MTR080256
MITRE TECHNICAL REPORT

ARGUS
Search and Stemming

Dr. Karine Megerdoomian

September 2008

MTR080256
MITRE TECHNICAL REPORT

MITRE

ARGUS
Search and Stemming

Karine Megerdoomian

September 2008

Sponsor: OSC
Dept. No.: G152
Contract No.: 2006*X005900*012
Project No.: 1508ME12

Abstract

Information retrieval systems often use stemming, i.e., the mechanical cutting off of inflectional and derivational affixes, to better match index terms to query terms. Stemming results differ, however, based on the affix list used, the depth of linguistic analysis applied, and the complexity of the language being analyzed. This report provides an overview of stemming, including its advantages and disadvantages for search applications and reviews the query syntax used in the Argus retrieval system. The report also presents guidance for analysts in performinig searches in Newsstand, with specific examples from several languages. In addition, the behavior of two specific systems – the stemmers for Arabic and Farsi (Persian) – are discussed in more detail. This report is intended for analysts performing search queries in Argus.

Table of Contents

List of Figures

List of Tables

1 Introduction to Stemming

Stemming is used to reduce words to their ***stem*** forms by cutting off the affixes. This process is often used in search engines and in Information Retrieval (IR) applications where the terms of the ***query*** (or search string) are compared with the ***index terms*** (important words or phrases appearing in the document that are also stemmed). If a match is found, then the document will be retrieved. For example, if the query in English contains the word *fishing*, the stemmer will reduce it to the form *fish* and will then return all documents containing the forms *fishing, fish*, *fished*, *fishes* and probably also *fisher*. Hence, stemming is generally accomplished by removing a set of prefixes and suffixes[1] from the word, such as '-ed' (for past tense of verbs) or '-es' (third person for present tense of verbs) and conflating the variant forms to a single term.

However, all stemmers are not created equal, and they can vary based on the sets of affixes (i.e., prefixes and suffixes) that they remove and also on how much linguistic knowledge they apply. In addition, languages also differ as to the level of difficulty they present for stemmers depending on their morphological or word-form complexity.

1.1 What is a Stem?

Stem: the part of a word that is common to all its inflected variants, where inflected words are the forms of the word that have affixes. In other words, if you take out the prefixes and suffixes, you are left with the stem.

With the example of *fishing*, once the '–ing' suffix is removed we obtain *fish* as the stem. But the stem doesn't always coincide with the base or dictionary form of the word. For example, if '–ing' is removed from *writing*, we will be left with *writ* which is not a word directly associated with *writing*, but this would probably be the stem form returned by most stemmers so that it can be matched against *write*, *writer*, or *rewrites* (which will also be stemmed to *writ*).[2]

Table 1 – Examples of stems, roots and lemmas in English

surface form	lemma	root	stem
interrupting	interrupt	rupt	remove –*ing* → interrupt
interruption	interruption	rupt	remove –*ion* → interrupt remove –*tion* → interrup
runs	run	run	remove –*s* → run
runners	runner	run	remove –*ners* → run remove –*s* → runner

[1] Prefixes are added to the beginning of a root or word. For example, in English 'non' is a prefix in *nonexistent* or 'dis' in *disillusioned* – both have a meaning of negation. Suffixes are added at the end of a root or word. Examples from English are '–ing' in *reading* or the plural '–es' in *peaches*. Prefixes and suffixes are generally called Affixes.

[2] Note that this search will also return all documents containing the word *writ* or *writs* meaning a formal legal document.

Root: the primary meaning part of a word that cannot be reduced in form any further. Note that the ***root*** doesn't have to be an actual word itself. For example, in English *rupt* is a root that is used to form the words *corrupt, interrupt, disrupt, rupture*; but on its own, it is not a word.

Lemma: that's what we normally think of as a word; it's the form you find in the dictionary.

Some examples are listed in Table 1. Note that although lemmas and roots are unique, a stem may vary based on the stemming algorithm used (i.e., based on the affix removed).

1.2 Level of Linguistic Analysis

- **Light stemming**

Most stemmers perform ***light stemming*** which does not involve deep word-level semantic or morphological analysis. These approaches strip off a small set of prefixes and suffixes, and may also provide limited substitutions, but do not deal with any changes in the internal word form. An example is the Porter algorithm for English that includes stripping rules as in 'EMENT → null' that removes the '-ement' suffix. For example, *replacement* would be stemmed to *replac.* The algorithm also contains substitution rules such as 'ATIONAL → ATE' (e.g., *relational* is stemmed to *relate*) or 'SSES → SS' (e.g., *tresses* is stemmed to *tress*).

Light stemmers are simple and relatively efficient. They may, however, miss certain forms or match word forms that are not relevant. For instance, irregular words are generally not handled in stemmers and pairs such as *goose* and *geese* will not be matched against each other, nor will *swim* and *swam*, unless certain special rules are included in the algorithm. If the stemmer is removing the suffixes '–ing' and '–ion' from words, it may not be able to conflate pairs of terms such as *computing* and *computation* as equivalent (which would be stemmed to *comput* and *computat*, respectively). On the other hand, a stemmer that stems the query *viewer* to *view* may retrieve all documents containing the forms *viewer, view, views, viewed, viewing* but also documents including words that are not necessarily related in meaning such as *preview* and *review*. Depending on the application needs, the user may not necessarily be interested in documents including e.g., *preview* when searching for *viewer*.

- **Lemmatization**

Lemmatizers stem the input word to a lemma form, a normalized word form that would be found in the dictionary. This is done by a series of stripping and substitution rules that derive the lemma, usually based on some grammatical knowledge of the word such as lexical category (e.g., noun, verb, adjective). Hence, the word *writing* will not simply be stemmed to *writ* by stripping off the '-ing' suffix, but it will also replace the affix with the '-e' ending to obtain the lemma form *write*. Another example is the plural form *properties*: basic stemming of the plural suffix would result in *properti*. A lemmatizer would replace the '-ies' suffix by '-y' resulting in the word form *property*. The idea of lemmatization is, therefore, to recover the initial word suffix to form the lemma. The advantage of this approach is that it is able to match irregular pairs such as *swim/swam* or *axis/axes.* Lemmatizers may use ***morphological analyzers*** (tools for doing complete analysis of word forms in a language) and sometimes take advantage of an existing ***lexicon*** (electronic dictionary) for the language in order to match the correct word forms.

- **Derivational stemming**

Some stemmers are limited to ***inflectional*** affixes while others also strip off ***derivational*** affixes. Inflectional affixes represent grammatical information on words such as number on nouns (e.g., '-s' makes a noun plural in English), tense on verbs (e.g., '-ed' marks the past tense on regular verbs in English), person on verbs (e.g., '-o' marks the first singular form in Italian as in *parlo* 'I speak'), definiteness on nouns (e.g., 'al-' in Arabic as in *al-kitaab* 'the book'), etc. A derivational affix, on the other hand, is used to form a new word and often changes the category of the original word. For example, *joy* is a noun in English that can become an adjective by adding the derivational suffix '-ful' to form *joyful,* which itself can become a noun by adding the derivational suffix '-ness' to obtain *joyfulness* or can become an adverb by adding the suffix '-ly' as in *joyfully*.

Because derivational affixes are used to form new words, stemming them may actually result in words that are unrelated to the original term. For instance, if a stemmer strips off the derivational suffix '-ize' which is used to form verbs as in *modernize*, then it will reduce the verb *authorize* to *author*. Clearly, if a user is searching for the term *authorize*, then he or she is probably not interested in documents containing the word *author* (this search will produce ***high recall*** with ***low precision*** as described below in section 1.5). Similarly, an aggressive derivational stemmer would also conflate *antiauthoritarianism* or *authorization* with the term *author* by removing all the derivational affixes from the search term (e.g., 'anti-', '-ism', '-ation'). For this reason, many stemmers are restricted to inflectional morphology. This choice is not so clear cut as sometimes derivational stemming may help in obtaining better precision. For example, a derivational stemmer would be able to match *notary* and *notarize* by removing the '-y' and '-ize' suffixes, respectively, and providing the identical stem *notar* for both terms.

1.3 Stop Words and Exception Lists

- **Stop words**

Most stemmers ignore common or high frequency words and characters such as *the, a, from, of,* and *how*, because they tend to slow down the search without improving the results. Stop words are eliminated from the representation of both documents and queries: A stop word typed into a search query will be ignored and stop words are not included in the index term. Generally, a stop word list (or stoplist) for each language is integrated in the stemming algorithm.

- **Exception list**

It is generally possible to create an exception list that contains surface forms and their associated stems to capture forms missed by the stemming algorithm. For instance, irregular verbs in English could be inserted in such a list, allowing the conflation of, e.g., *goose* and *geese*. Furthermore, special no-stem lists are sometimes included if a stemmer is too aggressive or if the user wishes not to stem certain words stripped by the algorithm.

1.4 Issues with Stemming

Certain problems encountered with different stemming algorithms have already been discussed. This section presents a more detailed discussion.

