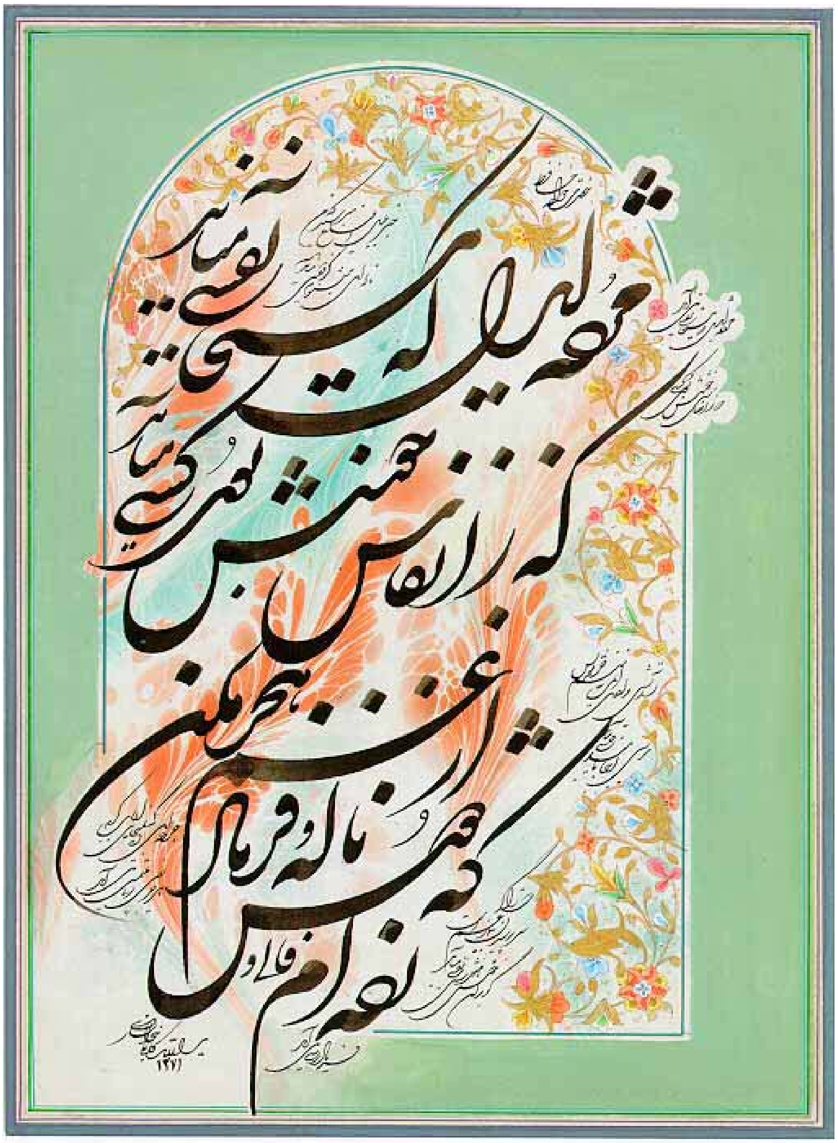
Department of Linguistics
Stockholm University
January 2004

FarsiSum

A Persian text summarizer   خلاصه نویس متون فارسی

Master Thesis   20 credits
Nima Mazdak   nima.mazdak@comhem.se



مژده ای دل که مسیحا نفسی می‌آید   که ز انفاس خوشش بوی کسی می‌آید   (حافظ)

Abstract

FarsiSum is an attempt to create a text summarization system for Persian, based upon SweSum. SweSum is an automatic text summarizer for Swedish that is also available for Norwegian, Danish, Spanish, English, French and German texts. SweSum is supposed to work for any language in a so called *generic mode*. The program in the *generic mode* uses only general extraction algorithms in the summarization, and these do not refer to any language specific information.

The aim of this study is

- To investigate how SweSum behaves when applied to a text which is not transcribed in a Roman writing system?
- To design and implement FarsiSum by using the methods and algorithms implemented in the SweSum project.
- To evaluate FarsiSum.

A live version of the automatic summarizer FarsiSum can be found at http://www.nada.kth.se/iplab/hlt/farsisum/index-farsi.html

TABLE OF CONTENTS

1. Introduction

1.1 Purpose

The aim of this thesis is to implement and evaluate a Persian text summarizer by using the techniques and algorithms developed in the SweSum project and by adding some new modules for handling of documents containing Unicode characters, since SweSum supports only ASCII characters.

1.2 Method

FarsiSum is a HTTP client/server application programmed in Perl. It uses modules implemented in SweSum and a Persian stop-list in Unicode format. The stop-list is a file including the most common verbs, pronouns, adverbs, conjunctions, prepositions and articles in Persian. The words not included in the stop-list are supposed to be nouns or adjectives. The idea is that nouns and adjectives are meaning-carrying words and should be regarded as keywords.

The application has been tested in a field test by several Persian speakers who had access to a text document and three different summaries generated by different methods. The methods used in the summaries were: with the stop-list enabled, with the stop-list disabled, and in the *generic mode* implemented in SweSum.

2. Background

Automatic text summarization is a technique which automatically creates a summary of a text. A text summarization device takes a text and produces a summary of the most important parts of the original text.

There are two major types of text summary: ***abstract*** and ***extract***.

Extract Summarization

The summarized text is extracted from the original text on a statistical basis or by using heuristic methods or a combination of both. The extracted parts are not syntactically or content wise altered.

Abstract Summarization

The summarized text is an interpretation of the original text. The process of producing it involves rewriting the original text in a shorter version by replacing wordy concepts with shorter ones. For example, the phrase "He ate banana, orange and pear" can be summarized as "He ate fruit" i.e. to produce a more general concept 'fruit' two or more topics, *orange*, *banana* and *pear* are fused together. Implementation of abstract methods requires symbolic world knowledge which is too difficult to acquire on a large enough scale to provide a robust summarization.

Automatic text summarization has been under development for many years, and there has recently been much more interests in it due to the increased use of Internet. For example this technique can be used:

- To summarize news to SMS or WAP[1]-format for mobile phone/PDA[2].

[1] Wireless Application Protocol

[2] Personal Digital Assistance

- To let a computer synthetically read the summarized text. Written text can be too long and tedious to listen to.
- In search engines, to present compressed descriptions of the search results (see the Internet search engine Google).
- In keyword directed news subscriptions of news which are summarized and sent to the user (see Nyhetsguiden in Swedish)
- To search in foreign languages and obtain an automatically translated summary of the automatically summarized text.

Many different approaches have been proposed for text summarization. Luhn (1959) first utilized word-frequency-based rules to identify sentences for summaries, based on the intuition that the most frequent words represent the most important concepts of the text. Edmundson (1969) incorporated new features such as *cue phrases*, *title/ heading words*, and *sentence location* into the summarization process, in addition to *word frequency*. The ideas behind these older approaches are still used in modern text extraction research.

2.1 Summarization process

According to Lin and Hovy (1997) there are three different steps in performing text summarization: *topic identification*, *interpretation* and *generation*.

2.1.1 Topic Identification

The goal of topic identification is to identify only the most important (central) topics in the text. Topic identification can be achieved by various techniques, including methods based on position, cue phrases, concept counting, and word frequency.

- In some text genres, certain *positions* of a text such as the title, the first sentence in a phrase, etc tend to carry important topics.
- *Cue phrases* such as '*in summary*', '*the best*', '*in conclusion*', '*the most important*', '*this article*', '*this document*', etc can be good indicators of important content.
- Words which are more frequent in a text (*Word Frequency*) indicate the importance of content, unless they are function words such as determiner and prepositions.
- The topics are identified by counting *concepts* instead of words (*Concept Frequency*).

2.1.2 Interpretation

For extraction summaries, the central topics identified in the previous step are forwarded to the next step for further processing. For abstract summaries however, a process of interpretation is performed. This process includes merging or fusing related topics into more general ones, removing redundancies, etc.

Example: He *sat down*, *read the menu*, *ordered*, *ate* and *left*. → He visited the restaurant.

As mentioned earlier it is difficult to implement such systems with good results.

2.1.3 Generation

The third step in the summarization process according to Lin and Hovy (1997) is the generation of the final output (summary). This step includes a range of various generation methods from very simple word or phrase printing to more sophisticated phrase merging and sentence generation. The following methods might be used:
Extraction: The terms or sentences selected in the first step of summarization are printed into the output.
Topic lists: Lists of the most frequent keywords or interpreted fuser concepts are printed into the output.
Phrase concatenation: Two or more phrases are merged together.
Sentence generation: A sentence generator produces new sentences. The input to the generator is a list of fuser concepts and their related topics.

2.2 Summarization Methods and Algorithms

Generating a high quality summary requires NLP[3] techniques such as discourse analysis, world knowledge inference, semantic parsing, and language generation that are still under research. As a result, most of the current automated text summarization systems produce *extracts* instead of *abstracts*. An *extract* is a collection of the important sentences in a document, reproduced verbatim (Lin 1999). There are other methods such as the *Local Salience Method*, a method developed by Boguraev et al. (2001) which extracts phrases rather than sentences or paragraphs.

In general, there are three major problems in creating *extract* summaries:

- Selecting the most important sentences.
- Generating coherent summaries.
- Repetitive information (redundancy) in the summary.

Much research has been done on techniques to identify the most important sentences that summarize a text document. Sentence extraction methods for summarization normally work by scoring each sentence as a candidate to be part of summary, and then selecting the highest scoring subset of sentences.
Some features that often increase the candidacy of a sentence for inclusion in summary are listed below:
Baseline: Each sentence is scored according to its position in the text. For example in the domain of newspaper text, the first sentence gets the highest ranking while the last sentence gets the lowest ranking.
Title: Words in the title and in following sentences are important and get high score.
Word Frequency (WF): Open class words[4] (content words) which are frequent in the text are more important than the less frequent. Sentences with keywords that are most often used in the document usually represent the themes of the document
Indicative Phrases: Sentences containing key phrases like “this report …“.
Position Score: The assumption is that certain genres put important sentences in fixed positions. For example, newspaper articles have the most important terms in the first four

[3] Natural Language Processing
[4] An open class word is a word to which an independent meaning can be assigned. Nouns, verbs and adjectives are commonly considered open class words.

paragraphs while technical documents have the most important sentences in the conclusion section.
Query Signature: Users often have a particular topic in mind when they request summaries. The query of the user affects the summary in that the extracted text will be compelled to contain these words. Normalized score is given to sentences depending on the number of query words they contain.
Sentence Length: The score assigned to a sentence reflects the number of words in the sentence, normalized by the length of the longest sentence in the text.
Proper Name: Sentences which contain proper nouns are scored higher.
Average lexical connectivity: The number of terms shared with other sentences. The assumption is that a sentence sharing more terms with other sentences is more important.
Numerical data: Sentences containing numerical data are scored higher than ones without numerical values.
Proper name: Proper names, such as the names of persons and places, are often central in news reports and sentences containing them are scored higher.
Pronoun: Sentences that include a pronoun (reflecting co-reference connectivity) are scored higher.
Weekdays and Months: Sentences that include days of the week and months are scored higher.
Quotation: Sentences containing quotations might be important for certain questions from user.
First sentence: The first sentence of each paragraph is the most important sentence.

