 7. HUM analyzer and SH parser sometimes tag a word as unknown (UNK). We see that unknown words amount to one third of the test set for HUM analyzer. Our SVM tagger performs best, reaching 97%. Remember though, that the accuracies in Table 7 are based on the very coarse tagset and they are not comparable with accuracies reported in other tables.

| | HUM Analyzer | SH Parser | SVM Tagger |
|---|---|---|---|
| Accuracy (Coarse Tagset) | 45.49% | 81.51% | 97.02% |
| UNK Tokens | 2898 (33.4%) | 233 (2.68%) | 0 (0%) |

**Table 7:** Taggers accuracies and UNK (unknown) tokens count as observed on Sajjad's test set using a comparable but very coarse-grained tagset.

### 5.2 Voting

As shown in Table 7, the accuracy of the HUM analyzer and SH parser appears to be surprisingly low even in the coarse tagset. Still, we believe these tools could contribute and propose a simple voting scheme to merge the suggestions from all the three taggers.

Our voting strategy implements the conventional voting style: each tagger has the power of 1 vote. If the tagger emits more than one tags for a token, this one vote is split uniformly (except SVM tagger, see below) among all the suggested tags. Votes for the unknown tag (UNK) are discarded and the tag that receives the highest sum of votes is selected. In case of a voting conflict, i.e. two or more tags receive the same score, we keep them all and

[11] We call HUM analyzer, SH parser and our tagger based on SVM Tool simply "taggers" in the following.

resolve the ambiguity later in Section 5.3. Table 8 illustrates a test sentence before and after voting. Tags in bold are the winners of the voting.

| | |
|---|---|
| HUM A. | UNK\|۔ UNK\|تھا UNK\|پڑا **VB-MUL**\|الٹا **I**-A\|ہی **PP**\|ویسا KER-**P**\|کا **PP**\|ویسے UNK\|ٹرک |
| SH P. | AA-TA\|پڑا **VB-MUL**\|الٹا **I**\|ہی NN\|ویسا A-KER-SE\|کا NN\|ویسے **NN**-PN-ADJ\|ٹرک **SM**-PM-EXP\|۔ AA-TA\|تھا |
| SVM T. | **SM**\|۔ **VB**\|تھا **VB**\|پڑا **PN**\|الٹا **I**\|ہی **ADV**\|ویسا **P**\|کا **ADJ**\|ویسے **NN**\|ٹرک |
| After Voting | SM\|۔ VB\|تھا VB\|پڑا PN-MUL-VB\|الٹا I\|ہی ADV-PP\|ویسا P\|کا PP-ADJ\|ویسے NN\|ٹرک |

**Table 8:** Output of HUM analyzer, SH parser and SVM tagger before and after voting.

By default, SVM tagger returns its single-best suggestion and always has the power of 1 vote, taking precedence over the other tools too often. To facilitate a smoother merge, we also consider taking more than just the single-best scoring tag from SVM based on its internal probabilities assigned to individual tag options. Taking all the options would not work either, because SVM returns on average more than 5 candidates which would make its votes to these tags too weak.

We thus resort to a fixed number (one, two or three) of considered tags from SVM tagger and we normalize probabilities of these tags to sum to 1, the total power of SVM's vote.

We apply one more hack to tackle the unreliability of SH parser when tagging nouns. Whenever the SH parser proposes NN among the set of suggested tags, we cut its vote for NN by half. An example of this is in Table 8 where the word ویسا received only the tags ADV-PP, despite SH parser was suggesting unambiguous NN.

## 5.3 Resolving outstanding ambiguities

As seen in Table 8, the word "زیادہ" had more tags reaching the highest score. We use two approaches to resolve such remaining ambiguities: either we use a static preference list, or a list based on the overall frequency of the tags in the training. Of the ambiguous tags, we pick the one that appears highest in the given list.