- **Understemming**

Understemming refers to the non-conflation of related words. For instance, the words *acquire* and *acquisition* are related to each other in form and in meaning and a user searching for documents on the "acquisition" of a certain company would probably want to also find all documents mentioning that the company was "acquired". Most light stemmers, however, would not be able to conflate the two terms since *acquired* would be stemmed to *acquir* or *acquire,* while *acquisition* would be stemmed to *acquis* (or *acquisit* or *acquisi* depending on the stemmer); therefore, the two words will not be matched against each other. In general, areas where understemming may be a problem is when the internal form of the word undergoes a change instead of simply taking an affix on the base form. Examples would be irregular plurals (e.g., *matrix/matrices* or *goose/geese*), but also derivational forms such as nominal forms derived from verbs (e.g., *acquire/acquisition, explain/explanation*) or noun and adjective derivations (e.g., *urgency/urgent, computing/computation*).

The most frequently asked question is why word X should be stemmed to x1, when one would have expected it to be stemmed to x2. It is important to remember that the stemming algorithm cannot achieve perfection. On balance it will (or may) improve Information Retrieval performance, but in individual cases it may sometimes make what are, or what seem to be, errors.
-Martin Porter

- **Overstemming**

Overstemming refers to cases where words that are not morphologically or semantically related are conflated. One example involves the words *stocks* and *stockings.* Hence, a search query on stock prices will probably return documents on stockings and vice versa, since both terms will be stemmed to *stock*. Another instance of overstemming occurs with the terms *organism* and *organization* which will be reduced to *organ* by a derivational stemmer. Clearly, the user searching for "organisms" would not be interested in documents containing the term "organization" since they are not related in meaning. Another example discussed earlier was the conflation of the terms *antiauthoritarianism* and *authors*.

- **Ambiguity in Meaning**

Another issue with stemmers involves words with multiple meanings. A user might be interested in only one sense of the word but since stemming is often based on the form of the word, other senses may also be retrieved. For example, the word *canine* can refer both to a dog or a kind of tooth, and both meanings will be returned. Since documents containing the senses that are not relevant could be returned, this will tend to reduce precision but give higher recall. On the other hand, if the user searches for the term *dog,* the stemmer will not relate the word to *canine* since they are not related in form and documents containing the latter would be missed. Hence, the result would be lower recall since relevant documents including related meanings of words are not returned.

- **Linguistic Differences**

Different languages may raise distinct challenges for stemming algorithms. English stemming is in general very efficient with only a few difficult cases such as *happier* needing to stem to *happy* rather than *happi* once the '-er' suffix is removed, or some irregular forms. However, stemmers become harder to design as the morphology, orthography, and character encoding of the target language becomes more complex. Languages with more complex declension forms and in particular languages with changes in the root form of the word, as in Hebrew and Arabic, raise a number of important issues for basic stemmers.

Italian stemming, for instance, is more complex than English because it has more verbal inflectional forms as can be seen with the conjugation of *arrivare* 'to arrive' in Table 2 compared to the two forms in English (i.e, *arrive* and *arrives*).

Table 2 – Italian verb *arrivare* in present tense

Italian Declension	English Translation
arrivo	I arrive
arrivi	you arrive
arriva	he/she/it arrives
arrivamo	we arrive
arrivate	you[pl] arrive
arrivano	they arrive

Agglutinative languages that can pile up several affixes together require a more complex affix list in order to obtain the word stem. Forms such as *kitaplarımda* 'in my books' is rather common in Turkish. This word consists of several affixes added to the noun *kitap* 'book': -*lar* marking the plural, -*ım* indicating the first singular possessive, and the locative case -*da* that can be translated as 'inside' or 'in'. Hence, a stemmer should be able to recognize and strip off the various combinations of suffixes such as '-lar,' '-ım,' '-da,' '-larım,' '-larda,' '-ımda,' and '-larımda.' This obviously makes stemming algorithms more complex for these language types.

Certain languages with heavy compounding such as German raise issues for basic stemmers that are unable to split off the subparts of the compound. For example, the German word *Bundesverfassungsgericht* meaning 'federal constitutional court' is written as a single word although it consists of the three separate tokens *Bundes* 'federal,' *Verfassung* 'constitution,' and *Gericht* 'court'. A user searching for the word *Gericht* 'court' will not be able to see any documents that only contain the compound word *Bundesverfassungsgericht* since the stem forms do not match. Another example is the Hungarian word *ópiumelöállítás* 'opium manufacture' where a search for *ópium* would not retrieve documents containing the compound word.

In addition, languages in which the internal form of the word varies significantly when forming related words raise a number of difficulties for conventional stemming algorithms. German often forms its plurals by modifying the form of the word as in *Haus* 'house' becoming *Häuser* 'houses'. The main challenge in this domain, however, is raised by semitic languages such as Hebrew and Arabic that are based on a ***templatic morphology*** system. In these languages, words are based on a template (generally consisting of three consonontal letters) and related words are derived by inserting different vowels. For example, the plural of *kitaab* 'book' in Arabic is *kutub* 'books'; both words are formed on the three-letter template 'k-t-b' but have different vowel combinations. Stemmers are generally unable to conflate these terms.

1.5 Evaluating Stemmers

Information retrieval systems are evaluated based on their ***recall*** and ***precision*** scores. Recall is defined as the percentage of relevant documents that are retrieved as relevant by the search system, and precision refers to the percentage of documents returned as relevant which were correctly classified. In practice, there is a trade-off between recall and precision since a system giving high recall, where documents with terms that are morphologically related to queries are returned, can give low precision output as it may return semantically unrelated documents. This would be the case, for instance, if the user enters the query *fishing* and is interested only in documents discussing the activity but ends up getting in addition documents involving *fish* in general. The stemmer in this case has high recall and low precision.

In general, light stemmers are effective and robust but rule-based linguistic stemmers provide more accurate results (although this may depend on the language as well). The choice between an aggressive stemmer and a conservative stemmer or even stemming instead of lemmatization clearly depends on the task the stemmer is going to be applied to. When used for document retrieval or similarity measures, aggressive stemmers can suffice as they reduce all words with a related form to roughly the same root. When the task involves a more delicate operation such as topic detection, the meaning of the words must be conserved as far as possible in which case lemmatizers taking advantage of deeper linguistic analysis would be more valuable. Another consideration is often the time and effort spent on the stemmer – light stemmers require considerably less effort than lemmatizers in general. Furthermore, lemmatizers also require more computational power in general since they perform a complete morphological analysis.

1.6 Query Syntax

Information retrieval systems differ based on the type of queries processed. One common type of query is the Boolean combinations of relevant keywords or phrases linked by terms such as AND, OR, NOT, etc. Search algorithms may also allow the user to enhance the search by using nested queries whereby the search terms and operators are enclosed in parentheses or brackets to specify the order in which they should be processed. Certain search systems can also process wildcards or the relative importance of search terms. A better understanding of the query terms for each retrieval system as well as the characteristics of the stemming algorithm being used allows the user to enhance the search accordingly and obtain better results. The operators used in Lucene are described in more detail in section 2.

2 Lucene Query Syntax

Apache Lucene is a high-performance text search engine library written entirely in Java. It is available as open source and is integrated within the Argus search system. Lucene provides a rich query language through the Query Parse which interprets a string into a Lucene Query using JavaCC. The user is able to enhance searches by combining various operators and terms. This Query Syntax is described in this section and Table 3 summarizes the various operators and their functions.

2.1 Terms

Terms include single words such as `flu` or phrases as in `` ``avian flu'' ``. Phrases are entered with quotation marks around them. In the Argus system, each query term is stemmed and conflated with the stemmed index terms in the documents. If a match is found, the document will be returned as relevant to the search. In the case of phrases, each term within the phrase is usually stemmed before matching against the index terms. Note, however, that stop words are discarded even if they are entered as a search term.

2.2 Boolean Operators

Boolean operators allow the user to combine words or phrases to broaden or narrow the search. Lucene supports the AND, OR, NOT, -, and + operators[3]. The default operator is OR (i.e., if no operator is specified, the OR operator is used). Note that Boolean operators have to be typed in all caps.

- **OR**

The OR operator finds a matching document if either of the terms specified exist in a document. For example, to search for documents containing either the term *flu* or the term *influenza*, the user can enter `flu OR influenza`. Since OR is the default operator, the search string `flu influenza` will also provide the exact same results.

- **AND**

The AND operator finds a matching document if both of the terms specified exist in a document. For example, to search for documents that include both *bird* and *flu* enter `flu AND bird`.

- **NOT**

The NOT operator excludes documents that contain the term following NOT. Hence if you are searching for documents about *avian flu* but not cases occurring in *Jordan*, you may search for `` ``avian flu'' NOT Jordan ``. It is not possible, however, to use NOT on a single term as in `NOT Jordan`.

- **+**

[3] Keep in mind that in Argus, the application of the Lucene operators is combined with the stemming of the query terms. Thus, if a stemmer is included for the language, the search term `influenza AND birds` should not only return documents containing both of the query terms, but also any of their variants such as *bird* or *influenzas*. In addition, stop words are ignored both in the search term and in the document, hence the query term "`enhance search`" may return a document containing *enhance the search*.