Despite the usefulness of the sentence extraction methods mentioned above, they cannot alone produce high quality extracts.
In addition, using word-level techniques such as *word frequency* have been criticized in several respects for the following reasons:

- Synonymy: one concept can be expressed by different words. For example *cycle* and *bicycle* refer to same kind of vehicle.
- Polysemy: one word or concept can have several meanings. For example, *cycle* could mean *life cycle* or *bicycle*.
- Phrases: a phrase may have a meaning different from the words in it. An *alleged murderer* is not *a murderer* (Lin and Hovy 1997).

Furthermore, extracting sentences from a text with the statistical keyword approach often causes a lack of cohesion. In order to improve the quality of the final summary, these methods are normally combined with other techniques. Some of these algorithms and methods are presented in the following section.

2.2.1 Sentence Selection Function for Extraction

Text summarization systems normally employ several independent modules to assign a score to each sentence, and then combine the scores each sentence has been assigned, in order to create a single score for each sentence. However it is not immediately clear how these different scores should be combined. Various approaches have been described in the literature. Most of them employ some sort of combination function, in which coefficients assign various weights to the individual scores, which are then summed. The coefficients are normally language and genre-dependent.

Simple combination function is a straightforward linear combination function, in which the coefficients of the above parameters (*title*, *numerical data*, etc.) are specified manually, by experimentation. The sentence score is calculated then according to the following formula:

Sentence score = $\sum$ **C**j**P**j (**C**oefficient, **P**arameter, j = 1...n, n = nr of parameters).

SweSum uses a *simple combination function* for evaluation of sentence scores and assigns the coefficients manually. For example, the value 1000 is assigned to the *baseline* parameter.
Decision tree combination function is a combination function, in which the coefficients are automatically specified using machine learning algorithms.

2.2.2 Knowledge-Based Concept Counting

Lin (1995) presented a new method for automatically identifying the central ideas in a text, based on a knowledge-based concept counting paradigm. According to Chin-Yew Lin, the word frequency methods recognize only the literal word forms and miss conceptual generalizations. For example the main topic in the sentence "*John bought some vegetables, fruit, bread, and milk.*" should be *groceries*, but we cannot make any conclusion about the topic of this sentence by using word counting methods. Word counting methods miss the important concepts behind those words: *vegetables*, *fruit*, *etc*. relates to *groceries* at the deeper level of semantics.
Concepts are generalized using *concept generalization taxonomy* (WordNet[5]). Figure 1 (a) shows a possible hierarchy for the concept *Computer Company*. According to this hierarchy, if we find NEC, Compaq, IBM, in a text, we can infer that the text is about *Computer Company*. And if in addition, the text also mentions *Nokia*, *Ericsson* and *Motorola*, it is reasonable to say that the topic of the text is related to *technical company*. Using a hierarchy, the question is now how to find the most appropriate generalization in the taxonomy hierarchy. According to this method the nodes in the middle of the taxonomy are most appropriate, since the very top concept is always a *thing* (everything is a *thing*) and using the leaf concepts give us no power from generalization.
Ratio (*R*) is a way to identify the degree of summarization. The higher the ratio, the more it reflects only one child. The ratio is defined with the following formula:

R = MAX (W) / SUM (W) where
W = the weight of all the direct children of a *concept*.

The *weight* of a parent concept is defined as the frequency of occurrence of a concept C and its sub concepts in a text.
For example the Ratio (*R*) for the parent's concept in the Figure 1 (b) is 6/ (1+1+6+1+1+) = **0.6** while it is **0.3** in the Figure 1 (c).

[5] WordNet® is an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets.

For determination of the degree of generalization, the *branch ratio threshold* (*Rt*) is defined. *Rt* serves as a cutoff point for the interestingness. If a concept's ratio *R* is less than *Rt*, it is an interesting concept.
For example consider Figure 1, in case (b) if the *Rt* = **0.4**, we should choose *Compaq* as the main topic instead of its parent since *Rt* < *R*. In contrast, in case (c) we should use the parent concept *Computer Company* as the concept of interest.

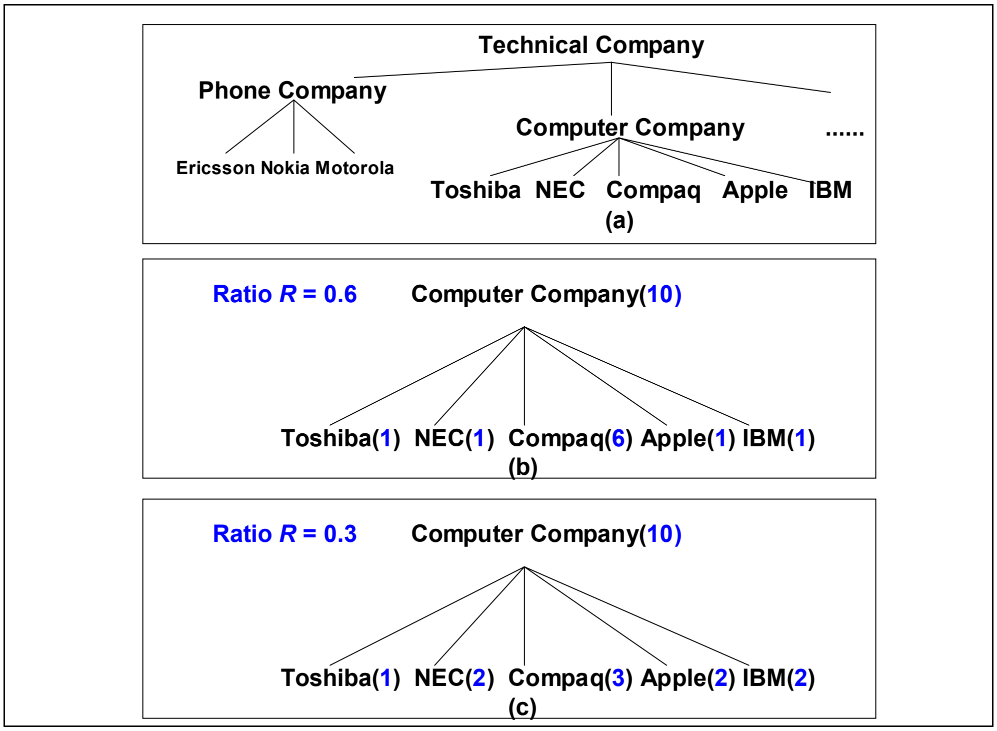


Figure 1: A sample hierarchy for *Computer Company*

2.2.3 Lexical chain methods

Word frequency is a good indicator of important content, but it does not consider the relation between words in different parts of a text. Extracting sentences from a text based on *word frequency* often causes a lack of cohesion. To address the problem, more sophisticated but still essentially statistical methods of sentence or paragraph extraction, such as lexical chains, have been investigated. A lexical chain is a set of words in the text that are related to each other (Brunn et al., 2001). The relation between words is found using lexical lists taken from a thesaurus or a computerized lexicon, such as for example WordNet. By using lexical chains, we can statistically find the most important concepts by looking at structure in the document rather than deep semantic meaning. All that is required to calculate these is a generic knowledge base that contains nouns, and their associations.

A general algorithm for computing a chain can be presented in the following way:

- Select a set of candidate words from the text.
- For each of the candidate words, find an appropriate chain to receive a new candidate word, relying on a relatedness criterion among members of the chains and the candidate words.
- If such a receiving chain is found, insert the candidate word in this chain and update it accordingly; else create a new chain.

Chains are scored according to a number of heuristics: their length, the kind of relation between their words, the position in the text where they start, etc. Sentences that are most connected to lexical chains are extracted.

One of the drawbacks of lexical chains is that they are insensitive to the non-lexical structure of texts, such as their rhetorical, argumentative or document structure. For example, they don't take into account the position of the elements of a chain within the argumentative line of the discourse, sometimes not even within the layout- or genre-determined structure of the document. Therefore, the relevance of chain elements is calculated irrespective of other discourse information.

.

2.2.4 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a statistical, corpus-based text comparison mechanism that was originally developed for the task of information retrieval, but in recent years it has shown remarkably human-like abilities in a variety of language tasks (Wiemer 1999). LSA can be used in sentence extraction methods in order to reduce the redundancy in the summary. Anti-redundancy was not explicitly accounted for in earlier systems, but forms a part of most of the current summarizers. Anti-redundancy scoring is computed dynamically as the sentences are included in the summary, to ensure that there is no repetitive information in the summary.

Gong and Liu (2001) present and compare sentence extraction methods using LSA and relevance measures. Use of relevance measures is a standard IR practice. The document is decomposed into sentences where each sentence is represented by a vector of words that it is composed of. The entire document itself is represented as a single vector of word frequencies. The word frequencies are weighted by local word weights and global word weights and these weighted vectors are used to determine the relevance. A sentence S that has the highest relevance is selected from the document and included in the summary and then all the terms contained in this sentence are removed from the document vector. This process is continued till the number of sentences included into the summary has reached a pre-defined value. The sentence that is most relevant to the document vector is the one that conveys the maximum information. So in each step, sentences that convey maximum information with regard to the current sentence vector are selected. The process of removing words from the document vector ensures that redundant information is not included in the summary. The latent semantic analysis approach uses a matrix decomposition mechanism called the *Singular Value Decomposition* (SVD) to generate an index matrix (Landauer et al. 1998). This index matrix is then chosen to select the appropriate number of sentences to be included in the summary

2.2.5 Vector-Based Semantic Analysis using Random Indexing

Vector-based semantic analysis is a technology for extracting semantically similar terms from textual data by observing the distribution and collocation of terms in text (Karlgren and Sahlgren 2001). The result of running a vector-based semantic analysis on a text collection is in effect a thesaurus: an associative model of term meaning.

Random Indexing uses sparse, high-dimensional random index vectors to represent documents (or context regions or textual units of any size). Given that each document has been assigned a random index vector, term similarities can be calculated by computing a terms-by-contexts co-occurrence matrix. Each row in the matrix represents a term, and the term vectors are of the same dimensionality as are the random vectors assigned to documents. Each time a term is found in a document, that document's random index vector is added to the row for the term in question. In this way, terms are represented in the matrix by high-dimensional semantic context vectors which contain traces of each context the term has been observed in. The underlying assumption is that semantically similar terms will occur in similar contexts, and that their context vectors therefore will be similar to some extent. Thus, it should be possible to calculate the semantic similarity between any given terms by calculating the similarity between their context vectors (mathematically, this is done by calculating the cosine of the angles between the context vectors). This similarity measure will thus reflect the distributional (or contextual) similarity between terms.

2.2.6 Pronoun Resolution

Automatically summarised text can sometimes result in broken anaphoric references due to the fact that the sentences are extracted without making any deeper linguistic analysis of the text. For example, if the summarization algorithm decides to select only the second sentence in the following discourse fragment, and *John* and *Lisa* are not mentioned earlier in the text, it will be impossible to know that '*He'* refers to *John* and '*her'* refers to *Lisa*.

John kissed **Lisa**. **He** has been in love with **her**.

Various methods have been implemented in order to resolve different types of pronouns. The Pronominal Resolution Module (PRM) is a method which resolves some types of pronouns in Swedish. The method is implemented as a text pre-processor, written in Perl (Hassel 2000).

PRM works as a preprocessor to SweSum and uses lists of *focus applicants*. Focus means person(s)/item(s) that are most prominent at a specific point in the discourse. These lists can be seen as stacks of focus applicants and applicants are pushed upon appropriate stack when found. In other words, every time a nominal phrase is recognized, it is categorized and pushed onto the appropriate list for that category.