Table 9 illustrates the sentence from Table 8 with preference-based or frequency-based resolution as well as the reference annotation.

| | |
|---|---|
| Preference-based | SM\|۔ VB\|تھا VB\|پڑا VB\|الٹا I\|ہی ADV\|ویسا P\|کا ADJ\|ویسے NN\|ٹرک |
| Frequency-based | SM\|۔ VB\|تھا VB\|پڑا VB\|الٹا I\|ہی PP\|ویسا P\|کا ADJ\|ویسے NN\|ٹرک |
| Reference | SM\|۔ TA\|تھا VB\|پڑا ADJ\|الٹا I\|ہی ADV\|ویسا P\|کا ADV\|ویسے NN\|ٹرک |

**Table 9:** Voted sentence from Table 8 after applying different fall back options.

## 5.4 Evaluation

We establish SVM tagger's (individual) accuracy as the baseline for our voting experiments. Table 10 provides the results. SVM-Tag-1, 2, and 3 are our voting setups where we used the top 1, 2, or 3 options from SVM before normalizing their probabilities to sum to the one vote of SVM. The final ambiguity resolution strategy is indicated in the column label.

"Voted Only" means that the ambiguous final output is produced, which has no chance to score well in comparison with the fully disambiguated test set.

A preliminary analysis of SVM Tag-2 and 3 voted output revealed that we make errors in cases where SVM tagger predicts only one tag (so this tag gets the vote of 1) but it is still not selected because it is considered less probable by the remaining two taggers. For such cases, i.e. when SVM had the chance to express its uncertainty but still decided unambiguously, we give it a preference. As indicated in the lines labeled "SVM Preferred If Sure", this gives again a little improvement.

| | | Voted Only | Voted & Static Fall Back | Voted & Freq. Based Fall Back |
|---|---|---|---|---|
| Baseline | – | | 86.18% | |
| SVM Tag-1 | – | 85.15% | 86.93% | 86.85% |
| SVM Tag-2 | Default | 86.48% | 87.72% | 87.64% |
| | SVM Preferred If Sure | 87.65% | 87.76% | 87.72% |
| SVM Tag-3 | Default | 86.66% | 87.94% | 87.84% |
| | SVM Preferred If Sure | 87.84% | **87.98%** | 87.68% |

**Table 10:** Accuracy of test corpus after Voting and applying fall back option.

## 6 Conclusions and future work

This paper investigated available data and tools for Urdu POS tagging. We unified their respective tagsets and trained our own tagger on the available training data. A comparison on an independent test set documented that our tagger clearly outperforms the other tools.

Additionally, we devised a simple voting scheme and obtained improvement by considering the suggestions of other taggers. The combined tagger reaches the accuracy of 87.98%.

In future, we would like to refine the voting strategy, making it more context-dependent, e.g. by adding one more custom tagger trained to pick the best tag. Also, we are aware that the current ensemble of taggers is somewhat impractical: three taggers have to be run and the final answer is available only after their voting. We plan to run this complex ensemble on a large monolingual corpus and use this data to train a single, standalone tagger. We also plan to release the standalone tagger.

As a separate future goal, we would like to add back the detailed morphological information we are now stripping off.

## Acknowledgments

We thank Hassan Sajjad for discussions on characterstics of different taggers and also, Muhammad Humayoun for helpling us with the addition of extra lexicon in his tool, which unfortunately we couldn't add due to time constraints and manual labour involved in it.

This work has been using language resources developed and/or stored and/or distributed by the LINDAT-Clarin project of the Ministry of Education of the Czech Republic (project LM2010013) and it was also partially supported by the grant P406/10/P259 of the Czech Science Foundation.

## References


Anwar, W., Wang, X., Lu-Li, and Wang, X.-l. (2007a). Hidden markov model based part of speech tagger for urdu. pages 1190–1198.

Anwar, W., Wang, X., Lu-Li, and Wang, X.-l. (2007b). Morphological ending – based strategies of unknown word estimation for statistical pos urdu tagger. pages 167–173.

Forsberg, M. and Ranta, A. (2004). Functional morphology. In *Proceedings of the Ninth ACM SIGPLAN International Conference on Functional Programming*, pages 213–223. ACM Press.

Giménez, J. and Màrquez, L. (2004). Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of the 4th LREC*, Lisbon, Portugal.

Hardie, A. (2003). Developing a tagset for automated part-of-speech tagging in urdu. Department of Linguistics, Lancaster University.

Humayoun, M. (2006). Urdu morphology, orthography and lexicon extraction. In *Master's Thesis*. Department of Computing Science, Chalmers University of Technology.

Hussain, S. (2004). Finite-state morphological analyzer for urdu. In *Master's Thesis*. National University of Computer & Emerging Sciences.

Sajjad, H. and Schmid, H. (2009). Tagging urdu text with parts of speech: a tagger comparison. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 692–700, Stroudsburg, PA, USA. Association for Computational Linguistics.