Challenges in Urdu Stemming (A Progress Report)

Kashif Riaz
University of Minnesota
4-192 EE/CS Building
200 Union Street SE
Minneapolis, MN 55455
riaz@cs.umn.edu

Abstract: This paper explains the challenges pertaining to Urdu stemming and presents a rule-based prototype with a few rules implemented for Urdu to motivate the intricacies. It shows that Urdu stemming is quite challenging because of Urdu's diverse nature and because Arabic and Farsi stemmers cannot be used for Urdu. Dictionary-based error-correcting schemes used by other stemmers cannot be applied to Urdu because of the lack of machine-readable resources. There has not been any work published regarding Urdu stemming or morphological analysis in the IR community even though interest in Urdu is growing. The goal of this paper is to show the challenges in writing an Urdu stemmer, not to present a stemmer.

Keywords: Urdu, stemming, multilingual IR

1. INTRODUCTION

A *stem* in linguistics is the combination of the basic form of a word (called the root) plus any derivational morphemes, but excluding inflectional elements. Some researches make a distinction between the concepts of *stem* and *root* but the difference has no practical value for stemming purposes. Alternatively, a stem is the form of the word to which inflectional morphemes can be added [15].

A *morpheme* is the smallest language unit that carries a semantic interpretation. Morphemes are generally a distinctive collocation of phonemes (the free form *pin* or the bound form -s of *pins*) having no smaller meaningful members. *Inflection* refers to modification of a word. More precisely, a word is modified so that it reflects grammatical information such as word gender, tense, or person. The root of the English verb form *destabilized* is stabil- (alternate form of *stable*); the stem is de•stabil•ize, which includes the derivational affixes de- and -ize, but not the inflectional past tense suffix -(e)d.

In languages with very little inflection such as English and Mandarin Chinese, the stem is usually not distinct from the "normal" form of the word. However, in other languages, stems are more noticeable. For example, the English verb stem *eat* is indistinguishable from its present tense (except in the third person singular).

2. STEMMER

A stemmer is a computer algorithm to reduce the words to their stem, base, or root form. A stemming algorithm has been a long-standing problem in Information Retrieval. The process of stemming, often called *conflation*, is useful in search engines, natural language processing, and other word processing problems. For example, a stemming algorithm reduces the words *fishing, fished, fish,* and *fisher* to the root word *fish.*

Stemming is mostly used to increase the efficiency of a search engine—more specifically, to increase recall to a large degree. Precision is sometimes increased or decreased depending upon the information need of the user. Earlier stemmers were rule-based written in BNF notation. The first paper on the subject was published in 1968 by Julie Lovins who listed about 260 rules for stemming the English language. She used Iterative Longest Match heuristic, which means that a rule is preferred over the other rule when its right side matches the most characters. The most notable work is presented by Martin Porter in 1980 [14] who simplified the rules of Lovin to about 60 rules. The algorithm attributed to this work is called Porter Stemmer and is most widely used in search engines.

Stemming can be classified as weak stemming and strong stemming. Weak stemming entails executing a few selected rules (e.g., one can just decide to use the rules that deal with plurals in English). The use of rule-based stemmers is very helpful because upon encountering new words, one can change the grammar rule instead of trying to solve the problem by ad hoc approaches [4]. Stemmers are common elements in query systems because a user who runs a query on cars probably also cares about documents that contain the word car (without the s).

3. STEMMING PROBLEMS

The fundamental problem with any stemming technique is that the morphological features being stripped away may well obscure differences in the word's meaning. For example, the word gravity has two word-senses, one describing an attractive force between any two masses and the other having to do with a serious mood. But once the word gravitation has been stemmed, we have lost the information that might constrain us to the first interpretation, this means that conflation has happened [4]. Porter [15] describes the problems of under-stemming, over-stemming and mis-stemming based on the suffix-stripping algorithm known as the Porter Stemmer [14]. Porter [15] mentions three classes of suffixes:

Attached: a particle word attached to another word
Inflected: part of the grammar to indicate the tenses and other aspects of the grammar e.g. telephone → telephoned
Derivation: forms a new word, often with a different grammatical category or a different word sense e.g. clean → cleanliness

Under-stemming happens when the stemming algorithms leaves the suffix attached to the word. Over-stemming happens when the algorithm removes too much of the suffix. Mis-stemming is removing suffix which was actually part of the word. Porter [15] suggests that over-stemming and mis-stemming can be corrected by the use of dictionaries.