Currently there are only two *focus applicant* lists in PRM; one for each natural gander. Choice of applicant for an anaphor is based upon salience (represented by the antecedent's position in a list) and semantic likelihood (based on what list the antecedent is to be found in).The latter is determined by using semantic information in a noun lexicon.

PRM uses a lexicon of nouns that contains information about each entry's natural (han/hon -he/she) or grammatical (den/det - it/it) gender. The current noun lexicon contains over 1500 gender specified first names.
The algorithm used in PRM consists of three distinct phases that act on three different levels: *discourse*, *sentence* and *word* level.
For each *discourse* (text):

- Identify and annotate abbreviations. This step is necessary for sentence segmentation.
- Identify and segment sentences.

For each *sentence*:

- Use templates to identify special cases such as active/passive phrases.
- If one or more pronouns/anaphoric expressions are found, annotate them using appropriate focus applicant lists.
- Identify and segment words.

For each *word* in each sentence:

- Search for pronoun/anaphoric expression (i.e. han (he), hon (she), den (it or the), det (it or the), företaget (the company), etc.). If a pronoun/anaphor is found, annotate it in AHTML with the most likely antecedent and sentence number found in the corresponding focus applicant list (based on gender and/or any other supplied semantic information). Pronouns are marked with the tag pair <! ANAPHOR REF="Referent" LINE="Line number"> and </! ANAPHOR>, For example, the pronoun *he* in the annotation <! ANAPHOR REF="Robert" LINE="16">he</! ANAPHOR> represents the antecedent "Robert" found in the sixteenth sentence.
- Search and compare with lexicon to see if it is a known noun.
- If a noun is found, place it first in appropriate focus applicant list (according to category) along with information on in which sentence it was found.

2.2.7 Machine Learning Techniques

Given a set of training documents and their extractive summaries, the summarization process is modelled as a classification problem: sentences are classified as summary sentences and non-summary sentences based on the features that they possess (Neto et al. 2002). The classification probabilities are learnt statistically from the training data, using Bayes' rule:

$$P\ (s \in\ < S \mid F1, F2... FN) = P\ (F1, F2\ldots FN \mid s \in S) * P\ (s \in S) / P\ (F1, F2\ldots FN)$$

where, s is sentences from the document collection, F1, F2…FN, are features used in classification and $P\ (s \in\ < S \mid F1, F2... FN)$ is the probability that sentence s will be chosen to form the summary S given that it possesses features F1, F2…FN.
Kupiek et al. (1995) have developed a trainable summarization program that is grounded in a sound statistical framework. For summaries that were 25% of the size of the average test document, it selected 84% of the sentences chosen by professionals.

2.2.8 SUMMARIST

The SUMMARIST project is developed by Information Sciences Institute (ISI)[6] at the University of Southern California (Lin and Hovy 1997). Extract as well abstract summaries can be generated by this system. The goal of SUMMARIST is to provide a robust text summarization based on the 'equation':
Summarization = topic identification + interpretation + generation.
Each step contains modules trained on large corpora of text. The identification stage filters the input document to determine the most important topics.
SUMMARIST combines NLP methods using statistical techniques (*extract*) with symbolic world knowledge (*abstract*) provided by dictionaries, WordNet and similar resources

[6] http://www.isi.edu/

3. SweSum

SweSum[7] (Dalianis 2000) is a web-based text summarizer developed at Royal Institute of Technology (KTH). It uses text extraction based on statistical and linguistic as well as heuristic methods to obtain text summarization and its domain is Swedish HTML-tagged newspaper text.

SweSum is available for Swedish, Norwegian, Danish, Spanish, English, French and German. It has been evaluated and its performance for English, Danish, Norwegian and Swedish is considered to be state-of-the-art. The French, German and Spanish versions are in prototype states (Dalianis 2000).

Some of the techniques described in the previous chapter are not employed by SweSum, since they require linguistic resources that are not available in the current implementation. For example the algorithms used in *Knowledge-Based concept counting* and *Lexical chains* methods require access to WordNet (not currently used in SweSum). Since SweSum generates only *extract* summaries, the methods in the SUMMARIST project for generating *abstract* summaries, are not implemented.

There are plans to implement *Latent Semantic Analysis* (LSA) or *Random Indexing* methods in the future releases.

3.1 Architecture

SweSum is implemented as a HTTP client/server application as shown in Figure 2. The summarization program is located on the server side and the client is a browser such as Internet Explorer or Netscape Navigator. The client interacts with the server using the HTTP protocol which helps in the accurate transfer of data from a browser and responses from the server.

The summarization process starts when the user (client) clicks on a hyperlink (*summarize*) in the SweSum Web site:

- The browser (Web client) sends a summarization request to the Web server where SweSum is located (marked 1 in Figure 2). The document/ (URL of the document) to be summarized is attached to the request.
- The request is forwarded to SweSum through a HTTP server (2).
- The document is summarized (3-6).
- The summary is returned back to the HTTP server (7) that returns the summarized document to the client (8).
- The browser then renders the summarized text to the screen.

3.2 HTTP

HTTP (**H**yper**T**ext **T**ransfer **P**rotocol) is the network protocol used to deliver resources on the World Wide Web. A resource is a file (HTML, image), a dynamically-generated query result, an output of a CGI script, or something else.

[7] A live version at http://swesum.nada.kth.se/index.html

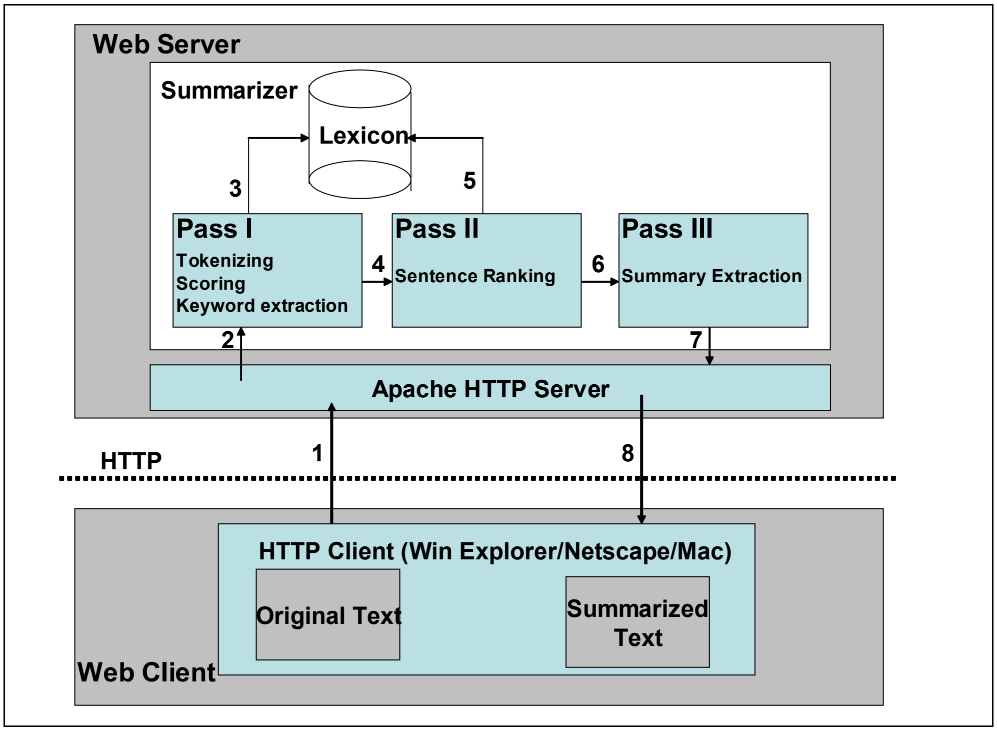


Figure 2: SweSum architecture

3.3 Web Client

Web client (browser) is an application for interpreting and displaying HTML documents, i.e. Netscape Navigator, Internet Explorer, Mozilla etc.

3.3.1 HTML

HTML (**H**yper**T**ext **M**arkup **L**anguage) is a markup language which is made up of various tags embedded in the text of a document. The tags are used to format the document. There are three different types of tags;

- Containers, in which there is a start and an end tag (<B> </B>).
- Optional, end tags are optional (not required) for example <p> </p>.
- Empty, contains no end tags (<IMG>[8], <BR>[9]).

HTML is platform-independent which means that the HTML documents are portable to different computer systems. [See Appendix A]

3.3.2 Original text

The original text is in text/HTML format on the Web or located on the PC.
HTML file: A HTML file located on WWW.
Text area: A text the end-user writes directly in the entry page to SweSum's web site.
Upload file: A text/HTML file located on the end-user's PC.

[8] Image
[9] Break or new line

3.3.3 JavaScript

JavaScript is an interpreted easy-to-use programming language that can be embedded in the header of an HTML document. It can increase the dynamic and interactive features of a web page by allowing the user to add special effects, check forms, and perform calculation and more.

3.4 Summarizer

The summarizer is located on the web server. It takes the original text as input and performs summarization in three passes and creates the final output (the summarized text).

3.4.1 Lexicon

SweSum uses a static lexicon containing many frequent open class words of the current language, Swedish[10], English etc. The lexicon is a data structure[11] for storing key/value pairs called ***root table*** where the key is the inflected word and the value is the stem/root of the word. For example *boy* and *boy**s*** have different inflections but the same root.

3.4.2 First pass

3.4.2.1 Tokenizer

The tokenizer reads the original text which is in ASCII[12] format and outputs the tokenized text. The tokenizer:

- Removes all new line characters "\n" in the original text.
- Marks all abbreviations[13] in the document with the tag <! ABBRV>. The word sv.[14], for instance will be written <! ABBRV>sv. </! ABBRV>.
- Invokes Pronominal Resolution which is only partially implemented for Swedish and is in a prototype state.
- Finds the sentence boundaries by searching for periods, exclamations, question marks and <BR> (the HTML new line) in the text. A new line character "\n" is inserted after each sentence boundary.
- Finds the word boundaries by searching for the characters such as "**.**", "**,**", "**!**", "**?**", "**<**", "**>**", "**:**", spaces, tabs and new lines.

The output of the tokenizer is the original text which has now the additional new line markers after sentence boundaries. The new line character "\n" is used to divide the text into different lines. Each line is put into a hash table called ***text table***. The key to the table is the line number and the value is the content of the line. This is shown in the following table:

[10] The Swedish version of SweSum uses a keyword lexicon that contains about 40,000 words and their 700,000 possible inflections.

[11] It is implemented as a hash table in Perl.

[12] American Standard Code for Information Interchange (ASCII) is the numerical representation of a character such as 'a' or '@'. Each character is represented by a byte (7-bit).

[13] If an abbreviation database is available for the current language.

[14] This is the abbreviation for svensk (Swedish).

Sentence	Line Nr
<html>	1
<title> War against Iraq </title>	2
<body>	3
American-led forces will stay in Iraq no longer than necessary.	4
….. ….. …..	….
</body>	n-1
</html>	n

Table 1: Text table

3.4.2.2 Keyword extraction

Keywords are meaning-carrying words but different researchers use different parts of speech as keywords. Keywords used by SweSum are nouns, adjectives and adverbs of time.
There are two different ways to find keywords:

1 Parsing or tagging the text.
2 Using a static lexicon

The second method is less complicated and faster than the first one but the lexicon in the second method needs to be updated as natural languages continually create new words. *Keyword frequency counting* in SweSum is based on the open class words[15] by using a static lexicon. The system asks the lexicon for a specific word (inflected form), the lexicon returns the lemma of the word which is stored in a ***word frequency hash table.*** I.e. the system does not count two identical words with different inflections as two different words.

