

Farsi Searching and Display Technologies

**Kazem Taghva, Ron Young, Jeffrey Coombs
Russell Beckley, Mohammad Sadeh, Ray Pereda**

Information Science Research Institute
University of Nevada, Las Vegas
Las Vegas, NV 89154-4021

Abstract

In this paper, we report on our ongoing research for the development of a Unicode-based search engine for Farsi. The activities consist of an I/O subsystem, Farsi stemmer, test collection preparation, and the search engine itself.

This engine is intended to be independent of the operating system platform using no special hardware or software. We are further planning to tune the system for other languages with Arabic related scripts.

1 Introduction

Farsi is the official language of Iran and one of the two official languages in Afghanistan (Pashto is the other language). In addition, Farsi is spoken in Tajikistan and parts of Uzbekistan [8]. There are more than two million Iranians living in the United States, and the Farsi language is taught at many universities and institutions in North America and Europe [3]. In recent years, many web sites have appeared that provide information in Farsi. In particular, most of Iranian newspapers and magazines have official websites with daily articles in Farsi. As more Farsi materials become available on the web, it is evident that search tools need to be developed to deal with Farsi information access needs.

The recent methodologies in Cross Language Retrieval and Machine Translation are one approach to addressing the non-English information access needs. Farsi is typically omitted from most of these methodologies due to lack of standards for Farsi fonts and input/display technology. For example, most of the Farsi websites render their information in specialized fonts or images of text. The lack of standards and the need to use the Internet without using an Arabic-based alphabet has even convinced a community of Iranians to develop a Latin-based script known as EuroFarsi [4].

In the early days of our project, we noticed that there was not much information that could help us develop our search engine. For example, there was no known Farsi stemmer, Farsi Collection, or Farsi stopword list. Also, we needed an input/display method that would allow us to enter Farsi query words in a Latin-based operating sys-

tem without any special software or hardware. It was further necessary to have a standard character encoding for text representation and searching. Most of our design decisions in this project were influenced by the above-mentioned problems.

This paper is a description of our ongoing activities in the development of a Farsi Retrieval Engine. Section 2 describes an input/output system for Farsi. Section 3 explains the development of the Farsi keyboard applet. Section 4 shows the architecture of our system. Section 5 gives a short overview of our stemmer and stopword list. Our text collection is presented in section 6. Section 7 describes the search engine based on both traditional methods and statistical language modeling [12, 20, 15]. Section 8 is the conclusion and future work.

2 Collection Level I/O

When we started compiling the test collection for our Farsi project, one problem became immediately obvious: "the anarchy of fonts." In processing the collection, we encountered documents encoded with the following:

Standard encodings:

- Mac OS Farsi
- CP 1256 (Microsoft Windows)
- ISO 8859-6.16
- UCS/UTF8 (Unicode)
- ISIRI 3342 & 2901 ([6])

Proprietary encodings:

- e.g. *Persian Nimrooz*

Other encodings:

- Adobe Distiller Embedded Fonts
- Images of text

By far, the most common encoding represented in our collection was *Persian Nimrooz* [5]. A conversion table between this encoding and UCS2 (Unicode) was developed manually. The standard encodings also have Unicode conversion tables widely available. Because of the relatively low occurrences of the other encodings, docu-

ments with these encodings were removed from the collection.

Processing the collection consists of interpreting the raw HTML document and extracting the text along with any corresponding '' tags. Once the text is extracted and converted to Unicode, it is then written as a UCS2 binary file.

A key requirement for our Farsi project was to allow the input and display of native Farsi content without requiring any special hardware or additional OS software capabilities. To satisfy this requirement the system provides the following capabilities:

- a web-browser based keyboard applet for input
- if the web-browser has the ability to process and display Unicode content, it will be used
- if the browser cannot display Unicode content, an auxiliary process will be invoked to render the Unicode content into a portable bitmap image with associated HTML to display the image in the browser.