3.1 Language Dependency

English stemmers are fairly trivial with only occasional problems in the retrieval performance such as axes being the plural of ax as well as axis. However, stemmers become harder to design as the morphology, orthography, and character encoding of the target language becomes more complex, mostly it is the inflection of the language. For example, an Italian stemmer is more complex than an English one (because of more possible verb inflections), a Russian one is even more complex, an Arabic is more complex than all of the above because of orthography and high inflection. For example, for plurals, Arabic has a word for singular, two-count plural, more than two-count plural, and it has infixes.

4. RELATED WORK

Although, there is no published work in the IR community describing challenges in Urdu stemming, there has been considerable work done towards computational morphological anaysis of Urdu. Butt [9] and Rizvi [2] describe computational analysis of different parts of speech in Urdu. Their work is very important for computational Urdu processing, but is focused on theoretical computational linguistics. Their work may certainly be used to build the rules for an Urdu stemmer. A morphological analyzer is available at Computing Research Laboratories (CRL) [7, 8]. After a brief analysis of the results produced by the CRL morphological analyzer and after examining the root forms, a number of false positives and errors were observed. We theorize that this is because the CRL software used the same rules for Urdu as for Arabic and Farsi.

Savoy [5] uses a dictionary-based approach to stem French words in a corpus. This work is quite interesting but unfortunately there are no machine-readable Urdu dictionaries available to see the utility of this approach with Urdu.

There has been significant work done on Arabic stemmers—most of it is statistical and heuristics-based [6, 10]. Although Farsi and Arabic are written with *similar* (not the same) scripts, stemmers for those languages are not adequate for Urdu stemming. Arabic stemmers produce a large number of over-stemming and mis-stemming errors for Urdu because of Arabic's high inflection and complex grammar. Farsi stemmers produce a number of incorrect stems because it accurately stems only the Farsi loan words and errors on native Urdu and Arabic loan words.

5. URDU: A CHALLENGING LANGUAGE FOR STEMMING

Urdu is a very rich language that borrows words from other languages readily. It is estimated there are about 60 million people who speak Urdu as their mother tongue, and one hundred and four million who consider Urdu as their second language [3]. Urdu has an interesting characteristic—is a composition of many languages and adopts words from other languages with ease. Besides having its own morphology, Urdu morphology is strongly influenced by Farsi (Persian), Arabic, and Turkish. Therefore, Urdu vocabulary is composed of the above mentioned languages along with many Sanskrit-based and English words. For example, the word *pachim* (Hindi) and *maghrib* (Arabic) both mean the direction west in English and are both Urdu words as well.

Urdu is a bi-directional language with an Arabic-based orthography. Bi-directional means that it is very common in Urdu to see an English word written in Latin-based characters. Sometimes an English word is written phonetically with Urdu characters (e.g. *pub* is written as پب). Although Urdu has Arabic orthography, its grammar is based on Sanskrit and Persian. Urdu has gender marking on its parts of speech (e.g. *paharh* (mountain) and *paharhi* (hill). Therefore, stemming Urdu words will increase recall and also conserve on space usage of the indices.

There are two ways to build a stemmer, rule-based (or linguistic-based) and statistical-based. Rule-based stemmers use prior linguistic knowledge or the morphology of the language to form stemming rules. In contrast, statistics-based approaches use statistical principles to infer the word formation rules (morphology) by analyzing a corpus. One general statistical stemming approach uses Bayes theorem to figure out the probabilities of the suffix given the word and then uses popular link based Hits algorithm to assign scores to the stems [13]. In this paper only the rule-based approach is explored.

Some issues that need to be addressed while writing the stemmer are listed below. The issues can be divided into two major sections. One is the morphological analysis of Urdu and the other is engineering approach (e.g., the operating system, choice of the programming language, IDE, the text editors, performance of the algorithm). The implementation of the stemming algorithm is orthogonal to the engineering issues. Since this research is work in progress, not all issues pertaining to Urdu stemming have been discovered yet.

5.2 Morphological Analysis

First we will address the morphological analysis. Some of the terminology is borrowed from Schmidt [1].

- Nouns: Most of the issues regarding nouns pertain to gender marking—determining if the noun is masculine or feminine and if it is marked or unmarked (e.g., marked noun *lardka* (boy) and unmarked noun جہازي *jahazi* (sailor)).
- Arabic loan words ending with "-at" (e.g., قیمت qeemat (price))
- Persian loan words: One needs to analyze the suffixes of the Persian loan words to determine if the noun is masculine and feminine (e.g. the suffix "-stan" for Pakistan is masculine where as the suffix "-i" for *dosti* (friendship) is feminine).
- Indigenous words: Marking on indigenous words can be found by determining if it is a regular noun, personal noun or if the suffix is diminutive suffix or a noun form suffix.