Word	Frequency
American-led	10
Force	5
Iraq	26
Baghdad	13
…. ….	…
War	20

Table 2: Word frequency table (wft)

3.4.2.3 Scoring

Scoring is used to decide on the importance of each line in the document. The HTML tagged lines which do not contain text are assigned the static value "not text" i.e. those lines will not be summarized. All text lines are assigned the value "text".

[15] Adjectives, nouns and adverbs of time.

Sentence	Line Nr	Value
<html>	**1**	**Not text**
<title> War against Iraq </title>	**2**	**Not text**
<body>	**3**	**Not text**
American-led forces will stay in Iraq no longer than necessary.	**4**	**Text**
….. **…..** **…..**	**….**	
</body>	**n-1**	**Not text**
</html>	**n**	**Not text**

Table 3: Text table value

Text lines are put into a data structure[16] for storing key/value called ***text table value***. The key to the table is the line content and the line number and the value is the score of the line. The line score depends on the position of the line and the scores of the words contained in the line. The idea is that high scoring sentences are kept in the final summary.

The following methods are used by SweSum for calculation of the line scores:
First line in the text should be included in the summarization. This is done by giving it a very high score. (Default value '1000')

Position score depends on the type of the text. SweSum supports two types of text: *Newspaper* and *Report*.
Line position is less important in reports than the newspaper text. In newspaper text the most important part of the text is the first line followed by other lines in descending order. The following formula is used for calculation of the position score for newspaper texts:

Position score = (1/line nr)*10.

Numerical values in a document such as dates are important. A constant value is added to the score of the line. (Default value '1')

Bold text (<B>) in the HTML coded text is important and given a higher score. In some of Swedish newspaper texts it indicates beginning of a paragraph and the first sentence of each paragraph usually introduces the paragraph. (Default value '100'.)

Keywords are the most frequent words in the text. Sentences containing keywords are scored higher than the ones with fewer keywords.

User keywords are entered to the system by the user such as via a query.

[16] It is implemented as a hash table in Perl.

Simple combination function
All the above parameters are put in a combination function with modifiable weights (default values) to obtain the total score of each sentence.

3.4.3 Second pass

In the second pass, the score of each word in the sentence is calculated and added to the sentence score in the *text table value*.

3.4.3.1 Sentence score

Word score = (word frequency) * (a keyword constant[17])
Sentence Score = ∑ *word score* (for all words in the current sentence)
The sentence weight is equal to the weighted sum of the weights of all the words in the sentence.

Example: ***American-led forces*** *will stay in* ***Iraq*** *no longer than necessary* (see Table 2).
Word score (American-led) = 10 * 0.333 => 3.33
Word score (force[18]) = 5 * 0.333 => 1.665
Word score (Iraq) = 26 * 0.333 => 8.658
Sentence Score = 3.33 + 1.665 + 8.658 =>**13.653**

3.4.3.2 Average sentence length

To avoid inaccurately awarding long sentences with a high ranking, the sentence score is multiplied by the *average sentence length* (ASL) and divided by the number of words in the current sentence to normalize for length.

Word-count = nr of words in the text.
Line-count = nr `of` lines in the text.
Average sentence length (ASL) = *Word-count / Line-count*
Sentence score = (ASL * *Sentence Score*)/ (nr of words in the current sentence)

The words and lines are counted dynamically during the summarization process. For example, if a text contains 20 sentences and 200 words, the *Sentence Score* for line 5 is calculated as shown in the example below:

Word-count = 40 (the nr of words in line 1-5)
Line-count = 5
Nr of words in the current sentence = **10**
Sentence score = **13.653**
ASL = *Word-count / Line-count*= 40/5=> **8**
Sentence Score = (8*13.653)/10 = **10.9224**

[17] Default value = 0,333
[18] The system counts the word *force*, the stem of the *forced* (inflected form).

3.4.3.3 Cutoff size and unit

The user entered values *cutoff size* and *unit* decides the amount and type of the summary. The *unit* has one of the values *percent*, *words* or *characters* and the *cutoff size* a numerical value. Examples:

Cutoff size = 30, unit = *percent* → keeps 30% of words in the original text.
Cutoff size = 300, unit = *words* → the summary contains 300 words.
Cutoff size = 1000, unit = *characters* → the summary contains 1000 characters.

3.4.3.4 Sorted text value

The sentences in the *text table value* are sorted according to the highest value and stored in an array called *sorted text value*. By using the information from the *cutoff size* and the *unit*, the number of lines to be kept in the final summarization is calculated. Those lines are the high-ranking sentences in the *sorted text value*.

3.4.4 Third pass

In the third pass, the final summary file (HTML format) is created. This file includes the following[19]:

- All *non text* html lines
- All the highest ranking text lines in the *sorted text value*.
- Statistical information about the summary, number of words, number of lines, the most frequent keywords etc.

3.5 Evaluation

Evaluating summaries generated by automatic text summarization systems is not a straightforward process.

The evaluation task is normally performed ***manually*** by individuals who subjectively compare different summaries and choose the best one.

A problem with this approach is that the individuals who perform the evaluation task normally have very different ideas on what a good summary should contain. In a test, Hassel (2003) found that at best there was a 70% agreement between summaries created by two individuals.

A further problem with manually performed evaluation is that it is an extremely time-consuming task.

Automatic evaluation is another approach in which the evaluation process is automatized. This is however still a research topic.

Using *gold standards* by sentence selection is one way to implement an automatic evaluation system. A gold standard summary contains the most frequently chosen sentences in a given text, produced by majority votes over a number of manually created extracts (Dalianis et al., 2003).

The gold standard can then be compared with the different summaries created by the summarization system.

A gold standard is supposed to be the correct, true or best result. This presupposes, however, that there is a single best result. In summarizations there appears to be no "one

[19] The variable n (line nr) preserves the line order in the original text.

truth", as is evidenced by a low agreement between humans in producing gold standard summaries by sentence selection.

Manually evaluation of SweSum

First evaluation of SweSum was carried out in the year 2000 in a field test within the framework of a 4-credit course at NADA/KTH (Dalianis 2000). Nine students were given the task of automatically (using SweSum) summarizing 10 texts, by gradually lowering the size of the summary (in percent) and noting in a questionnaire when coherence was broken and when important information was missing. The purpose was to see how much a text can be summarized without loosing coherence or important information.
The results showed at 30 percent summarization for good coherence and 24 percent summarization for good content. (30 percent summarization means to remove 70 percent of the text).
Fallahi (2003) presented a thorough manual evaluation of SweSum carried out at the Swedish newspaper Sydsvenska Dagbladet[20]. He compared the performance of SweSum as opposed to human editors in summarizing 334 Swedish news texts.
He found that in general, SweSum performed well, even if a number of shortcomings showed up.

- Sometimes SweSum cut sentences by a mistake in sentence boundary detection.
- At the end of a long article, sometimes the first sentence of a paragraph was omitted while the second or third sentence was kept, so that the quality of the summarized text was affected.
- Sentences of an unformatted text were put together in a single paragraph. The latter problem is however fixed in the current version of SweSum.
- For cutting down news to SMS size (maximum 160 characters), SweSum performed remarkably well.

Evaluation of SweSum using an extract corpus

In order to evaluate SweSum, Hassel (2003) has created an extract corpus for Swedish called *KTH eXtract Corpus*[21]. The corpus contains a number of original texts and asset of their different manually extracts. The extracts are created by different informants who are asked to choose a predefined number of important sentences in a given text.
The KTH extract tool gathers statistics on how many times a specific sentence from a text has been included in a number of different summaries. It generates then the gold standard summary which is produced by majority votes containing the most frequently chosen sentences.
The extract summaries with SweSum were manually compared on sentence level with the gold standard extracts.
The result of a field test by using the KTH extract tool showed that the summaries generated with SweSum shared as many as 57.2% of the sentences with the gold standard.

[20] http://w1.sydsvenskan.se/index_css.html

[21] A live version can be found at http://www.nada.kth.se/iplab/hlt/kthxc/showsumstats.php

3.6 Notes on SweSum

3.6.1 Summarization algorithms

Cohesion:
SweSum employs a linear summarization technique:
1 The whole text is divided into sentences.
2 Each sentence is scored separately.
3 The sentences with highest score are extracted for the final summary.
But the extracted sentences may or may not relate to each other.

Important information in the text:
The most important (central) topics in the text are identified by using a statistical keyword approach (word frequency). But this method cannot detect all important information such as synonyms in the text. This can be somewhat helped by expanding keywords with light semantics, for example LSA (*Latent Semantic Analysis*) or by *Random Indexing.*

Redundancy:
High keyword ranking can introduce redundancies in the summary. The summary become concentrated around one specific topic. Methods such as LSA can be used in order to reduce the amount of redundancy.

3.6.2 Program Structure

The SweSum project started 1999 by implementing a text summarizer for Swedish and since then it has been developed and new versions of SweSum have been expanded with English, French, German, etc. SweSum uses a plain structure i.e. all languages are handled in the same module and that make the program code unreadable and difficult to modify. Especially for languages such as Persian using Unicode characters, it is necessary to handle these languages in different modules.

3.6.3 HTML parser

The current implementation of HTML parsing in SweSum handles only very simple html pages. Advanced pages including frames, images, etc are not supported. There are a lot of problems with parsing. Sometimes some important information such as *charset*[22] in the header of the summarized text is missing and consequently the presented text by the browser is unreadable ASCII characters.

3.6.4 Programming language

Perl is a very powerful and flexible script language for text management (tokenizing). It is easy to put together short Perl programs to perform tasks that might otherwise require hours of developments effort with any conventional programming language. But programmers who are used to Java, C or C++ can find its syntax (especially in regular expressions) and data types unusual.

[22] This parameter contains information about the used encoding in the document.

4. FarsiSum

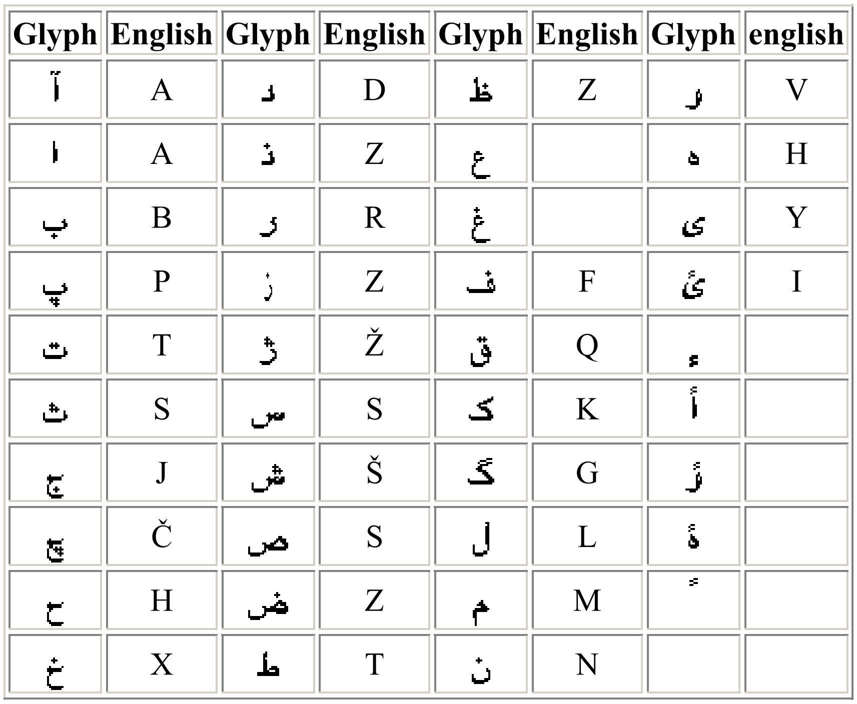
FarsiSum is a web-based text summarizer for Persian based upon SweSum. It summarizes Persian newspaper text in HTML/text encoded in Unicode format.