The + operator requires that the term following it appear in the document. Hence, if you are looking for a document that must contain *influenza* and it may contain *birds* but the latter is not necessary, use `+influenza birds`. In this case, Lucene would provide a higher relevancy score to documents that also include *birds,* but since articles are typically listed by publication date in Newsstand rather than by relevancy score, this search is equivalent to simply searching for `influenza`.

- **-**

The - operator requires that the term following it not appear in the document. Hence, if you are looking for a document that contains *influenza* but should not include *birds*, use `influenza -birds`. Note that logically this search is equivalent to the search query `influenza NOT birds`.

2.3 Precedence and Grouping

The ***order of precedence*** for Boolean operators in Lucene is not clear as several bugs have been reported in the precedence relation[4]. It is a good idea, when a search query contains more than one operator, to always use the ***grouping*** mechanism by explicitly enclosing the search terms and operators in brackets in order to specify the order in which they should be processed.

So, if the query entered is `flu OR influenza AND avian`, two groupings are possible. If the query is bracketed as `(flu OR influenza) AND avian`, first all documents containing either *flu* or *influenza* or both will be retrieved and then only the ones also containing the word *avian* will be returned. In essence, the retrieved documents could contain *flu* and *avian* (but no *influenza*), or *influenza* and *avian* (but no *flu*), or all three terms; there will be no document not containing the term *avian*.

On the other hand, with the search query `flu OR (influenza AND avian)` first all documents containing both *influenza* and *avian* will be retrieved, then the system will look for any documents containing the word *flu.* The result would consist of documents either containing the combination of *influenza* and *avian* or documents containing *flu* or documents containing all three terms.

If more than one set of parentheses is used in the query, the innermost (or nested) parentheses will be processed first.

2.4 Term Modifiers

A number of modifications can be made to the basic terms to enhance the search.

- **Wildcards**

Wildcards are used to match patterns. There are two wildcards that can be used in searching: the single character wildcard '?' and the multiple character wildcard '*'. For example, in the query term `te?t`, the wildcard will be replaced by another character; this query could then match either *text* or *test* or *tent*, etc. Note that this wildcard does not match the absence of a character and this query will not return *tet*. The multiple character wildcard, on the other hand, looks for 0 or more characters. Hence, the term `medic*` will return documents containing words like *medic, medicine, medical, medicinal, medicate, medicant,* etc. The two wildcards may also be combined within the same term but wildcards may not be used as the first character in the word.

[4] cf. http://wiki.apache.org/jakarta-lucene/BooleanQuerySyntax.

One should be careful when using wildcards not to allow too many matches since that would result in a large number of irrelevant documents being returned. For example, the search term `elect*` could return documents containing *elect, elected, electoral, election* but also terms like *electronic, electron, electrical, electricity,* etc. Similarly, the query term `avia*` will retrieve all documents containing the words *avian, aviary, aviation, aviator*, etc. A search for `g*se` with an embedded multi-character wildcard could find documents including *goose* and *geese* but will also retrieve articles with any of the following: *galvanise, glimpse, greenhouse,* or *glucose*. While a search for `g??se` with embedded single character wildcards matches the following: *goose, geese, guise, grise,* and the proper name *Ghose*.

- **Fuzzy search**

Fuzzy search can be performed with the tilde '~'; it allows the user to look for words "similar" to the search term. For example, in the query term `roam~`, the search will find terms like *roams* or *foam.* Fuzzy search is based on the Levenshtein Distance (or Edit Distance) algorithm. Levenshtein distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. Hence, in the *roam / foam* example above, the edit distance would be 1 since one substitution of 'r' to 'f' has taken place. This feature is useful when matching proper name variants. For example, *Mohamad* is closer to *Mohammad* than to *Mahmoud* according to this algorithm.

The user can specify the value of the similarity desired by adding a number between 0 and 1 after the tilde as in `roam~0.8`. If the value is closer to 1, only terms with a higher similarity will be matched. The default value is 0.5.

It is not recommended to use fuzzy search in the Argus search system, however, since the results are rather unpredictable (the "similar" word could be obtained by deleting, substituting or inserting any of the characters in the word) and return too many documents.

- **Proximity search**

With proximity searching the user can specify how near or distant the search terms should be from each other in the text. This is helpful if you want to make sure that the two terms being searched are not just appearing anywhere in the document but are closely related to each other (e.g., within the same sentence or paragraph). The proximity operator is also the tilde '~' but in this case it is used after a phrase and it should be followed by a whole number indicating the proximity of the search terms. For example, the search term ``` ``malaria avian''~10 ``` will return documents where the words *malaria* and *avian* appear within 10 words of each other within the same document. Note that this query requires that both terms be present in the document and stop words are not included when determining the distance between words.

The order of the terms is important; in other words, ``` ``malaria avian''~10 ``` is not equivalent to ``` ``avian malaria''~10 ```.

- **Boosting**

Boosting allows the user to indicate the relative relevance level of the terms. To boost a term, a caret '^' is used at the end of the term followed by a number indicating the boost factor. The higher the boost factor, the more relevant the term will be. For instance, '^2' means that the word is twice as important as a word with no caret, while '^3' means it's three times as important. The search engine will then list the more relevant documents first.

For example, the search query `malaria avian` will return any documents containing *malaria* or *avian* or both. However, if one of the terms is boosted as in `malaria^4 avian` then the documents containing that term (i.e., *malaria*) will be listed as more relevant. Phrases can also be boosted as in the following example: ``` ``avian flu''^4 ``measles vaccine'' ```. In this case, documents containing the phrase *avian flu* will be listed as more relevant (i.e., at the top of the list) than the ones containing *measles vaccine*.

With the current Newsstand system, however, since retrieved documents can only be listed chronologically, or ordered by news source or country source in Newsstand, documents cannot be displayed by relevancy score. Hence, the boosting feature is not currently available.