Another area of difficulty that we encountered is that the presence of whitespace used to separate words in the document is dependent on the display geometry of the glyphs. Since Farsi is written using the Arabic script (a cursive form), each character can have up to four different display glyphs. These glyphs represent the four different presentation forms:

- isolated*: the standalone character
- initial*: the character at the beginning of a word
- medial*: the character in the middle of a word
- final*: the character at the end of a word

We found that depending on the amount of trailing white space following a final form glyph, a space character may or may not be found in the text. This situation came to light when our subject matter experts were developing our test queries. We found that since the glyphs used to display the final form of characters had very little trailing space, they were manually adding space characters to improve the look of the displayed queries.

3 Keyboard Applet

The keyboard applet is written in javascript. We also developed a java version but decided on javascript to minimize the support issues related to the Arabic script capabilities of the java virtual machines included in the various browsers. The applet displays a Farsi keyboard image with the ability to enter characters from both the keyboard and mouse. The applet also handles character display conversion and joining of the input data.

The keyboard layout is based on the ISIRI 2901:1994 standard layout as documented in an email by Pournader [14]. Figure 1 shows the keyboard applet being used to define test queries.



Figure 1: Example use of the keyboard applet

Display of the input data is normally performed by using the preloaded glyph images. However, if a character has not been preloaded, it can be generated on the fly. Most of the time, these generated characters are "compound" characters. Farsi (and other Arabic script languages) may use "compound" characters which are a combination of two or more separate characters. For example, the rightmost character of خُرما, the Farsi word for "date" (that is, the fruit), is a combination of a khe with a damma.

4 General System Architecture

As part of this project, we wanted to have a robust and easily extendible environment for conducting information retrieval experiments. To this end, we have developed a modular framework which allows the ad hoc loading of extended commands into a core command line interpreter.

The core command line interface provides commands to load/unload extended command libraries, user defined symbol management, and an embedded HTTP web server. The symbol table management routines are based on the ones in the *PCCTS 1.3mr33 toolkit* written by Parr and maintained by Moog [11]. The embedded HTTP web server that we use is *libHTTPD* by Hughes Technologies [16]. The *cli* command line interface processing routines uses the *cmd* command parser

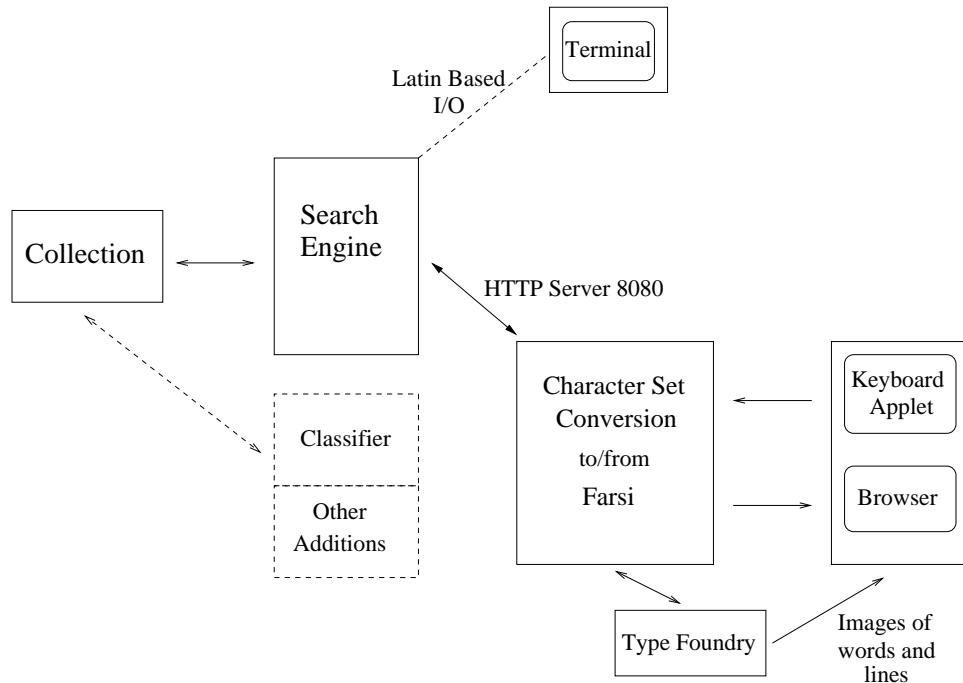


Figure 2: Farsi Search Engine Architecture

library by Columbia University [18].