4.1 Persian Language

Persian (also known as **Farsi** or **Parsi**[23]) is a language spoken in Iran, Tajikistan and Afghanistan. Persian is a member of the Indo-European family of languages, and within that family, it belongs to the Indo-Iranian (Aryan) branch, within which, the Iranian sub-branch consists of the following chronological linguistic path:
Avestan/Old Persian -> Middle Persian (Pahlavi) -> Modern Persian.

4.1.1 Writing System

Old Persian was based on the cuneiform writing system (pictogram style) as early as the 6th century B.C. Later, the Persians invented a new alphabet called Pahlavi to replace the cuneiform alphabet. However, after the Arabic conquest in 651, the Persians adopted a unified Arabic script for writing.



Glyph	English	Glyph	English	Glyph	English	Glyph	english
آ	A	د	D	ظ	Z	و	V
ا	A	ذ	Z	ع		ه	H
ب	B	ر	R	غ		ی	Y
پ	P	ز	Z	ف	F	ئ	I
ت	T	ژ	Ž	ق	Q	ء	
ث	S	س	S	ک	K	أ	
ج	J	ش	Š	گ	G	ؤ	
چ	Č	ص	S	ل	L	ۀ	
ح	H	ض	Z	م	M	ٔ	
خ	X	ط	T	ن	N		

Table 4: Persian alphabet

The modern Persian writing system (also known as Perso-Arabic) uses the Arabic alphabet, but with the addition of four letters which do not occur in Arabic. These are: ‘پ چ گ ژ‘, **ž, g č, p**. Not all of the sounds in the Arabic alphabet exist in the Persian language; as a result, more that one letter may represent more than one sound. For example, there are four letters in Persian for the sound **z** (**ز ض ظ ذ**) and three for the

[23] The term "Farsi", while being an accepted term for the language in Iran, came about due to the fact that Arabic does not contain the letter "p", so Arabs pronounced "Parsi" as "Farsi".

sound **s** (س ص ث). Also, a single sound in Persian may have many symbols that correspond to it, which may also add to the confusion. A complete list of Persian alphabet is presented in Table 4.

The main rules for writing Persian:

- Persian characters take one of four forms: initial, medial, final, and stand-alone, depending on where they occur in the text stream.

Initial	Medial	Final	Isolated	
گـ	ـگـ	ـگ	گ	G
بـ	ـبـ	ـب	ب	B

Table 5: Persian character forms

- The last character in the word marks the end of the word and is written in the final or isolated form.
- Letters in a word are connected to each other except for the letters [āzrdv]‘ د ر ذ ز ا و آ **’.** These letters, because of their form, cannot be written so that they connect to the following character in a word.
- The alphabet is read from right to left while numbers are read from left to right.
- Short vowels *a*, *i* and *u* are not represented in a word except when they are in the initial or final position in a word.
- The long vowel [a] is represented by آ in the initial position and by ا otherwise, for example:

آب	**āb**	water
باد	**bād**	wind

Table 6: the long vowel [a]

- Long vowels [i:], [u:] are represented by the consonants y and v.

4.1.2 Numbers

Persian numbers have the same origin as the Latin numbers and are written left to right.

		Unicode			Unicode
0	۰	06F0	5	٥	06F5
1	۱	06F1	6	٦	06F6
2	۲	06F2	7	۷	06F7
3	۳	06F3	8	۸	06F8
4	٤	06F4	9	۹	06F9

Table 7: Persian Numbers

4.1.3 Word Boundaries

In Persian text, word boundaries can be delimited by space, punctuation, and the forms of the characters indicating its position within a word. Some morphemes may appear in either attached or detached form.

Space

Word boundaries are usually denoted by space. However, compound words and light verb constructions may appear without a space separating them.

Punctuation

The stop (.) marks a sentence boundary, but it may also appear in the formation of abbreviations or acronyms. The slash (/) is used in the numbers and the dash (-) could be used to separate compound words. Other punctuation marks including the comma, quotes, brackets, question mark and colon unambiguously indicate word boundaries.

Character form

As explained earlier, Persian characters take one of four forms: initial, medial, final, and stand-alone and a final form character indicates the end of a word. This can be used for instance by a tokenizer to determine word boundaries.

4.1.4 Acronyms and Abbreviations

Acronyms

Persian acronyms are formed from the initial letter of each of the successive parts of a compound term.

ساواك	سازمان امنیت و اطلاعات کشور
SAVAK[24]	Sazman-e **A**mniyat **V**a **A**ttelaat-e[25] **K**eshvar

Table 8: Persian acronyms

Foreign acronyms in Persian normally consist of one or more characters followed by a stop. The roman characters in acronyms are transliterated into Persian. There exist, however, variations to this format. Certain magazines and newspapers form the acronyms without a stop as illustrated in the example below:

بی بی سی	بی.بی.سی	BBC
bibici	**bi.bi.ci**	
اف بی آی	اف.بی.آی	FBI
efbiay	**ef.bi.ay**	

Table 9: foreign acronyms

Certain words that are considered acronyms in English are treated as proper nouns in Persian and are not represented by a letter by letter transliteration, instead they are written as they are pronounced as shown in the following example.

[24] Iran's National Intelligence and Security Organization during the Shah’s rule.

[25] It is pronounced **e**ttelaat.

سیا	sīyā	CIA
ایدز	aydz	AIDS

Table 10: foreign acronyms

Abbreviations

Abbreviation can appear as a single character with a stop. However, the format used for abbreviations is not very consistent. Two characters abbreviating a lexical element can be written with stops following both, or a stop following only the last character, or simply separated by a space without any stops. This is illustrated in the following example:

ه.ق.	ه ق.	ه ق	
h.Q.	h Q.	h Q	[26]

Table 11: Persian abbreviations

4.1.5 Personal Pronouns

Pronouns include personal as well as quantifying pronouns such as everyone and someone. Personal pronouns can appear either as separate lexical elements or as morphemes on the noun, verb or adjective. Example:

Ketāb-aš	**ketāb-e ū**
Book-his/her	**Book-ezafe[27] he/she**
His/her book	**His/her book**

Table 12:Personal pronouns

asb-e safīd-am	**asb-e safīd-e man**
Horse-ezafe white-my	**Horse-ezafe white-ezafe I**
My white horse	**My white horse**

Table 13: Personal pronouns

Dīd-am-aš	**Man ūrā dīd-am**
Saw-1sg.pron-her/him	**I him/her saw-1sg-pronoun**
I saw her/him	**I saw her/him**

Table 14: Personal pronouns

[26] **H**ejri **Q**amari (The Islamic Lunar Calendar).

[27] The *ezafe* suffix is the element joining the Persian noun phrase constituents to each other.

4.2 *Encodings*

4.2.1 The ISO[28] 8859

ISO 8859 is a full series of standardized multilingual **single-byte** coded (8bit) graphic character sets for writing in alphabetic languages:
Latin1 (West European), Latin2 (East European), Latin3 (South European), Latin4 (North European), Cyrillic, Arabic, Greek, Hebrew, Latin5 (Turkish), Latin6 (Nordic).
The ISO 8859 *charsets* are not even remotely as complete as the truly great Unicode but they have been around and usable for quite a while (first registered Internet *charsets* for use with MIME[29]) and have already offered a major improvement over the plain 7bit US-ASCII.
Unicode (ISO 10646) will make this whole chaos of mutually incompatible *charsets* superfluous because it unifies a superset of all established *charsets* and is out to cover all the world's languages.

4.2.2 Unicode & UTF-8

Unicode is a **two-byte** (16bit) encoding which covers all of the world's common writing systems. Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. The encoding is the system by which the characters in a set are represented in binary form in a file. The Unicode set may be represented using three encodings: UTF-8, UTF-16 and UTF-32.
UTF-8 *Unicode Transformation Format* is an algorithmic mapping from every Unicode scalar value to a unique byte sequence.

Character	Unicode	ASCII	UTF-8
B ب	0628	╪¿	&#1576;
D د	062F	╪»	&#1583;

Table 15: Unicode example

4.3 *Implementation of FarsiSum*

The design and implementation includes Web Programming on a Windows-based System with Perl as programming language. Since the current implementation is in a prototype state, not all SweSum functionality has been implemented for this version of FarsiSum:

- A stop-list is used instead of a Persian dictionary.
- Abbreviation and acronym modules are not used

FarsiSum uses the same structure used by SweSum (see Figure 3) but it modifies some modules to be able to handle documents with Unicode content and UTF-8 encoding.

[28] **I**nternational **O**rganization for **S**tandardization (ISO)

[29] *Multipurpose Internet Mail Extensions* (MIME) extends the format of Internet mail to allow non-US-ASCII textual messages, non-textual messages, multipart message bodies, and non-US-ASCII information in message headers.

The summarization process starts when the user (client) clicks on a hyperlink (*summarize*) in the FarsiSum Web site:

- The browser (Web client) sends a summarization request including a document to be summarized, through a HTTP server to the Web server where FarsiSum is located.
- The document is summarized in three phases using a Persian stop-list.
- The summary is returned back to the client through the HTTP.
- The browser then renders the summarized text to the screen.

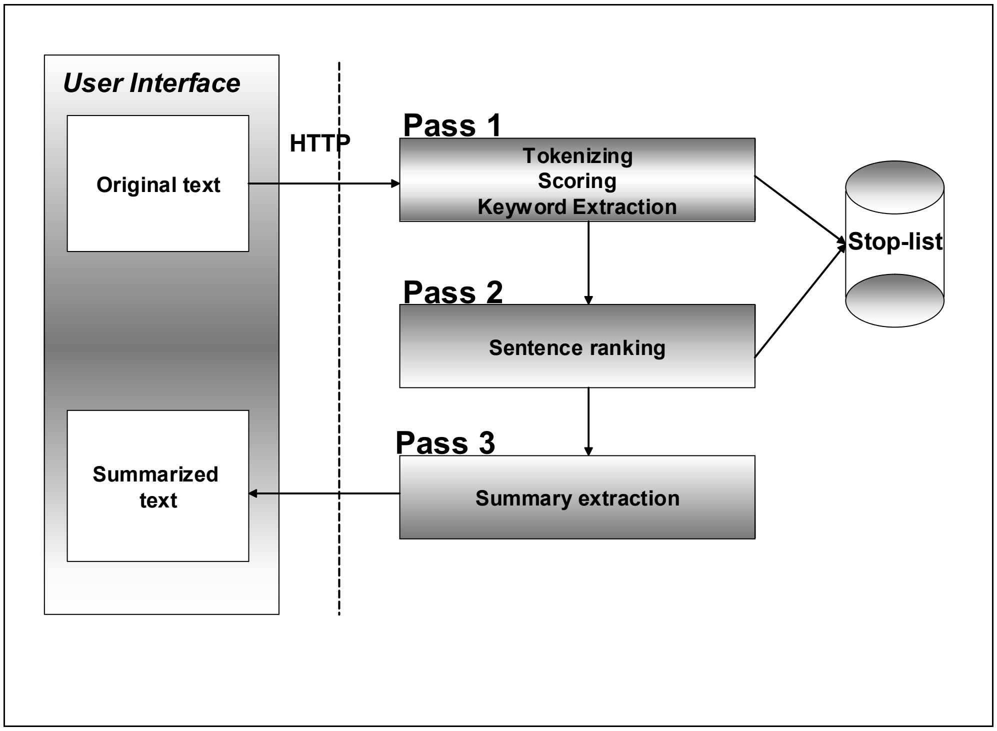


Figure 3: FarsiSum architecture

4.3.1 User Interface

The user interface includes:

- The first page of FarsiSum on WWW presented in Persian.
- A Persian online editor for writing in Persian.
- The final summary including statistical information to the user, presented in Persian.