Table 3 – Lucene operators and their functions

Operator	Example	Function
OR	`influenza OR birds`	Finds documents that have either the term *influenza* or the term *birds* or both.
AND	`influenza AND birds`	Finds documents that have both *influenza* and *birds.*
NOT or -	`influenza NOT birds` `influenza -birds`	Finds documents that have only *influenza.* The term *birds* is excluded.
+	`+influenza birds`	Finds documents that have the term *influenza*. These documents may also contain the term *birds* or they may not.
?	`hospitali?e`	Finds documents that contain the term *hospitalize*, *hospitalise*, *hospitalite*, etc.
*	`hospi*`	Finds douments that contain the words *hospital*, *hospitality*, *hopsice*, *hospitalize*, *hospitalized*, *hospitalizations*, etc.
~	`roam~`	Finds documents that contain words similar to *roam* such as *foam, roams, goal, boat, room.*
	``` ``influenza bird''~3 ```	Finds documents that contain both *influenza* and *birds* and where they appear within 3 words of each other.
^	`influenza^3 birds`	Finds documents that have either the term *influenza* or the term *birds* or both, and lists the ones containing *influenza* on top as more relevant. (Not available in Newsstand)
" "	``` ``avian flu'' ```	Finds documents that contain the phrase *avian flu*.
( )	`avian OR (flu AND vaccine)`	The operator within brackets is applied first. Finds documents that contain both *flu* and *vaccine* or documents containing *avian*, or documents including all three terms.

3 Searches on Newsstand

Understanding the stemmers' capabilities for each language combined with knowledge of the query syntax in Lucene can allow for enhanced searches. This section presents a set of guidelines that can be followed to cut back on redundancy in the query terms, reduce the number of irrelevant retrievals, and render the results more precise. The examples are provided from several of the Newsstand languages.

3.1 Highlighting and Display Issues

The highlighting of the words in the retrieved documents in Newsstand is independent of the stemmers and is handled by a separate algorithm. Currently, only exact matches are highlighted. Even if the document returned contains the relevant variant form for the search query, it may not be highlighted if it is not an exact match in form to the original query word. Note also that even exact matches are not highlighted in the title of the document or if they are split between lines. Furthermore, if the encoding of the characters in the search term and the one in the returned document are not identical, the word will not be highlighted in the text. This is the case especially with Farsi documents that alternate between the use of the Arabic and the Farsi ک and ی characters. What is crucial then is to realize that, just because a retrieved document does not include highlighted words, it does not indicate that the stemmer and search engine did not perform correctly. The analyst can look for the desired term in the retrieved document by using the 'Find' function (Control-F in Firefox and Microsoft tools). Note, however, that the term in the document may be a variant of the form that was entered as the query; thus only the stem form should be searched in the document, avoiding any encodings that may not be matched correctly (e.g., the Farsi ک and ی characters).

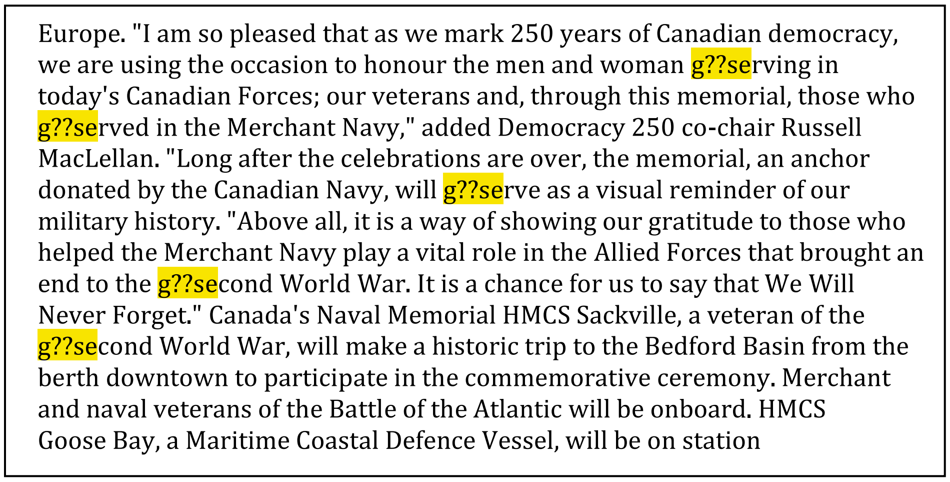
Europe. "I am so pleased that as we mark 250 years of Canadian democracy, we are using the occasion to honour the men and woman g??serving in today's Canadian Forces; our veterans and, through this memorial, those who g??served in the Merchant Navy," added Democracy 250 co-chair Russell MacLellan. "Long after the celebrations are over, the memorial, an anchor donated by the Canadian Navy, will g??serve as a visual reminder of our military history. "Above all, it is a way of showing our gratitude to those who helped the Merchant Navy play a vital role in the Allied Forces that brought an end to the g??second World War. It is a chance for us to say that We Will Never Forget." Canada's Naval Memorial HMCS Sackville, a veteran of the g??second World War, will make a historic trip to the Bedford Basin from the berth downtown to participate in the commemorative ceremony. Merchant and naval veterans of the Battle of the Atlantic will be onboard. HMCS Goose Bay, a Maritime Coastal Defence Vessel, will be on station

Figure 1 – Spurious highlighting example in Newsstand

Another issue noted in the highlighting mechanism involves spurious highlights. For instance, if a stop word (generally a two-letter form) appears in the query syntax, it may be highlighted throughout the document. Again, the stemmer ignores the stop words and the highlighting is not indicative of the stemmer functionality. In addition, searching with a wildcard sometimes causes the query term to be displayed in the document although it is clearly not part of the original text. For example, a search in English for `g??se` returns documents containing the spurious highlights shown in Figure 1, which does not reflect the performance of the stemmer in this case.

Other display issues include lines cut off at every word, especially in the Farsi and Arabic documents and unidentified fonts that show up as gibberish. Once again, these display issues are independent of the stemmer and search engine functionalities.

3.2 Exact Search

All query terms are stemmed in the system and it is not generally possible to force a word not to be stemmed. For example, a Farsi search for "کودکان کار" (*kudækane kar*) 'child laborers' would return documents containing that exact phrase, but it will also stem the two words in the search query and return documents that include the phrase کودک کار کرد (*kudæk kar kærd*) 'the child worked'. In this case, the term *kudækan* has been stemmed to the singular form *kudæk* and matched against the index term in the corresponding document. The system does not allow the option to search for an exact match without any stemming. This can be accomplished, however, by listing the word or phrase in a no-stem list if absolutely necessary.

Note that in the original Lucene search system, the documents containing an exact match will receive a higher score and will be listed on top, followed by those that include variant forms of the query. However, the retrieved documents are listed by publication date in Newsstand (with an option to order them by source name or source country) but an ordering based on the Lucene relevancy score is not available in the current system.

3.3 Stop Words in Search

Stop words should not be included in searches as they are filtered off prior to stemming and search in both the query term and the document itself. The stop word will therefore not play any part in the search itself even if it is included in the query term. For example, a search in Italian for `dell'attentato suicida con un'autobomba` contains several stop words (i.e., *dell'*, *con*, *un'*), each of which will be removed prior to running the search. This term is in fact equivalent to the search query: `attentato suicida autobomba`. Similarly, the prepositions از (*æz*) 'from' in the Farsi query "مردن پرنده از" (dying-of bird from) and مانند (*manænd*) 'like' in "مانند انفلوانزا" (like influenza) will not be included in the search and are therefore redundant. Thus, only concept words are relevant in a query term.

As another example, consider the French query term ``` ``achats de panique'' ```. In this case, the stop word *de* is included in the term yet it is ignored by the search engine. Hence, this phrase will match all of the following phrases found in a text: *les achats-panique, les achats en panique, des 'achats paniques', des achats de panique.* In fact, the term will also retrieve a document with the following phrase: … *au pouvoir d'achat "Panique sur les prix"…*

Proximity searches are also affected by the number of intervening stop words between search terms, since the stop words are ignored in computing the distance between the terms. For example, a search for ``` ``flu Indonesia''~1 ``` will retrieve a document that includes *flu fatalities in Indonesia* since the preposition *in* is a stop word and is not counted when determining the distance between the two search terms.

3.4 Using Stemming

Although each language-specific stemmer performs differently, query terms are typically stemmed, and there is no need to list variant forms of the query term unless the stemmer misses these particular forms. For example, the Italian stemmer is able to conflate the singular and plural forms of nouns such as *uccello* and *uccelli*, or *tacchino* and *tacchini*. It is, therefore, not necessary to enter the search query as: `tacchino OR tacchini`. A query including simply `tacchino` would retrieve the same set of documents. In most stemmers, verb forms are also stemmed. Thus, a query for `augmenter` in French would also retrieve documents not only containing the various tenses of the verb but also the noun `augmentation.`

The Turkish stemmer also conflates distinct variants of the word. Thus, the query term `eksiklik` would return documents including any of the following words: *eksikliğinin, eksikliği, eksikliğini, eksikliklerini,* or *eksikliğimidir*. It is therefore not necessary to enter a search query with the two variants as in `eksiklik OR eksikliği`.

Note that if a query is a phrase, each term is stemmed prior to conflation. Hence, the following Russian query term `"горят леса" OR "горит лес"` seems redundant since each one of those phrases will be conflated with the other. In other words, a search for e.g., `"горит лес"` would suffice. Also, the Russian query `"Необычная болезнь"` 'unusual disease' would return documents containing any of the following related forms: необычного в болезнях, необычной болезни, необычных болезней, необычной "болезни Моргеллонов", необычным болезням. Similarly, `attentato suicida` in Italian would also return documents including variants such as *attentati suicidi*. In Farsi, the search term"اقدام متقابل" (*eqdame moteqabel*) would also retrieve documents containing the variant forms such as "اقدامات متقابل" (*eqdamate moteqabel*) or "اقدام متقابلش" (*eqdame moteqabelæsh*).

3.5 Lack of Stemming

The English language system currently does not provide any stemming at all. Therefore, the following query is valid where all the relevant variant forms are listed explicitly:

```
outbreak epidemic AND ((bird OR birds OR cow OR cows) AND (dead OR death OR
deaths OR dying OR sick OR ill))
```

In the systems for other languages where the stemmer fails to produce the related variants, the many forms should be explicitly listed. One such example is typically irregulars where the internal form of the word is modified. In the case of the Farsi stemmer, which fails to conflate the singular and plural forms of e.g., نماینده / نمایندگان (*næmayænde / næmayændegan*) 'representative / representatives', both forms should be explicitly listed when running queries.

For languages that may represent the compounds either as separate words (or hyphenated) or attached to each other, both forms may need to be included in the query term since stemmers would not be able to conflate subparts of a compound word. For example, a search for the English `toothbrush` will fail to match a document that contains the term *tooth brush*, with the two subparts written separately. Also in Farsi, the term `خشکسالی` 'drought' will not match the word خشک‌سالی (subparts separated by a half-space) or the word خشک سالی (subparts separated by a full space), and therefore all three forms may need to be included in the query term.

3.6 Queries with Operators

In order to reduce the number of retrieved documents that are not directly relevant to the search query, the Lucene operators and term modifiers can be used to enhance the searches or to make the results more precise. AND and NOT are very useful in limiting search results, but in addition the term modifiers can be used to filter irrelevant documents.

For example, the Farsi search for مغز AND ورم (brain AND swelling) can return a document on the topic of garlic where it discusses a head of garlic and its healing benefits for swollen body parts. But if the term is modified with a proximity search such as "مغز ورم"~5, forcing the two words to appear within 5 words of each other, the results become more relevant and the article on the healing properties of garlic is not returned. The proximity search makes it more probable that the two words in the query are directly related to each other in the article since they are appearing within the same sentence or paragraph.

NOT can be used to restrict the retrieval of unrelated documents. For instance, an English search for *goose* may return articles discussing *Mother Goose* . These could be eliminated by the query `goose NOT mother` or more precisely, `goose NOT ``Mother Goose''`.

Wildcards when available can be very useful in searching for related terms, especially if the user is aware that the stemmer misses the variants (e.g., due to changes in the internal form of the word). For example, a search for `sw?m` can return both forms *swim* and *swam*. But as mentioned in section 2.4, wildcards can also increase the recall exponentially and return many irrelevant documents.

4 Language-specific issues: Arabic

Arabic displays complex affixation whereby a given word can appear in a number of different forms that should be conflated for search and information retrieval applications. Besides elements like plurals and tense markers, Arabic words may be found with definite articles, conjunctions, and particles attaching to the beginning of the word, and pronouns and gender markers attaching to the end of the word. One of the main issues in Arabic stemming, however, is the presence of the root-pattern word-formation system (also known as templatic morphology). In this language, most words are derived from a few thousand roots by changing the internal form of the word. For example, based on the root *ktb*, Arabic can build a number of nouns, adjectives and verbs by inserting various vowel patterns such as *kitaab* (book), *kutub* (books), *maktab* (office), *kataba* (he wrote), *naktubu* (we write), *kaatib* (writers), *kitaaba* (writing), *kitaabaat* (writings). Although in some cases, we would like to be able to relate some of these forms for information retrieval purposes by stripping the words down to the basic roots, the resulting words may also not be directly related to each other semantically as in *kitaab* (book) and *maktab* (office). In these cases, it is preferred to simply strip off affixes and reduce the word forms to basic stems rather than the smaller unit of the root.

This section provides a brief overview of the Arabic stemmer used in Argus and a description of the stemming capabilities. In addition, a number of existing issues are discussed and potential solutions are recommended if possible.

4.1 The Buckwalter Analyzer

The current Arabic stemmer is based on the Buckwalter system which includes three distinct lexicon files listing all prefixes, suffixes, and stems. These lexicons are supplemented by three compatibility tables that basically list whether the combination of the elements in the lexicon files are valid in Arabic. Hence, there is a table listing all possible prefix-stem combinations, another table includes the valid stem-suffix combinations, and finally a third table controls the prefix-suffix combinations.

Given an input word, the token is segmented into possible combinations of prefix, stem, and suffix and then looked up in the corresponding lexicons. First, the algorithm determines whether all three word elements are found in their respective lexicons. If this is the case, then the morphological categories for each element (listed in the lexicon) are used to determine whether they are compatible by looking it up in the corresponding compatibility table. The morphological categories are defined within the system and include terms like `Pref-Wa`, `PVSuff-ah`, and `Ndu`, among many others. This entails that if an element is not listed in the system lexicon, it will not be processed correctly. The disadvantage of this system is that, in order to improve the coverage of the language or to focus on a specific domain vocabulary, it is necessary to list not only the words in the lexicon but also their corresponding sets of prefixes and suffixes.

For an example of how the Buckwalter analyzer functions, consider the word وصفه (transliterated[5] as `wSfh` in the Buckwalter system). The segmentation finds a number of possible ways to strip off the word into a prefix, stem, and suffix combination as shown in

[5] *Transliteration* is the representation of characters of one alphabet by those of another. In general, transliteration is performed into ASCII or Latin-script characters. A transliteration maps the written characters, letter by letter, but does not really need to represent how they are actually pronounced in the language.

Figure 1. Then each potential solution is checked against the pre-determined compatibility relations to make sure that the selected prefix+stem+suffix set is valid.