- Noun plurals: There are different plural suffixes for masculine marked nouns (e.g., *lardka* (boy) → *lardkey* (boys)), feminine marked nouns (e.g. *lardki* (girl) → *lardkian* (girls)) and feminine unmarked nouns(*e.g., kitabein* (books) → *kitabein* (books)). Masculine unmarked nouns have no plural suffixes (e.g., *ghar* (home) → *ghar* (homes)).
- Pronouns: Is our heuristic correct that we don't process pronouns because their length is not more than four?
- Adjectives: change in marked adjectives for agreement, for gender (masculine/ feminine), number (singular/plural), and case marking
- Postpositions: Is our heuristic correct that we don't process postpositions because their length is less than four?
- Verbs: Stem according to different forms for verbs: root form, imperfect participle, and perfective participle.
- Exception list for number, dates, months etc.
- Persian elements in Urdu: use of *izafat* (possessive marker) and word forming affixes
- Arabic elements in Urdu: The trilateral root is one the basic structures of the Arabic root; Arabic definite article *al*
- Use of Dictionaries: Porter [15] claims that over-stemming and mis-stemming errors can be corrected by using an online dictionary. There is no machine readable Urdu dictionary available to the community. Therefore, another method needs to be devised to remedy these errors.

5.2 Engineering Issues

Some of the language engineering issues are listed below:

- Unicode: One needs to be very familiar with Unicode—in this case, the Arabic orthography Unicode standards.
- Programming language: Not all programming languages support Unicode and bi-directional languages. Java and C# are the two most widely used ones that do.
- Operating System: One needs to use an operating system that has Urdu support. Windows XP™ has this support built in. The support on Linux is not available for all versions.
- For quick prototyping of some concepts, it is useful to have the debugging mechanism available. Although a programming language may support Unicode, the IDE might not support it completely. For example, Eclipse™ does not support Unicode in its display console.
- One needs a text editor to view Urdu characters that is scalable and supports Unicode. UltraEdit™ is one such editor.

6. PROTOTYPE

A prototype of an Urdu stemmer is implemented that incorporates four rules for processing plurals and possessives and a heuristic for skipping a word so it does not get stemmed. It was observed that the order in which the rules are executed is important, and changing the order improved the results considerably. The prototype is implemented in Java on a Windows XP machine. The possessive rule is really a gender-marking rule for feminine marking—it addresses both the noun marking and the adjective feminine marking. The plural rules do not completely encapsulate all the morphological variations. The plural rules are for indigenous Urdu words, and there is one rule for plural in Persian. One very interesting phenomenon was observed while implementing the plural rule. There are some root forms of Urdu that are shared amongst the indigenous form, Persian form, and Arabic form, and the word has to be processed by each of the rules (not one of the rules) to reach the proper root form. The side effect of this process is that recall is increased, but precision is decreased considerably because polysemy is introduced. The following example can illustrate the phenomenon.

If we try to derive the root form of the Urdu word *khabr* خبر (news), then it could arrive from many different source words with different meanings. The possible non-stemmed word could be a loan word from another language or indigenous to Urdu as illustrated below:

اخبار *akhbar*: *newspaper* in Urdu and Persian, but plural for *news* in Arabic
خبر *khabr*: *news* in Urdu, Arabic, and Persian – root form
خبریں *khabrain*: plural of *news* in indigenous Urdu and Hindi
اخبارات *akhbarat*: plural of *newspaper* in Urdu and Persian

The results of the prototype are encouraging, but far from complete. The stemmer was run on 1065 words taken randomly from the Becker-Riaz Urdu Corpus [11], and the four rules were applied on each word only once. Out of 1065 words, 569 were treated by the non-stemming heuristic and four rules for stemming. Out of 569 words, 211 were stemmed, and, out of 211 words stemmed, 32 were stemmed incorrectly. The reason for the false hits was unimplemented rules—words that matched the Persian profile but were adjectives in Arabic. Additionally, some transliterated words of English like ائرمارشل (Air Marshall) and اسکواڈرن (squadron) were found in the list. The algorithm correctly reduced squadron to سکوڈ (squad), but reduced Air Marshall to ئرمر *aimr* (a non-word) because it mistakenly assumed that it was the Persian plural.