4.3.2 The Stop-List

The stop-list is a HTML file (UTF-8 encoding) containing about 200 high-frequency Persian words including the most common verbs, pronouns, adverbs, conjunctions, prepositions and articles. The assumption is that words not included in the stop-list are nouns or adjectives. The stop-list has been successively built during the implementation phase by running FarsiSum in order to find the most common words in Persian.

Table 16 below shows 18 most common Persian words (newspaper text) identified by FarsiSum.

Rank	Persian	English	Rank	Persian	English	Rank	Persian	English
1	و	and	2	در	in	3	به	to
4	از	from	5	که	ke[30]	6	را	ra[31]
7	این	this	8	با	with	9	بودن	be
10	هر	every	11	برای	for	12	بر	on
13	داشتن	have	14	کردن	do	15	آنها	they
16	یا	or	17	گفتن	say	18	شدن	become

Table 16: the most common Persian words

4.3.3 Pass I & II

Tokenizer:
The tokenizer is modified in order to recognize Persian comma, semi colon and question mark.

- Sentence boundaries are found by searching for periods, exclamations, question marks and <BR> (the HTML new line) and the Persian question mark (؟)[32].
- The tokenizer finds the word boundaries by searching for characters such as “.”, “,”, “!”, “?”, “<”, “>”, “:”, spaces, tabs and new lines. Persian semi colon, comma and question mark can also be recognized.
- All words in the document are converted from ASCII to utf-8. These words are then compared with the words in the stop-list which are in the utf-8 format.
- The word order in Persian is SOV[33] , i.e. the last word in a sentence is a verb. This knowledge is used to prevent verbs from being stored in the *Word frequency table*.

Sentence Scoring:
There are no changes in the scoring algorithms used by SweSum. The sentences are scored and put into a ranking list. Each sentence is scored according to the position of the sentence and the scores of the words contained in it. The highest-ranking sentences are then identified and kept in the final summary.
A sample sentence-list produced in the second pass is shown below:

Nr 1 Rank 12.7054545454545, من از جنگ دوم جهانی گفتم و از ژاپن و آلمان مثال آوردم که
\.در حقیقت پس از رهایی از چنگال فاشیسم به دموکراسی رسیدند

Nr 2 Rank 7.52, متاسفانه جنگ ها همزاد انسان بوده و از نخستین دوران بشر اولیه تا به امروز
ادامه داشته است.

[30] Conjunction

[31] Object marker.

[32] The Persian question mark is represented by “&#1567;” in UTF-8 encoding.

[33] **SOV** stands for **S**ubject, **O**bject and **V**erb.

4.3.4 Pass III

Post-processing module is mainly used for creating the final summary file (HTML format) presented in Persian. The file includes all non-text html lines, highest ranking text lines and statistical information about the summary, number of words, number of lines, the most frequent words, etc.

4.4 Notes on the Current Implementation

The current implementation of FarsiSum is still a prototype. It uses a very simple stop-list in order to identify the important keywords in the text. Persian acronyms and abbreviations are not detected by the current tokenizer.

In addition, Persian syntax is quite ambiguous in written form, which raises certain difficulties in automatic parsing of written text and automatic text summarization.

For example, selection of important keywords in the *topic identification* process will be affected by the following word boundary ambiguities:

- Compound words may appear as two different words.
- Bound morphemes may appear as free morphemes or vice versa.

Some of the factors which contribute to the ambiguity are listed and explained below. These ambiguities are not resolved in the current implementation.

4.4.1 Word Boundary Ambiguity

The stop (.) marks a sentence boundary, but it may also appear in the formation of abbreviations or acronyms.

Compound words and light verb constructions may also appear with or without a space separating them.

4.4.2 Ambiguity in morphology

Certain ambiguities arise in a computational analysis of Persian text since the same surface form can represent different morphemes (Megerdoomian and Rémi 2000).

In addition, short vowels are not marked in written text, which results in different possibilities for analysis. The word **کرم** (***krm)***, for instance, can be pronounced with different vowel combinations resulting in five possible lexical elements as shown in the example below.

کرم ***krm***				
kerm	***karam***	***kerem***	***krom***	***karm***
worm	generosity	cream	chrome	vine

Table 17:ambiguity in morphology

A reader uses the context to determine the word in the sentence.

Furthermore, certain affixes always appear bound whereas others can also appear as free morphemes. For example the affix *mī* in “man *mī ravam*“(I go) can be written in three different ways as shown in the example below:

- As free morpheme *mĪ* with space between *mĪ* and *ravam*.[34]

[34] The capital letter ‘*Ī’* in “*mĪ”* indicates that the ‘*Ī’* is in final form i.e. the word boundary.

- As free morpheme *mĪ* without space in “*mĪ ravam*”.
- As bound morpheme “*mīravam*”.

Free morpheme with space	Free morpheme without space	bound
می روم	**می‌روم**	**میروم**
mĪ rvm	**mĪrvm**	**mīrvm**
mī ravam (I go)		

Table 18:free/bound morphemes

There exist other lexical elements, such as the preposition ***be***, the postposition ***rā***, or the relativizer ***ke***, that usually appear as separate words in written text, but which can also be found as attached morphemes.

4.4.3 Word Order

The canonical word order in Persian is SOV, but Persian is a free word order language and the sentential constituents can be moved around in the clause. This is especially the case for preposition phrases and adverbials. Adverbs may appear almost anywhere in the clause, in between the various constituents. Despite the relatively free word order, the sentences in the written text often remain verb-final.

4.4.4 Phrase Boundaries

There are no overt markers, such as case morphology, to indicate the function of a noun phrase or its boundary; only specific direct objects receive an overt marker “rā”. Since Persian is a verb final language, the resulting structure is then *Subject Object-(Object marker) Verb* or *Subject Predicate Copula*, but there are no obvious markers to determine where the Subject ends and the Object or the Predicate begins.

4.4.5 Possessive Construction

In English, the link between the two nouns is marked by”`s" (e.g., John's car) or the preposition "of" (her brother's car). The element joining the Persian noun phrase constituents to each other is the *ezafe* suffix. The *ezafe*, however, is usually pronounced as the short vowel /e/ and is therefore not marked in written text. The result, in Persian written text, is a series of consecutive nouns without any overt links or boundaries as shown in the following example:

Māshīn	**dūst**	**brādr**	**Ali**
Car	**friend**	**brother**	**Ali**
Ali's brother's friend's car			

Table 19: the ezafa suffix

The actual pronunciation for this example in spoken language is “Māshīn-*e* dūst-*e* barādar-*e* Ali”, where the *ezafe* morpheme is represented by the –*e*.

4.4.6 Light Verb Construction

Persian light verb constructions consist of a preverbal element, which could be a noun, adjective or preposition, followed by a light verb, such as *kardan* (do, make), *dādan* (give), *zadan* (hit, strike) which has partly or completely lost its original meaning.
The elements of a light verb construction can appear either as two separate words or as a compound word, but the meaning of these light verb constructions cannot be obtained by translating each element separately as examples in Table 20 illustrate.
The number of verbs that can be used as light verbs is limited, but these constructions are extremely productive in Persian (Megerdoomian and Rémi 2000).

fekrkardan	**فكركردن**	**"thought do"**	**to think**
gūš dādan	**گوش دادن**	**"ear give"**	**to listen**
Jārūzadan	**جاروزدن**	**"broom hit"**	**to sweep**
īmel zadan	**ايميل زدن**	**"email hit"**	**to (send) email**
kelīk kardan	**كليك كردن**	**"click do"**	**to click (on a mouse)**
be donyā āmadan	**به دنياآمدن**	**“to world come”**	**to be born**
az donyā raftan	**آزدنيارفتن**	**“from world go”**	**to die**

Table 20: light verbs

5 Evaluation

In order to evaluate the impact of the stop-list on the final summary, FarsiSum was used in a field test (in two different modes). Seven native speakers were given the task to subjectively compare three different summaries of a Persian text generated by three different methods. This procedure was repeated for three different Persian text documents.
The methods used in the summaries were: enable/disable the stop-list and the *generic mode* implemented in SweSum.
Stop-list enabled: The application has access to the stop-list, i.e. only adjectives and nouns are regarded as keywords.
Stop-list disabled: No access to the stop-list.
The generic mode in SweSum: In this mode there is no Unicode capability i.e. the Persian comma, semi colon and question mark are not recognized by the application as sentence/word boundaries. There is no access to the stop-list.

The participants carried out the test by reading a Persian newspaper text and three different summaries (30%) of the same text generated by FarsiSum (with/without the stop-list) and SweSum in the *generic mode*. In a questionnaire they had to answer to the following questions:

1. Which summary was the best one?
2. Given a scale of 1-5 (1 for the lowest), what score would you assign to each summary?
3. Which summary was the most coherent one?
4. Which summary preserved the most important information?

Results

M1 = Method one in FarsiSum (the stop-list is enabled).
M2 = Method two in FarsiSum (the stop-list is disabled).
M3 = Method three in SweSum (the generic mode).
T1…T3 stands for text 1…3.
The answers to the above questions by the participants are shown in Table A-D (see Appendix C) and Table 21-24 below (given in percent).

1- The best method

As shown in Table 21 and Table 22 (see also table A, B in Appendix C), the test with FarsiSum in the stop-list enable mode gives the best result (average 52.3%) in all texts (T1, T2, and T3). The average for M1 and M2 is 52.3 + 14.3 = 66.6% (methods implemented by FarsiSum).

Text	**M1**	**M2**	**M3**
T1	57,1%	14,3%	28,6%
T2	57,1%	0%	42,9%
T3	42,8%	28,6%	28,6%
Average	***52,3%***	***14,3%***	***33,4%***

Table 21: The best method

Text	**M1**	**M2**	**M3**
T1	39,1%	33,3%	27,6%
T2	37,5%	27,8%	34,7%
T3	35,8%	29,9%	34,3%
Average	***37,5%***	***30.3%***	***32,2%***

Table 22: The best method (score of 1-5)

2- Cohesion

As can be seen from Table 23 and Table C (see Appendix C) the method one gives us the most coherent summary (40.1%).

Text	**M1**	**M2**	**M3**
T1	50,0%	30,0%	20,0%
T2	36,4%	27,2%	36,4%
T3	36,4%	36,4%	27,2%
Average	***40,1%***	***31,2%***	***27,9%***

Table 23: Cohesion

3- Important information preserved

Table 24 (see also Table D, Appendix C) shows that the important information in the text was best preserved in the method one (54.6%).

Text	M1	M2	M3
T1	44,4%	33,3%	22,3%
T2	77,8%	11,1%	11,1%
T3	41,7%	25%	33,3%
Average	***54.6%***	***23,2%***	***22,2%***

Table 24: Important information preserved

The results of the evaluation show that:

- M1 gives the best result, i.e. using the Persian stop-list improved the quality of the final summary.
- Excluding final verbs in sentences from *word frequency table* in the method two (M2) did not improve the final summary, but when the excluding of verbs was combined with the use of the stop-list in Method one (M1), then the final summary was improved in both *cohesion* and preserving the *important information* in the text.