```
LOOK-UP WORD: wSfh
SOLUTION 1: (waSafahu) [waSaf-i_1]
waSaf/VERB_PERFECT+a/PVSUFF_SUBJ:3MS+hu/PVSUFF_DO:3MS
(GLOSS): + describe/characterize + he/it <verb> it/him
SOLUTION 2: (waSafahu) [waSaf-i_1]
waSaf/VERB_PERFECT+a/PVSUFF_SUBJ:3MS+hu/PVSUFF_DO:3MS
(GLOSS): + prescribe/give a prescription to + he/it <verb> it/him
SOLUTION 3: (waSofh) [waSof_1] waSof/NOUN+hu/POSS_PRON_3MS
(GLOSS): + description/portrayal/characterization + its/his
SOLUTION 4: (waSofh) [waSof_2] waSof/NOUN+hu/POSS_PRON_3MS
(GLOSS): + characteristic + its/his
SOLUTION 5: (waSaf~ahu) [Saf~-u_1]
wa/CONJ+Saf~/VERB_PERFECT+a/PVSUFF_SUBJ:3MS+hu/PVSUFF_DO:3MS
(GLOSS): and + arrange/classify + he/it <verb> it/him
SOLUTION 6: (waSaf~h) [Saf~_1] wa/CONJ+Saf~/NOUN+hu/POSS_PRON_3MS
(GLOSS): and + line/row/class + its/his
```

Figure 2 – Sample analysis of Buckwalter system for وصفه (wSfh)

In this case, solutions 1 (وصف+ه) and 6 (و+صف+ه) are valid combinations as shown in Table 4 and Table 5, respectively. Solution 1 is the verb *wasaf* 'to describe' followed by the 3rd person masculine singular suffix representing either the subject or the object of the verb meaning 'he described' or 'described it/him'. Note that a null prefix is allowed in this system represented by `Pref-0`. Solution 6, on the other hand, involves the noun *saf* 'line, row' and combined with the *wa* 'and' prefix and the *hu* suffix, it can be translated as 'and his/its line'. In the stemming application in Argus, the prefix and suffix elements are stripped off and only the stem is considered in the retrieval process.[6]

Table 4 – Analysis of وصفه (wSfh); solution 1

(null)	(null)	Pref-0	(null)
wSf	waSaf	PV	describe;characterize
h	ahu	PVSuff-ah	he/it <verb> it/him <pos>+a/PVSUFF_SUBJ:3MS+hu/PVSUFF_DO:3MS</pos>

Table 5 – Analysis of وصفه (wSfh); solution 6

w	wa	Pref-Wa	and <pos>wa/CONJ+</pos>
Sf	Saf~	Ndu	line;row;class
h	h	NSuff-h	its/his <pos>+hu/POSS_PRON_3MS</pos>

4.2 Stemming Process

- **Basic stemming**

[6] For more information on the Buckwalter analyzer see the following websites:
http://www.nongnu.org/aramorph/english/ and http://www.qamus.org/morphology.htm
A demo of the system is available at http://www.arabic-morphology.com/.

The Argus Arabic stemmer can successfully strip off various prefixes (e.g., و ، بـ ، الـ) and suffixes (e.g., ـة ،يه ، ان ، ها), providing the correct stem for the word. So, given the query term الدجاج 'the poultry', the stemmer will correctly stem it to دجاج by removing the definite article *'-al'* and match it against other inflected forms of the word that are also stemmed to the same form such as دجاجة 'bird, fowl', والدجاج 'and the poultry', لدجاج 'for the poultry', الدجاجة 'the fowl, the chicken'.

Similarly, all the following inflected forms of the word are correctly stemmed and conflated, allowing the successful retrieval of documents:

سفلوة – السفارة – بالسفارة – للسفارة – وسفارتهما – سفارات – سفارتكم– سفارتها ← سفارة

محافظ – المحافظة – والمحافظة – المحافظات – بمحافظ – بمحافظة ← محافظ

بترول – البترول – بالبترول ← بترول

String searches (i.e., phrases in quotation marks) are handled correctly. Each term is stemmed and searched and adjacent related word forms are retrieved successfully. For example, the search query "التنفس المسموم" could match combinations of تنفس and مسموم.

- **Normalization**

The characters with diacritics are normalized so that they can be matched against the same character without an overt diacritic. For instance, "alef with hamza below" and "alef" (without hamza) are equated when matching queries. Similarly, any vowel diacritics are stripped off when matching terms. Thus, searching for the term انفلونزا 'influenza' (without a hamza on the alef) will also return documents containing the term إنفلونزا 'influenza' (with a hamza below the alef).

The stemmer also removes any punctuation such as period, comma, exclamation point, or quotation mark when running queries. So, if the word appears next to a punctuation mark in the document, it will still be matched correctly.

- **Stop words**

Stop words in Arabic such as من 'from' or في 'in' are discarded by the stemmer. If they are included in the search query, they will be ignored. Hence, a search for "من الدجاج" is equivalent to a search for الدجاج.

4.3 Issues in the Arabic Stemmer

The Arabic stemmer fails to process certain elements. Some of these are expected given the design of the system while others seem to be bugs in the system causing the stemmer to miss certain forms that we would expect to stem correctly. This section lists these issues. It is useful to keep these problems in mind when designing search queries.[7]

- **Word-internal changes**

The Arabic stemmer will not be able to relate words whose internal forms have changed considerably. For example, if the query السفارة (*alsifaara*) is entered, the stemmer will be able to match it to the plural form سفارات (*sifaaraat*) as already mentioned, because the latter is formed by adding a suffix -*aat* to the end of the word. However, if the query is مرض (*marađ*)

[7] This section provides a description of the stemmer issues in order to aid analysts in performing Arabic searchers. However, the source code was not investigated and the specific causes of the bugs have not been determined.

the stemmer will not find its plural form امراض (*amraad*) since the latter is not obtained by adding an affix to the word but by modifying the internal form of the word.

Recommendation: In these cases, the user would need to include the variants of the word explicitly in searches.

- **Word not in lexicon**

Since the system is based on a look-up table of prefix, stem and suffix lists, if a word does not exist in its lexicons, the stemmer will be unable to perform any stemming, normalize diacritics, or remove punctuation. For example, although إنفلونزا (*influnzaa*) is stemmed correctly in the current lexicon, its form with "alef with hamza above", i.e, أنفلونزا (*aanflunzaa*) is not stemmed. Similarly, الكوليرا (*alkuliraa*) is not stemmed correctly[8]. In these cases, only an exact match of the query term is returned.

This also affects string searches (i.e., phrases in quotation marks). If one of the terms does not exist in the lexicon, it will not be stemmed. One such word is طيور 'birds' that is not stemmed by the current system and only exact matches are found in documents. Given the search query "الانفلونزا الطيور" 'bird flu,' only الانفلونزا will be stemmed while الطيور will be found in a document only if it matches the exact form of the word. Hence, this query could match the phrases listed below but not one that contains the form **طيور.**

الانفلونزا الطيور- انفلونزا الطيور - وانفلونزا الطيور- بإنفلونزا الطيور- إنفلونزا الطيور- لانفلونزا الطيور

Recommendation: Further investigation of the source code is required to determine whether the reason for this behavior is due to the word not being listed in the lexicon or if there is a bug in the program that does not allow it to be accessed successfully. To add a word to the lexicon in the Buckwalter system, however, requires that the user be aware of the stem and the potential prefix and suffix combinations of the word. Meanwhile, the user should be aware of this potential behavior and enter the various word forms for the terms that are failing to be stemmed in Arabic.

- **Overstemming**

There may be cases of overstemming where two unrelated words are reduced to the same stem and conflated. For example, منطقة (*mantaqa*) 'region' and منطق (*mantiq*) 'logic' may not be related semantically although they reduce to the same form, resulting in retrieved documents that are irrelevant to the search.

Recommendation: Unfortunately, not much can be done to stop this output. Note that entering a search query such as منطقة NOT منطق is not an option since the result will be null (as both words stem to the same form).

- **Unanalyzed suffixes**

Certain suffixes are not being stemmed by the current system, which seems to be a bug in the Arabic stemmer.

One of these suffixes is the dual marker –*in*; thus given the search term محافظ 'conservative', the form محافظين 'two conservatives' will not be stemmed correctly and the document

[8] Probably because it's not in the lexicon although the source code has not been investigated to detect the specific reason for this behavior.

containing it will not be retrieved. Similarly, the following pairs are not conflated: مقاتل and مقاتلين 'two cmbatants', متمرد and متمردين 'two rebels', عنصر and عنصرين 'two elements/components'. In these cases, only an exact match is found.

Certain feminine nouns ending in "ta marbuta" are not being stemmed correctly such as الشركة 'the company' or العنصرية 'racism/the racist woman'[9]. Similarly, the feminine plural marker *–aat* is sometimes not stemmed on nouns as in the case of دجاجات 'the chickens/fowls' or حيوانات 'the animals'. Only exact matches are processed in these instances. However, other nouns with the *–aat* ending are correctly stemmed, such as سفارات 'the embassies' or وزارات 'the ministries'.

Two other suffixes that seem not to be stemmed correctly are the ي suffix and the و suffix. To illustrate, the query term مقاتل 'combatant' will match the following forms:

المقاتل– مقاتل – المقاتلون – والمقاتلون

but will fail to conflate the following:

المقاتلة – مقاتلي – ومقاتلة – مقاتلين – والمقاتلين – مقاتلو

Recommendation: Further investigation of the source code is required to determine the reason for this behavior. Meanwhile, the user should be aware of the list of unanalyzed suffixes and enter the various word forms for these terms if needed.

- ***Shadda* normalization**

Although diacritics are normalized, the *shadda* diacritic is not. Hence, if a word is queried without the *shadda,* it will not be matched against the same word in a document if the latter contains the *shadda* diacritic. For example, the query term متطرف 'extremist, radical' will not be conflated with متطرّف .

Recommendation: If a word may carry a *shadda*, both forms should be entered as query terms.

- **Query syntax**

Wildcards do not function with Arabic terms.

[9] Adjectives, however, seem to be stemmed correctly as in مهمة 'important (f)' or محافظة 'conservative (f)'.

5 Language-specific issues: Farsi

Farsi (Persian) words are formed by adding prefixes and suffixes to a root form. For instance, although Farsi is not as agglutinative as Turkish, it does allow the attachment of several affixes to each other as in the word انقلابیترینهایشانند (*enqelabitærinhayeshanænd*) that translates into a whole sentence in English meaning 'They are the most revolutionary ones among them.' This word consists of the following elements *enqelab-i-tærin-ha-yeshan-ænd*: *enqelab* 'revolution,' the *i* affix deriving an adjective from a noun, the superlative suffix *tærin*, plural marker *ha*, pronoun for third person plural *yeshan*, and the 'to be' verb form *ænd* meaning 'they are'. The decomposition is shown below:

enqelab	*i*	*tærin*	*ha*	*yeshan*	*ænd*
revolution	ADJ	SUPERL	PL	PRON.3PL	BE.3PL

'the most revolutionary ones among them'

The ***morphotactics*** – i.e., the order in which the affixes attach to a word – is rather restricted. For example, as exemplified in the word *enqelabitærinhayeshanænd*, the superlative suffix on adjectives precedes the plural marker and the attached pronoun forms appear after the plural suffix. A different order will be ungrammatical. Nevertheless, the number of all possible combinations of suffixes poses significant problems for Farsi stemmers.