By using an embedded HTTP server, we can handle non-Latin scripts easily. The search engine performs all processing with Unicode encoded documents. When an item is displayed, the Unicode encoding is sent to the embedded HTTP server. If the browser supports Unicode encoded fonts, the item is displayed using the browser. If the browser doesn't support Unicode fonts, the item is converted by the "type foundry" into a series of portable network graphics (PNG) images. We use the *freetype* engine by the Freetype Project to render the Unicode script into the images [17]. These images are then sent to the browser for display. Figure 2 depicts an overview of the system.

5 Stemmer and Stopword List

The Farsi stemmer identifies common Farsi affixes in Farsi words. Conditionally, it removes the affixes, producing an effective stem.

The Farsi stemmer is similar to the Porter stemmer [13]. For example, both are based on the morphological rules of their respective languages. Both stemmers match words with a set of suffixes and use multiple phases to conform to the rules of suffix stacking. Also, they both enforce lower bounds on how much information a stem must retain. However, there are differences. For example, the Porter stemmer counts substrings of consecutive consonants and vowels, estimating the information content, before deciding to remove a suffix. In Farsi, many spoken vowels are not written, so the stemmer cannot count them. Therefore, the Farsi stemmer uses stem length to define a lower

bound on information content (in the current version, minimum stem length = 3). Another difference is that the Farsi stemmer identifies prefixes, unlike the Porter Stemmer.

The stemmer is aggressive, removing every terminal substring that matches a known suffix, sometimes removing part of the root word. However, it rarely removes enough of a root to overconflate.

The stemmer attempts to match a suffix to a word ending by feeding the word to a finite state machine. A word may be fed the machine as many as three times, identifying as many as three suffixes stacked on the same root. The state machine reads the word backwards, i.e., it first reads the final character (the last character in reading order), then proceeds toward the first character. If the machine halts in an accepting state, we know that a suffix matches the end of the word and that the input word is long enough to retain the minimum stem length after removing that suffix. The stemmer associates each accepting state with a specific suffix and a suffix class such as *plural*.

For example, suppose the input is *نمیرفتیم* ("we were not going"). The finite state machine will identify the suffix *یم* as the longest matching suffix. The final state of the machine tells the stemmer that the suffix belongs to the *verb* class.

The suffix classes determine the modifications the stemmer makes and the additional affixes the stemmer attempts to match. The general rules for modifying words according to suffix class are:

Verb Remove the identified suffix. In accordance with the rules of prefix stacking, remove as many prefixes

Farsi Word	English Equivalent
آن	that
چرا	why
آنجا	there
است	is
باید	must
بود	be
با	with/by

Figure 3: Potential Farsi Stopwords

as possible while retaining sufficient stem length. The expression $\{ناب\}\{می\}$ describes the language of relevant prefixes.

Possessive Remove the identified suffix. Feed the remainder to the finite state machine to check for non-possessive noun suffixes.

Plural Remove the identified suffix. Feed the remainder to the finite state machine to check for non-possessive, non-plural noun suffixes.

All other suffix types Remove the identified suffix.

Consider the previous example, *نمیرفتیم*. After identifying the suffix and suffix class, the stemmer will remove the suffix *یم*, leaving *نمیرفت*. Then it will look for the prefix *ب*, and not find it. It will then find the prefix *ن*. It will check the word length, and remove the prefix, leaving *میرفت*. It will then find the prefix *می*, check length, and remove the prefix. It outputs *رفت*, which is the infinitive “to go”, with the terminal *ن* removed.

Now, suppose the input is *کتابهام*, “my books.” The finite state machine will identify the suffix *م*, and remove it, leaving *کتابها*. Because *م* is a possessive suffix, the stemmer feeds the remainder to the finite state machine to look for more noun suffixes of the same word. Now, the finite state machine will identify the suffix *ها*, and remove it, leaving *کتاب*. Because *ها* is a plural suffix, the stemmer feeds the remainder to the finite state machine to look for more noun suffixes. This time, the finite state machine halts in a rejecting state, signifying an absence of more suffixes, so the stemmer returns *کتاب*, “book.”