Limitations

The current evaluation uses a very simple evaluation algorithm since there are no NLP resources such as corpora with manual extracts available for Persian.
In addition, the number of participants and texts used in the evaluation process are very low and the texts are chosen from a limited numbers of Persian sites using UTF-8 encoding.
Therefore the current evaluation can only result in some indications on validity and is to a considerable extent based on subjective judgments of the participants.

6 Future Improvements

6.1 SweSum

6.1.1 Methods

The main task in text summarization using extraction methods is to find an accurate balance between the *coherence* and preserving the *important information* in the text. In extreme cases, a summarization method that has the focus on the *important information* may extract the lines A-B1-C2 (as shown in the figure below) as a summary, i.e. the final summary is not coherent. On the other hand if the focus is on the cohesion, the summary may include the lines A-B-C. In this case the important information in lines B1 and C2 are missing.

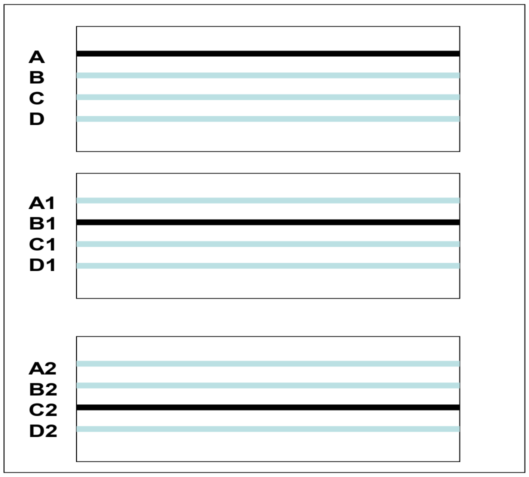


Figure 4: Cohesion & Important Info

To avoid loosing *coherence* in the text we should combine the current linear structure used in SweSum with non-linear methods that operate on block level i.e. collection of sentences rather than sentences. One way to achieve this is to give higher score to lines adjacent to a line with high score.
Another way is a better utilization of the object oriented structure in the HTML (XHTML), for example by using the HTML tags <OL>[35], <P>[36], <UL>[37], etc. These tags operate on a collection of sentences (Appendix A).
One way to improve identification of the *important topics* is to use various tools such as Swedish WordNet-SwordNet[38] for Synonym Resolution.

6.1.2 HTML Parser

- Increasing the *coherence* of the summarized text by improving the parsing process and using HTML tags such as paragraph (<P>), lists: Ordered List (<OL>), Unordered List (<UL>), etc which operate at block-level[39] rather than sentence level.
- The HTML tag <STRONG> should be handled as a <BOLD> i.e. sentences containing this tag should get a higher score.
- Support for *frames* in the HTML code.
- Saving the *charset*[40] parameter in the HTML *header*. It can be used in recovering of the encoding in case it is missing in the final summary.

[35] It is a list of ordered items.

[36] It is a list of unordered items which is marked with a "*bullet*".

[37] Paragraph.

[38] SwordNet is developed at the department of Linguistics at Lund University: http://www.ling.lu.se/projects/Swordnet/

[39] A block contains several sentences.

[40] This parameter contains information about the used encoding in the document.

6.1.3 Structure

As mentioned earlier, the structure of SweSum should be changed. Three possible solutions are discussed below:

6.1.3.1 Solution I

The current program structure is preserved, but some units are improved:

1 Improvement of the parsing process.

2 Each language has its own module or at least similar languages are in the same programming block. For example a partition between languages according to the used encoding (Latin1, Unicode, etc) is suitable.

6.1.3.2 Solution II

The current HTML parser in solution I is replaced with an external HTML parser.

6.1.3.3 Solution III

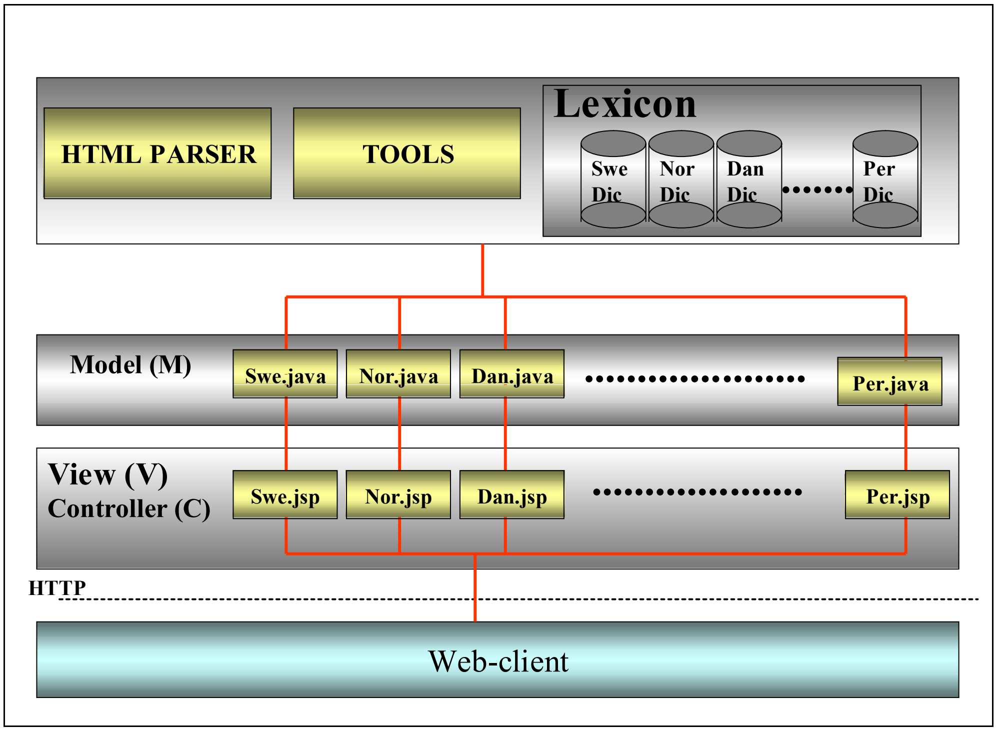


Figure 5: SweSum Structure (Java Solution)

The ideal solution is the object oriented approach i.e. using an external HTML parser and an object oriented programming language such as Java, C++ or Object Oriented Perl. Java is the best option since it has support for Unicode and provides a growing number of Internet tool resources such as Servlet[41], JSP[42], JavaBeans[43], etc.

[41] Servlets are Java technology's answer to CGI programming. They are programs that run on a Web server and build Web pages. Java servlets are more efficient, easier to use, more powerful, more portable, and cheaper than traditional CGI technologies.

An architecture proposal for implementation of SweSum using JSP and JavaBeans is shown in Figure 5. The building blocks are JSP-pages and JavaBeans in an MVC-architecture. The goal of the *Model-View-Controller* (MVC) design pattern is to separate the application object (model) from the way it is represented to the user (view) from the way in which the user controls it (controller).

The "C" (Controller) examines the requests from the user, invokes JavaBeans, controls error handling and module flow.
The "M" (Model) consists of a set of JavaBeans which:

- Implement the summarization methods.
- Handle the database (lexicon) access.
- Handle access to other external resources such as HTML parser.

The "V" (View) is used for presentation of the data fetched by the beans, i.e. generation of the final summary.
The controller and the view are implemented as JSP-pages or as Java Servlets.
For each language in the application there are one controller, one view, and one or more JavaBeans.

6.2 FarsiSum

Future improvements include some language-specific solutions in the tokenization process, new evaluation algorithms, etc. in addition to improvements suggested earlier for SweSum.

Tokenizer
The tokenization process is an important part of any NLP application. For Arabic script languages such as Persian, however it plays a more crucial role due to lack of representation of short vowels in the script and word/phrase ambiguities. The current tokenizer should be extended in order to recognize the final forms of the characters indicating word boundaries. Handling of other syntactic ambiguities (phrase, morphology) requires syntactic/semantic analysis.

Topic Identification
In the current implementation, the important keywords are identified using a very simple stop-list containing almost 200 words. I.e. it cannot exclude all the verbs and function words (not included in the stop-list) from the keyword-list, due to the limited size of the stop-list.
Furthermore, two identical words with different inflections counts as two different words. These problems can be handled either by extending the size of the stop-list or having access to NLP applications such as a Persian dictionary.

[42] JavaServer Pages (JSP) is a web-scripting technology similar to Microsoft Active Server Pages (ASP). However, it's more easily extensible than ASP, and it isn't proprietary to any one vendor or any particular web server. Although the JSP specification has been managed by Sun Microsystems, any vendors can implement JSP in their own systems.

[43] A JavaBeans component is an object that conforms to a communication and configuration protocol, as prescribed by the JavaBeans specification.

Language-specific solutions

The sentences in a text are scored in SweSum by using a *simple combination function*, in which the values of the different parameters (title, numerical data, etc.) are assigned manually. These empirical initial values for Swedish texts (currently used in the sentence scoring procedure by FarsiSum) should be adapted to the Persian text parameters, in the future versions.

New Methods

Implementation of some new modules may help to increase the quality of summaries:

- Resolving acronyms and abbreviations.
- Implementation of co-reference methods such as *Pronoun Resolution*, recognition of personal names, known places, etc.
- Using new evaluation methods such as *gold standard* by creating a Persian extract corpus.

7 Conclusion

- As expected the field test showed that despite the ambiguity problems in Persian texts and use of a very simple stop-list, the final summary was improved both in the *coherence* and the preservation of *important information*.
- Use of an object oriented programming language which has support for Unicode, in the implementation of the future versions of SweSum is necessary.
- Tokenization process in languages using an Arabic writing system is different due to lack of representation of short vowels in the script and word/phrase ambiguities.
- Most of methods used in SweSum are applicable to Persian but in some cases language-specific solutions are required. For example the initial scoring values are empirical and language-dependent.
- To use co-reference methods such as *Pronoun Resolution*, *Synonym Resolution*, recognition of personal names, known places, etc in order to make the summarized text more coherent.