In addition, a number of Farsi affixes can appear either attached to the word, separated from the word by a full whitespace, or separated by a ***half-space***. These are exemplified for the plural marker *ha* in Table 6. The example in (1) shows the affix attached to the word *ketab* 'book' meaning 'books'. In (2), it is separated from the word by an intervening space. In (3), the last letter of the word *ketab* appears in the final form (i.e., it is not attached to the following affix) but there is no whitespace separating it from the affix. In this case, there is what is known as a half-space (نیمفاصله) separating the two elements[10]; the half-space is represented as a tilde /~/ in the English transliteration. A Farsi stemmer should be able to process all of these forms.

Table 6 – The forms of the separable suffix *ha*

Surface Form	Farsi Script	Transliteration
(1) attached form	کتابها	ktabha
(2) separated form	کتاب ها	ktab ha
(3) half-space form	کتاب‌ها	ktab~ha

Farsi has borrowed a large number of words from Arabic and has thus inherited the Arabic words based on the root-pattern word-formation system; this has led to word-internal changes that are typically difficult for stemmers to process. For example, based on the root *ktb*, Farsi has the words *ketab* 'book', *kotob* 'books', *mæktæb* 'school', *mæktub* 'written', *kateb* 'scribe', *kætibe* 'inscription'[11].

[10] The half-space can be formed on Word 2007 by typing Ctrl-Shift-2 and is represented as \u200c in Unicode. It is technically known as ZWNJ or Zero-Width Non-Joiner.

[11] Note that the pronunciation of words of Arabic origin as well as some meanings of the words are different in Farsi.

This section provides a brief overview of the Farsi stemmer used in Argus as well as a description of the stemming capabilities and existing issues.

5.1 The Farsi Stemmer

The Farsi stemmer implemented in Argus is a basic light stemmer that uses a list of prefixes and suffixes (cf. Table 7) and strips them off from the query and index terms. In addition, the stemmer uses a stop word list consisting of the most frequent non-content words, including affixes that are written separated from the word (cf. Table 8). Stop words are not included in the index and are not used in search queries. There is no in-depth morphological analysis or lexicon used in this system, hence there is no way to ensure that a stem is in fact an actual word in the language. The decisions made in developing the Farsi stemmer are based mainly on the frequency of occurrence of the affixes. The algorithm was designed with the goal of reducing any overstemming or understemming errors, but these decisions often end up involving a trade-off, missing some other important forms.

Table 7 – Farsi suffix and prefix list

Suffixes							Prefixes
یم	را	مان	ات	یان	ان	ها	می
یی	تر	تان	اتی	یانی	انی	های	نمی
ییم	ترین	شان	جات	یانم	انم	هایی	
یید	ی	یش	جاتی	یانت	انت	هائی	
یند		یمان	ین	یانش	انش	هایم	
یست		یمان	ینی	یانمان	انمان	هایت	
ست		یتان	ون	یانتان	انتان	هایش	
		یشان	ونی	یانشان	انشان	هایمان	
						هایتان	
						هایشان	

Table 8 – Farsi stop word list

Words							Affixes	
هم	گفته	شده	حتی	پس	بار	آن	است	ها
همچنان	ما	شما	حدود	پیش	باید	آنان	ایم	های
همچنین	مانند	شود	خواهد	پی	بدون	آنها	اید	هایی
همین	مقابل	طبق	خواهند	تا	بر	آنچه	اند	هائی
هنوز	من	طی	خود	تحت	برابر	آنکه	مان	هایم
هیچ	میان	عین	در	تمام	برای	از	تان	هایت
یا	نخواهد	قبل	درباره	تنها	برخی	اما	شان	هایش
یك	نفر	کرد	دنبال	توسط	بسیار	او	تر	هایمان
یکی	نیز	کرده	دیگر	چند	بسیاری	این	ترین	هایتان
	نیست	کنار	روی	چنین	بعد	اینکه	را	هایشان
	و	کند	زیر	چه	به	اکنون	بی	ام
	ولی	که	سوی	چون	بود	اگر	نمی	ات
	هر	گفت	شد	حتا	بین	با	می	اش
								ای

As can be seen from the list of suffixes in Table 7, only suffixes for content-carrying elements such as nouns and adjectives are included in the system and verbal affixes are not generally listed except for the prefixes. Hence, this stemmer does not conflate the various verbal forms. In addition, some of the more frequent suffix combinations (where several suffixes are attached to each other) are listed such as هایی- (-*hayi* = plural+indefinite) or

انشان– (*-aneshan* = plural+pronoun.3pl). However, this is not a comprehensive list of all possible suffix combinations and certain combinations may be missed in stemming such as انشانیم- (*–aneshanim* = plural+pronoun.3pl+be.1pl) as in دوستانشانیم *dustaneshanim* 'we are their friends'.

In order to avoid conflating many unrelated terms, certain suffixes were not included. As an example, the suffix *–æm* (written as the single character م- 'm') is excluded from the list since stripping it off from all words can lead to high recall with low precision. This suffix can represent the possessive pronoun for the first person singular 'my' (دوستم), the object pronoun 'me' (دوستم دارد), the verb 'to be' for first person singular 'I am' (مریضم), or verbal agreement ending for the first person (نوشتم). But removing it every time it is found attached to a word could lead to overstemming errors since many words end in 'm' – for example, تحریم or مستقیم should not be stemmed to تحری or مستقی since the 'm' ending is not a suffix in these cases.

Another measure used to restrict overstemming is the implementation of a minimum requirement for the length of the stem. In the Farsi stemmer algorithm, a stem has to have at least 4 characters. Hence, words such as رهبران (*ræhbæran*) 'leaders' would be stemmed to رهبر (*ræhbær*) 'leader' but جنگها (*jængha*) 'wars' will not be stemmed to جنگ (*jæng*) 'war'. This was implemented to avoid stemming words like ایران (*iran*) to ایر (*ir*).

The attached and the half-space forms of the affixes (see Table 6) are listed in the prefix and suffix table. Separated affixes, however, are simply treated as stop words and are ignored in search. Hence, the term سفارتها or سفارت‌ها are processed by the stemmer and the plural ها- ending is removed in both cases, resulting in the stem form سفارت 'embassy'. But in the case of سفارت ها where the plural affix is separated by a whitespace, the plural ها is simply ignored and only the singular form سفارت is used in the search.

5.2 Stemming Process

- **Basic stemming**

The Argus Farsi stemmer can successfully strip off the affixes listed in its table of prefixes and suffixes and provide the correct stem for the word[12]. So, given the query term گوسفندان 'the sheep (plural)', the stemmer will correctly stem it to گوسفند by removing the plural ending for animate nouns *'-an'* and will match it against other inflected forms of the word that are also stemmed to the same form such as گوسفنداری 'some sheep', گوسفندهایشان 'their sheep (plural)', گوسفندرا 'the sheep (object)', گوسفند 'sheep (singular)' or گوسفندي 'a sheep'.

Similarly, the query term مسموم 'poisoned', کودک 'child' and بزرگتر 'bigger' will retrieve all documents containing the following variants, respectively:

مسموم – مسمومین – مسمومهایی – مسمومهایش – مسمومست

کودک – کودکلن – کودک‌های – کودک هایش – کودکلنمان – کودکانی – کودکست

بزرگتر – بزرگـتر – بزرگ – بزرگي – بزرگان – بزرگترین – بزرگ ترین – بزرگانی

String searches (i.e., phrases in quotation marks) are handled correctly. Each term is stemmed and searched and adjacent related word forms are retrieved successfully. For example, a search for the term "واکسن آنفلوآنزا" 'flu vaccine' could retrieve documents

[12] One exception is the attached ها (*ha*) suffix (see Section 5.3).

containing واکسنهای آنفلوآنزا or واکسن آنفلوآنزاست, or the query term "پروتئین‌های داخلی" ‘internal proteins’ would return files containing پروتئین داخلی, پروتئینهای داخلی, or پروتئین داخلتر.

Note that the Farsi stemmer does not use a lexicon and therefore any word will be stemmed as far as it contains affixes listed in the system.

- **Normalization**

Farsi documents may use the Farsi characters ک and ی, as well as the Arabic representation of these characters: ك and ي. The Farsi stemmer is designed to process both encodings.

The stemmer also removes any punctuation such as period, comma, exclamation point, or quotation mark when running queries. So, if the word appears next to a punctuation mark in the document, it will still be matched correctly.

- **Stop words**

As already mentioned, frequent non-content words such as از ‘from’ or در ‘in’ are discarded by the stemmer. If they are included in the search query, they will be ignored. It is important to note that, since separated affixes are treated as stop words in this stemmer, they will be ignored even if they are included in the query. Hence, a search for واکسن ها ‘vaccines’ is equivalent to a search for واکسن ‘vaccine’ since the separated ها is eliminated from the representation.

Stop words are also ignored within the text of the document. Thus, the به ‘to’ preposition in the query term "انسان به انسان" ‘person to person’ will be ignored in the search query but in addition, any occurrence of a stop word between the two instances of the word انسان ‘person’ will also be ignored during conflation. This particular search query will match documents containing any of the following phrases:

انسان و انسان – انسان است و اگر انسان – نگاه انسانی و انسان محوری – ما انسانیم، هر انسان می‌خواهد ...

Proximity searches are also affected by the number of intervening stop words between search terms, since the stop words are ignored in computing the distance between the terms. For example, a search for "افزایش جهانی"~2 ‘global increase’ will retrieve a document that includes the sentence chunk قابل **افزایش** است و از مرتفع ترین برج های **جهان**. where the bolded elements are the two search terms (or variants thereof). As can be seen from this result, the stop words است ‘be’, و ‘and’, and از ‘of/from’ are all ignored when determining the distance between the query terms.

5.3 Issues in the Farsi Stemmer

Because the Farsi stemmer is a very basic light stemmer, it fails to process certain elements. Most of these are expected given the design of the system and therefore the analyst should be aware of the behavior of the stemmer when performing searches in order to obtain more accurate results.

- **Unanalyzed suffixes**

One important bug in the Farsi stemmer is that it fails to analyze the plural suffix ها- when it appears attached to a word. Note that the more complex forms of this plural (e.g., هایتان, هایی, های) are stemmed correctly and it is only the attached basic form that is missed by the system. For example, the search term واکسن will match واکسنهای but not واکسنها. Also, a search for the term واکسنها will return only an exact match.

The Farsi system is a basic stemmer that only strips off affixes but does not perform any substitutions. Therefore, affixes that are not simply added to the end or beginning of the word but actually replace another affix are not included in the current system. For instance, the plural suffix گان– (*-gan*) in پرندگان 'birds' substitutes the suffix ه- (*-e*) of the singular form پرنده 'bird'. In order for a stemmer to conflate these two forms of the word without using substitution rules, it should reduce both the singular and the plural forms to the stem form پرند. This can be accomplished if both ه (*-e*) and گان (*-gan*) are listed as suffixes to be stripped off. However, since the stripping off of the ه (*-e*) suffix can give rise to too many errors, these elements were not included in the final design of the system.[13] A similar issue is encountered with the plural suffix ات– (*at*) as in مذاکرات or کلمات replacing مذاکره and کلمه, respectively.