In addition, we are using our Farsi document collection to generate a list of stopwords. Stopwords are identified by frequency analysis and expert judgment. Figure 3 shows potential stopwords from the list.

6 Document Collection

Our document collection consists of 1850 documents and 60 queries. These documents were collected within a six month period from many websites. Some of these websites originate in Iran and they typically represent electronic versions of popular Iranian newspapers and magazines. The rest of the articles are from websites originating in the U.S. or in Europe. These documents mainly cover topics such as politics, economic issues associated

آیا در اثر عدم رعایت ضوابط بهداشتی در ایران سلامت مردم درخطر است؟
Does the lack of health standards in Iran endanger the population?

Figure 4: An Example Query

with oil, social problems, and historical changes in the Middle East.

The sixty queries were designed by native speakers of Farsi familiar with these documents. An example query and its translation appears in Figure 4. These queries were input using our keyboard applet for the purpose of relevancy judgment gathering.

We also built a web-based application to display the queries and the articles side-by-side for our Farsi experts. After an expert reads the article, he makes binary decisions as to the relevance of the document to each query. No relevancy ranking is assessed.

7 Search Engine

Our search engine will contain several methodologies because we plan to study their effectiveness in retrieving Farsi language texts. Initially two methods will be applied: the vector space and the probabilistic.

In the well-known vector space model, documents and queries are represented as n -dimensional vectors of term weights. For a given query vector Q documents are ranked in terms of their similarity to the query as determined by the cosine measure:

$$\cosine(Q, D_d) = \frac{Q \cdot D_d}{\|Q\| \|D_d\|} = \frac{\sum_{t=1}^n w_{q,t} \times w_{d,t}}{W_q W_d}$$

D_d represents the vector of a document d . The inner product $Q \cdot D_d$ is interpreted in the standard way as the sum of the products of the weights of corresponding terms in the query and document, $w_{q,t}$ and $w_{d,t}$ respectively. Weights are determined using the usual $tf \times idf$ scheme. The values W_q and W_d are the Euclidean lengths of the query and a document [19].

Our Farsi retrieval engine will also use probabilistic techniques. In a probabilistic search engine, documents are ranked by the probability that given a query q , document d should be retrieved. Using Bayes theorem and dropping the constant probability for the fixed query from the denominator, we have:

$$p(d | q) \propto p(d)p(q | d) \quad (1)$$

Recent studies have shown that the specific probabilistic approach called *statistical language modeling* has proven more effective than others on English collections [12, 20]. Some success has been shown for Arabic texts as well [9]. Statistical language modeling rejects the assumption of other probabilistic approaches that there is an underlying distribution model which the data should fit. For example, Bookstein and Swanson assumed that terms would

occur in documents with a Poisson distribution [1]. Instead, statistical language modeling uses non-parametric techniques to estimate $p(q|d)$.

Based on the concept of a language model developed for speech recognition, a query is viewed as a randomly generated set of terms. The expression $p(q|d)$ is then understood as the probability that certain terms will occur in a query *given* a particular model of the documents [12, 20]. Of course, queries are not typically generated randomly, but from the “perspective” of the document model, how the query is created is unknown and thus “appears” to be random.

Since $p(d)$ is typically assumed to be uniform in equation (1), the expression $p(q|d)$ alone requires an estimate. The basic approach in statistical language modeling is to begin with the maximum likelihood estimate,

$$p_{mle}(t|d) = \frac{tf_{t,d}}{DL_d}$$

where $tf_{t,d}$ is the number of occurrences of a term t in a document d , commonly called *term frequency*, and DL_d is the document length of a document d , defined as the total number of term occurrences in d , that is, $\sum_t tf_{t,d}$. However, the maximum likelihood estimate has two primary shortcomings in this context. First, the estimate will assign a probability of zero to a document if one or more query terms are not present, and second, documents only contain sparse data [12, 20, 15].