6.1 Stemming Evaluation

There are many ways to do an evaluation of a stemming algorithm. One of the most basic methods is called the *direct assessment method* in which performance of the stemmer is judged by examining its behaviour when the algorithm in applied on a sample word [16]. The utility of this method in isolation is not quite clear. It is typically used with other complementary techniques. We used the direct assessment method to analyze the challenges for Urdu stemming approaches. There are two other popular techniques for the evaluation of stemming algorithms. One is based in the discipline of Information Retrieval where recall and precision measures are used to judge performance; an increase in recall indicates a good stemmer. An error counting approach for the evaluation of a stemming algorithm is quite robust and desirable when used along with the direct assessment method. An error counting scheme counts the over-stemming and under-stemming errors by using a sample of grouped words [16].

7. CONCLUSION AND FUTURE WORK

Working with Urdu's stemming intricacies brings a myriad of challenges. The prototype was written to motivate the challenges associated this task. Future work entails enriching the rules by doing morphological analysis of the terms in the Becker-Riaz Urdu Corpus [11] and the EMILLE Urdu corpus [12]. We would like to evaluate the stemming framework, Snowball, written by Martin Porter to see if it is suitable for Urdu. Snowball is a stemming framework that presents a language to represent stemming algorithms. The stemming programs can be generated into various programming languages. The language and the approach seem quite promising. We would like to study stemming algorithms written for Hindi to see if they would apply to Urdu. We would also like to see if statistical stemming approaches are suitable for Urdu [15].

This Urdu stemming exercise is part of a larger goal to build a *robust* Urdu search engine where Urdu documents could be searched either by issuing Urdu queries or English queries. We already have a simple search engine that has indexed 2.5 million Urdu words from about 6000 Urdu documents from the Becker-Riaz Urdu Corpus. We would like to evaluate our stemmer using this search engine to measure recall and precision.

REFERENCES

[1] Schmidt, R. (1999) *Urdu: An Essential Grammar.* Routledge Publishing, New York.
[2] Rizvi, J et. al. (2005) *Modelling case marking system of Urdu-Hindi languages by using semantic information.* Proceedings of Natural Language Processing and Knowledge Engineering.
[3] http://www.crulp.org/English%20Site/..%5CPublication%5CCrulp_report%5CCR03_05E.pdf (visited: July 2007)
[4] Belew, R. (2000) *Finding Out About.* Cambridge Press, MA.

[5] Savoy, J. (1993) *Stemming of French words based on grammatical categories.* Journal of the American Society for Information Science 44(1), 1-9.
[6] Chen, A. Gey, F. (2002) *Building and Arabic Stemmer for Information Retrieval.* Proceedings of the Text Retrieval Conference.
[7] http://crl.nmsu.edu/Resources/lang_res/Text_Pre-processor_MCCS-04-332-v1.pdf (visited: July 2007)
[8] http://crl.nmsu.edu/Resources/lang_res/urdu.html (visited: July 2007)
[9] Butt, M. King, T. (2001) *Non-Nominative Subjects in Urdu: A Computational Analysis.* Proceedings of the International Symposium on Non-nominative Subjects, Tokyo, December, pp 525-548.
[10] Mokhtaripour, A., Jahanpour, S (2006) *Introduction to a New Farsi Stemmer.* Proceedings of CIKM Arlington VA, USA.
[11] Becker, D., Riaz, K. (2002) *A Study in Urdu Corpus Construction.* Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization at the 19th International Conference on Computational Linguistics. Taipei, Taiwan.
[12] Baker, P., Hardie, A., McEnery, T., Jayaram, B.D. (2003) *Corpus Data for South Asian Language Processing.* Proceedings of the 10th Annual Workshop for South Asian Language Processing, EACL.
[13] Goldsmith J., Higgins, D., Soglasnova, S. (2001) *Automatic Language-Specific Stemming in Information Retrieval.* Lecture Notes in Computer Science, Springer vol. 2069 pp 273-284.
[14] Porter, M. (1980) *An algorithm for suffix stripping.* Program vol. 14: pp 130-137
[15] Porter, M. (2001) *Snowball: A language for stemming algorithms.* https://snowball.tartarus.org/texts/introduction.html (visited: July 2007)
[16] http://www.comp.lancs.ac.uk/computing/research/stemming/Links/evaluation.htm (visited: July 2007)