It was earlier described that certain affix combinations are included in the suffix list, such as هایی- (*-hayi* = plural+indefinite) or انشان– (*-aneshan* = plural+pronoun.3pl), while others are omitted. For example, مریضهایشرا 'his/her patients (object)' will not be processed since, although both هایش– (*-hayæsh*) and را– (*-ra*) are listed as suffixes, there is no suffix هایشرا– (*-hayæshra*) in Table 7. Also note that conversational forms of affixes are not included in the list and therefore the term قربانیانمان (*-qorbaniyane<u>man</u>*) 'sacrifices' will be stemmed but not قربانیانمون (*-qorbaniyane<u>mun</u>*).

- **Word-internal changes**

The Farsi stemmer will not be able to relate words whose internal forms have changed considerably. For example, if the query مرض (*mæræz*) is entered, the stemmer will be able to match it to the plural form مرضهایی (*mæræzhayi*) because the latter is formed by adding a suffix *-hayi* to the end of the word. However, the query will not be conflated to the Arabic plural form امراض (*æmraz*) since the latter is not formed by adding an affix to the word but is rather obtained by modifying the internal form of the word.

- **Stem length**

As mentioned above, the Farsi stemmer requires the stem to be at least 4 characters. Thus, words such as رهبران (*ræhbæran*) 'leaders' or مریضهایمان (*mærizhayeman*) 'our patients' will be stemmed but not the terms جنگها (*jængha*) 'wars' or سگان (*sægan*) 'dogs'. Note that this restriction then correctly blocks the system from stemming words such as امکان, فرمان, ایران, or بحران.

- **Overstemming**

Since the Farsi stemmer does not include a lexicon, there may be a number of cases of overstemming where two unrelated words are reduced to the same stem and conflated. For example, واکسین (*vaksin*) 'vaccine' and واکس (*vaks*) 'wax' may not be related semantically although they reduce to the same form, resulting in high recall and low precision (i.e., irrelevant documents returned). Similarly, a search for the last name رضائیان (*rezayian*) would return documents containing رضائی (*rezayi*) or رضائی نژاد (*rezayi nejad*).

[13] Many words in Farsi end in this character such as پرده, آینده, گروه or پادشاه and treating it as a suffix will also affect these forms of the word, potentially creating too many matches. In addition, if this suffix is removed reducing e.g. پرنده to پرند, the system will then fail to conflate it with the form پرندههایی which will be stemmed to پرنده. In other words, without developing a more complex stemmer that performs substitutions, the results would be a trade-off between the correct processing of different words (i.e., if پرندگان is captured, پرندههایی might be missed).

Note that entering a search query such as واکسین NOT واکس is not an option since the result will be null (as both words stem to the same form).

- **Normalization of Diacritics**

Diacritics are not normalized in the Farsi system. This includes the basic vowels such as /æ/, /e/ and /o/, the *tashdid*, the *hamze* (e.g., ارتقاء), and the *keshide* (e.g., کـــه). Thus, if a word is queried without the diacritic, it will not be matched against the same word appearing with a diacritic in a document.

- **Compounds**

Compounds are words that consist of two or more elements that are independent words, such as *loudspeaker* or *high school*. In the Farsi writing system, compounds are often written separated from each other by a whitespace. This raises some recall issues since any word that matches one of the tokens in a compound will be matched and the document in which the compound appears will be retrieved even if there is no semantic relation with the search query. For instance, if the search query is وطن (*vætæn*) 'fatherland', it might be beneficial to also return documents that contain the compound word وطن فروشان (*vætæn forushan*) 'traitors – lit. country-sellers' as this may be relevant to the search. But if the search term is جوجه (*juje*) 'chicken', it is probably not valuable to return documents containing the compounds جوجه تیغی (*juje tighi*) 'hedgehog' or جوجه کباب (*juje kæbab*) 'chicken kabob.' Similarly, a search for جلوگیری (*jelogiri*) 'prevention' will also retrieve documents containing a form of the compound verbs جلوگیری کردن (*jelogiri kærdæn*) 'to prevent' and جلوگیری شدن (*jelogiri shodæn*) 'to be prevented.'

Recommendations

The current Farsi stemmer is a very light stemmer that covers mainly frequent nominal and adjectival affixes using a basic stripping mechanism. The stemmer capabilities can be enhanced by making a few modifications to the design:

- Provide a more complete affix list that includes the various combinations of suffixes as illustrated by the word انقلابیترینهایشانند (*enqelabitærinhayeshanænd*) 'They are the most revolutionary ones among them'
- Add verbal declension forms in the affix list
- Allow substitution operations – this would allow the correct analysis of singular and plural word pairs such as پرندگان/پرنده 'bird/birds' or مذاکرات/مذاکره 'negotiation/negotiations'
- Include diacritic normalization for all vowels, *tashdid*, *hamze* and *keshide* characters
- Fix the bug that fails to stem the attached form of the plural marker ها (*ha*)

These modifications will improve the results of the current stemmer considerably.

Glossary

Affix: Linguistic elements that are attached to a root or stem to form a word. Affixes can be *derivational* or *inflectional*. They can also vary based on the position in which they appear with respect to the root or stem; the most common forms in languages are *prefixes* and *suffixes*.

Character encoding: A code used in computational applications that pairs a sequence of characters (e.g., alphabet of a language) with a sequence of computer-readable representations.

Compounding: Formation of new words by putting together two independent words (e.g., *science fiction*).

Conflation: In stemming, conflation refers to instances where a query term and an index term are identified as one and matched.

Derivation: Variation in the form of a word or a stem to derive a new word. Derviation often changes the class of the word (e.g., from noun to adjective as in *national* or from verb to noun as in *amazement*).

Derivational stemming: Stemming approach that removes not only inflectional affixes but also derivational ones. See **Derivation**.

Index term: Important concept words or phrases in the text of the document that are used as keywords to retrieve documents in search applications.

Indexing: Indexing is used in search engines to collect, parse, and store data to facilitate fast and accurate Information Retrieval.

Inflection: Variation in the form of a word, typically by means of an affix, that expresses a grammatical contrast. Examples include plural on nouns or verbal declension for various teneses.

Information Retrieval (IR): A subdomain of Natural Language Processing focused on searching for documents or for information within documents.

Lemma: The basic dictionary form of a word. Also called a headword, a citation form, or a canonical form.

Lemmatization: Stemming approach that uses deeper linguistic knowledge, such as morphological analyzers or lexicons, to provide the lemma form of words.

Lexicon: Electronic dictionary.

Light stemming: Stemming approach that strips off a small set of affixes. It may perform limited substitutions but does not provide in-depth linguistic analysis.

Morphology: Field of linguistics that studies the internal structure of words.

Normalization: Process of transforming text into a consisten and/or standard form, especially useful when matching characters or words in a text.

Precision: Percentage of documents correctly retrieved by a search system.

Prefix: An affix that attaches to the beginning of a root or stem (e.g., **un**do).

Query syntax: The set of rules that define the format in which a search query should be entered.

Query term: String used to perform a search. Query terms are often stemmed in search applications.

Recall: Percentage of relevant documents retrieved as relevant by a search system.

Root: The primary meaning part of a word that cannot be reduced in form any further (e.g., dis**rupt**).

Root-and-pattern morphology: See **templatic morphology**.

Stem: The part of a word that is common to all its inflected variants, obtained by removing all prefixes and suffixes from the word. The definition and size of a stem may vary, however, depending on the Stemming algorithm used.

Stemmer: A computational tool used to perform stemming. Stemmers vary based on the sets of affixes that they remove and the level of linguistic knowledge they apply**.** See **Light Stemming**, **Lemmatization**, and **Derivational Stemming**.

Stemming: The process of cutting off affixes from words in a text.

Stop words: Frequent non-content words in text that are filtered out prior to performing search.

Stoplist: A list containing stop words for a language.

Suffix: An affix that attaches to the end of a root or stem (e.g., read**ing**).

Templatic morphology: Languages where words are based on a template or root form and related words are derived by applying different patterns to the template, typically be inserting different vowels inbetween the characters of the root. Hebrew and Arabic are such languages. This is also known as **root-and-pattern morphology**.

Transliteration: The representation fo the written characters of an alphabet by those of another, typically using ASCII or latin-script characters.

Acknowledgments

Azar Ulrich (MITRE) was instrumental in testing the Lucene Query Syntax, and Megan Ward (MITRE) provided invaluable feedback in the writing of this technical report.

Selected References

Halácsy, Péter (2006). Benefits of Deep NLP-based Lemmatization for Information Retrieval. In *Working Notes for the Cross Language Evaluation Forum (CLEF 2006) Workshop*.

Hull, David A. 1998. Stemming algorithms: A case study for detailed evaluation. In *Journal of the American Society for Information Science and Technology,* Volume 47:1, pp. 70-84.

Jurafsky, Daniel, and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall.

Kraaij, Wessel and Renée Pohlmann. 1996. Viewing Stemming as Recall Enhancement. In *Proceedings of SIGIR'96*. Zurich, Switzerland.

Larkey, Leah S., Lisa Bellesteros, and Margaret E. Connell. 2002. Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis. In *Proceedings of SIGIR'02*, Tampere, Finland.

Piotrowski, Michael. 1998. NLP-Supported Full-Text Retrieval. Clue Technical Reports, Number 3. Friedrich-Alexander-Universität Erlangen-Nürnberg Institut für Germanistik. Germany.

Soudi, Abdelhadi, Violetta Cavalli-Sforza and Abderrahim Jamari. 2001. A Computational Lexeme-Based Treatment of Arabic Morphology. In *Proceedings of the Arabic Natural Language Processing Workshop, Conference of the Association for Computational Linguistics (ACL 2001)*, Toulouse, France.

Taghva, Kazem, Russell Beckley, and Mohammad Sadeh. 2003. A Stemming Algorithm for the Farsi Language. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05)*. IEEE Computer Society.

What is Stemming? [http://www.comp.lancs.ac.uk/computing/research/stemming/general/]
Wikipedia. http://en.wikipedia.org

Index