To solve these problems, *smoothing techniques* are employed. The general form of a smoothing technique can be expressed as

$$p(t|d) = \begin{cases} p_s(t|d) & \text{if } t \text{ occurs in } d \\ \alpha_d p(t|C) & \text{otherwise} \end{cases}$$

where $p_s(t|d)$ is the smoothed version of the maximum likelihood estimate, $p(t|C)$ is the “collection language model” and coefficient α_d determines the amount of probability transferred to terms not in the query. A number of techniques have been proposed to define these expressions [2, 20, 21, 12]. We plan to investigate several of these to see how they impact effectiveness of retrieval on Farsi text, especially those which move beyond the independence assumption of the unigram model [15, 10].

8 Conclusion and Future Work

The explosion in availability of Farsi websites and the eventual electronic conversion of Farsi literature are the driving forces behind our project. We believe standard fonts, operating system independence, and input/display technology are critical factors in future development of search tools for Farsi particularly. Because of the close relationship between Arabic and Farsi, our research can indirectly benefit Arabic as well. For example, over 40% of all words in Farsi are Arabic, and thus one should expect common research issues between our stemmer and an Arabic stemmer [7].

Future work includes performance testing of our stemmer and search engine. We are also planning to make detailed reports available on our input/display technology, stemmer, stopword list, and search engine implementation at www.isri.unlv.edu.

References

- [1] Abraham Bookstein and Don Swanson. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, pages 312–318, 1974.
- [2] Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard Univ., 1998.
- [3] Center for Advanced Research on Language Acquisition (CARLA). Less commonly taught languages (LCTL) project. <http://carla.acad.umn.edu/LCTL/>.
- [4] Manou & Associates Inc. Eurofarsi experiment. <http://www.iranonline.com/eurofarsi/experiment.html>.
- [5] Neda Rayaneh Institute. Persian nimrooz. <http://neda.net.ir/downloads/fonts.shtml>.
- [6] ISIRI. Institute of standards and industrial research of Iran. <http://www.isiri.org/>.
- [7] Shereen Khoja. Stemming. <http://www.comp.lancs.ac.uk/computing/users/khoja/index.htm#stemming>.
- [8] Tore Kjeilen, Sidahmed Abubakr, and D. Josiya Negabban. Encyclopedia of the orient. <http://i-cias.com/e.o/>.
- [9] Leah Larkey and Margaret Connell. Arabic information retrieval at umass in trec-10. In *The Tenth Text Retrieval Conference, TREC 2001 NIST Special Publication 500-250*, pages 562–570, 2002.
- [10] D. H. Miller, T. Leek, and R Schwartz. A hidden markov model information retrieval system. In *SIGIR'99*, pages 214–221.
- [11] Terence J. Parr and Thomas Moog. Pccts 1.3mr33 toolkit. <http://www.polhode.com/pccts.html>.
- [12] Jay Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR'98*, pages 275–281, 1998.
- [13] M.F. Porter. An algorithm for suffix stripping. *Program*, 14, 1980.

- [14] Roozbeh Pournader. Farsi keyboard. <http://lists.sharif.edu/pipermail/persiancomputing/2000-August/000129.html>.
- [15] Fei Song and W. Bruce Croft. A general language model for information retrieval. In *SIGIR'99*, pages 279–280, 1999.
- [16] Hughes Technologies. libhttpd. <http://www.Hughes.com.au>.
- [17] David Turner, Robert Wilhelm, and Werner Lemberg. Freetype project. <http://freetype.sourceforge.net>.
- [18] Columbia University. Cli processing routines. <ftp://ftp.cc.columbia.edu/mm/mm-ccmd-0.91.tar>.
- [19] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes: Compressing and indexing documents and images*. Morgan Kaufmann, 2nd edition, 1999.
- [20] ChengXian Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR'01*, pages 334–342, 2001.
- [21] ChengXiang Zhai and John Lafferty. Two-stage language models for information retrieval. In *SIGIR'02*, pages 49–56.