Acknowledgement

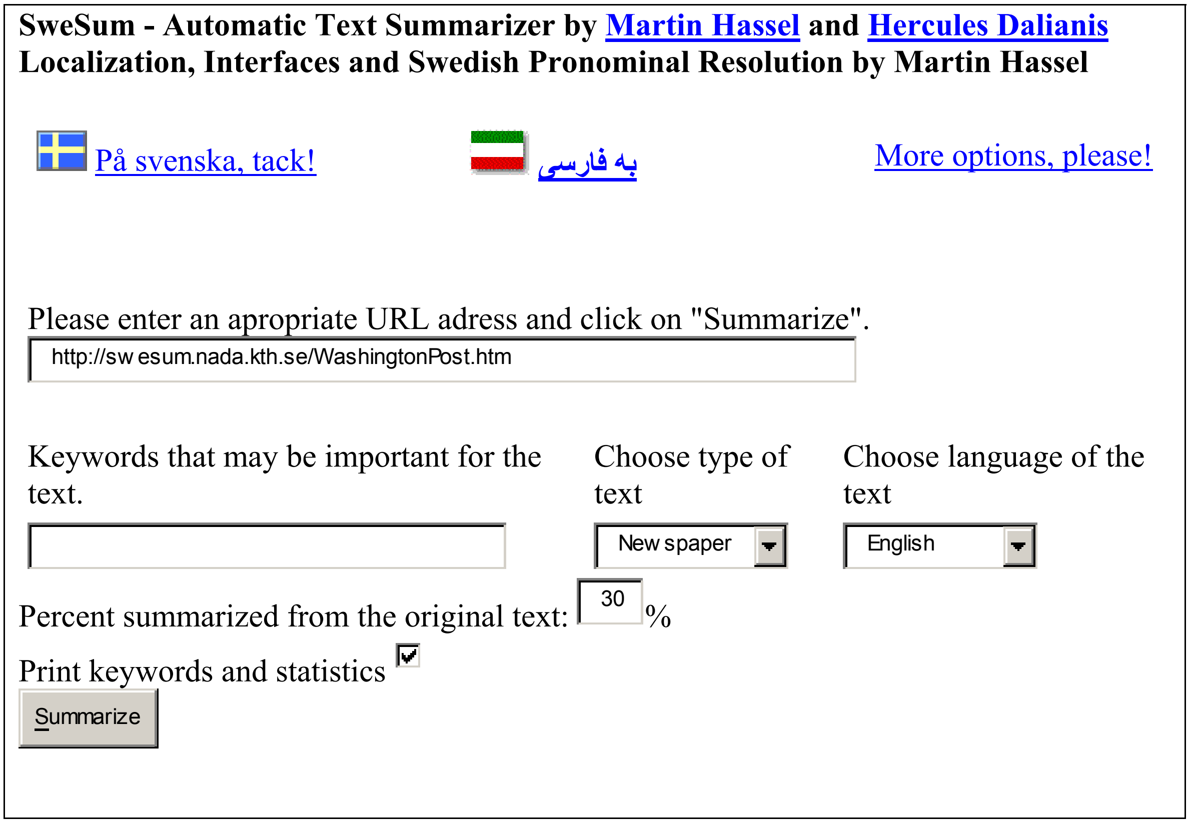
I would like to thank my supervisors: Prof. Martin Volk at Stockholm University, Dr. Hercules Dalianis and Martin Hassel at KTH for their guidance and support.

I would also like to thank Dr. Jennifer Spenader for her guidance and helpful comments on my thesis.

Finally I would like to thank all the participants in the field test.

8 Run the program

Go to http://swesum.nada.kth.se/index.html



SweSum - Automatic Text Summarizer by Martin Hassel and Hercules Dalianis
Localization, Interfaces and Swedish Pronominal Resolution by Martin Hassel

På svenska, tack! به فارسی More options, please!

Please enter an apropriate URL adress and click on "Summarize".

http://swesum.nada.kth.se/WashingtonPost.htm

Keywords that may be important for the text.

Choose type of text: Newspaper

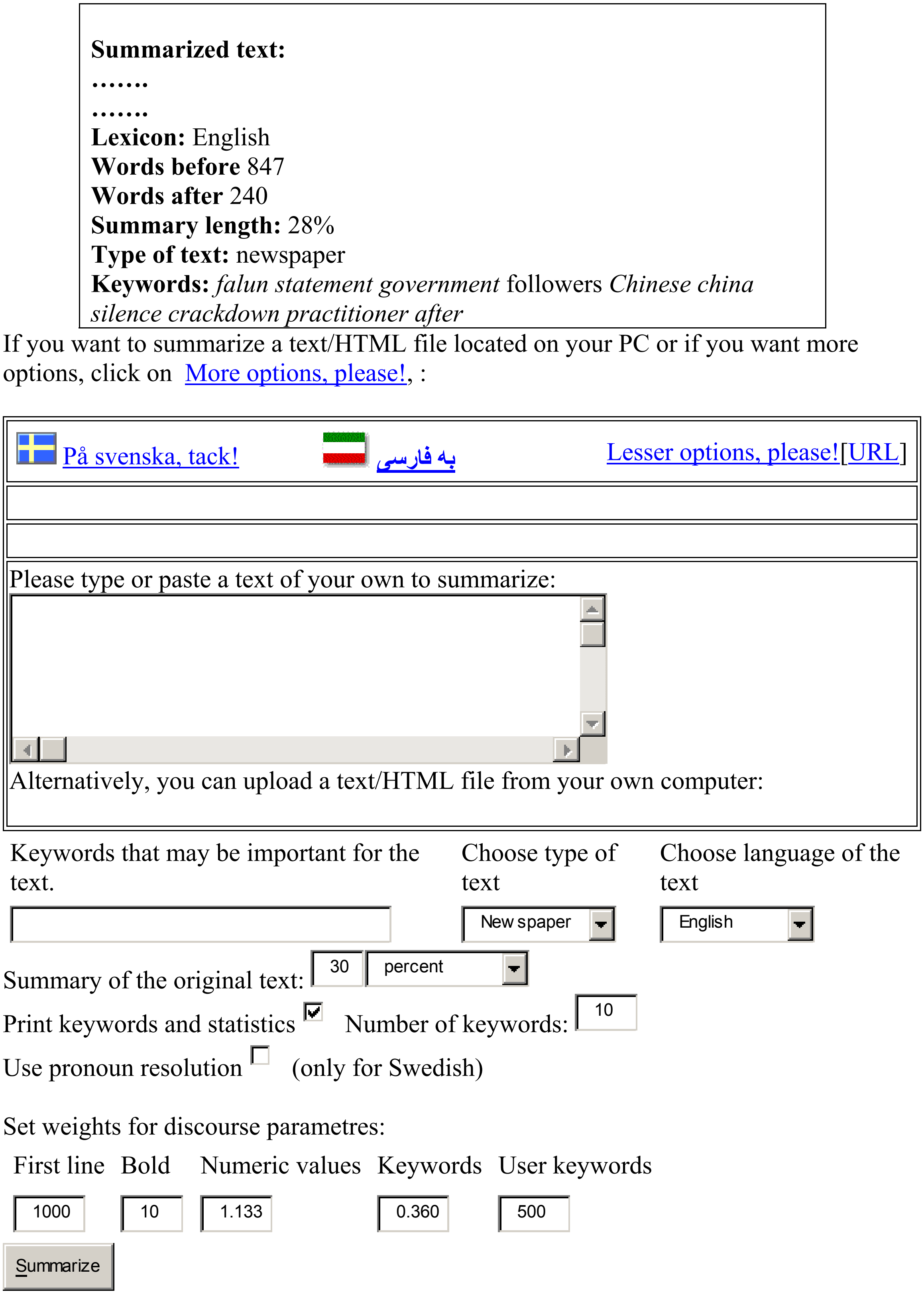
Choose language of the text: English

Percent summarized from the original text: 30 %

Print keywords and statistics ☑

This page is also available in Swedish and Persian. [Appendix B]
To summarize a text:

- Choose an appropriate URL. Default URL: http://swesum.nada.kth.se/WashingtonPost.htm
- Choose language of the text. Available languages are Swedish, English, Persian, German, Spanish, Danish, French, Norwegian and generic.
- Choose type of text: *newspaper* or *academic*.
- Important keywords (user defined).
- Percent rate of the text to be summarized (0-100 %).
- Print statistics (Yes or No).
- Click the summarize button.
- The summarized text will be displayed:



Summarized text:
…….
…….
Lexicon: English
Words before 847
Words after 240
Summary length: 28%
Type of text: newspaper
Keywords: *falun statement government* followers *Chinese china silence crackdown practitioner after*

If you want to summarize a text/HTML file located on your PC or if you want more options, click on More options, please!, :

På svenska, tack! به فارسی Lesser options, please![URL]

Please type or paste a text of your own to summarize:

Alternatively, you can upload a text/HTML file from your own computer:

Keywords that may be important for the text.	Choose type of text	Choose language of the text
	New spaper	English

Summary of the original text: 30 percent

Print keywords and statistics ☑ Number of keywords: 10

Use pronoun resolution □ (only for Swedish)

Set weights for discourse parametres:

First line	Bold	Numeric values	Keywords	User keywords
1000	10	1.133	0.360	500

References

Boguraev, B. and Kennedy, C. 1997. Salience-based Content Characterization of Text Documents. In Proceedings of the Workshop on Intelligent Scalable Text Summarization at the ACL/EACL Conference, 2-9. Madrid, Spain.

Brunn, M., Chali, Y. and Pincha, C.J. 2001. Text summarization using lexical chains, in Document Understanding Conference (DUC), New Orleans, Louisiana USA, September 13-14, 2001.

Dalianis, H. 2000. SweSum - A Text Summarizer for Swedish, Technical report, TRITA-NA-P0015, IPLab-174, NADA, KTH, October 2000.

Dalianis, H. and Hassel, M. 2001. Development of a Swedish Corpus for Evaluating Summarizers and other IR-tools. Technical report, TRITA-NA-P0112, IPLab-188, NADA, KTH, June 2001

Dalianis, H. and Åström, E. 2001. SweNam - A Swedish Named Entity recognizer, its construction, training and evaluation. Technical report, TRITA-NA-P0113, IPLab-189, NADA, KTH.

Dalianis, H., Hassel, M., de Smedt, K., Liseth, A., Lech, T.C. and Wedekind, J. 2003. Porting and evaluation of automatic summarization. In Holmboe, H. (ed.) Nordisk Sprogteknologi 2003. Årbog for Nordisk Språkteknologisk Forskningsprogram 2000-2004 Museum Tusculanums Forlag 2004 (Forthcoming).

Edmundson, H.P. 1969. New Methods in Automatic Extraction. Journal of the ACM 16(2) pp 264-285.

Fallahi, S. 2003. Presentation at Fifth ScandSum network meeting, Jan 25-28, 2003, Norway.

Gong, Y. and Liu, X. 2001. Generic text summarization using relevance measure and latent semantic analysis. In Proceedings of SIGIR 2001, page 19-25.

Hassel, M. 2000. Pronominal Resolution in Automatic Text Summarisation. Master Thesis, University of Stockholm, Department of Computer and Systems Sciences (DSV).

Hassel, M. 2003. Exploitation of Named Entities in Automatic Text Summarization for Swedish. In Proceedings of NODALIDA 03 - 14th Nordic Conference on Computational Linguistics, May 30-31 2003, Uppsala, Sweden.

Karlgren, J. and Sahlgren, M. 2001. Vector-based Semantic Analysis using Random Indexing and Morphological Analysis for Cross-Lingual Information Retrieval, Technical report, SICS.

Kupiec, J., Pedersen, J.O. and Chen, F. 1995. A Trainable Document Summarizer. In Research and Development in Information Retrieval, 68–73.

Landauer, T., Laham, K. and Foltz, D. 1998. Learning human-like knowledge by Singular Value Decomposition: A progress report. In M. I. Jordan, M. J. Kearns & S. A. Solla (Eds.), Advances in Neural Information Processing Systems 10, (pp. 4551). Cambridge: MIT Press.

Lazard, G. 1992. Grammar of contemporary Persian. Costa Mesa, California: Mazda Publishers, 1992.

Lin, C.Y. 1995. Knowledge Based Automated Topic Identification. In the Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics. Cambridge, Massachusetts, USA, June 1995.

Lin, C.Y. and Hovy, E. 1997. Identify Topics by Position, Proceedings of the 5th Conference on Applied Natural Language Processing, March.

Lin, C.Y. 1999. Training a Selection Function for Extraction. In the 8th International Conference on Information and Knowledge Management (CIKM 99), Kansa City, Missouri, November 2-6, 1999.

Luhn, H.P. 1959. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development pp 159-165.

Mani, I. and Maybury, M. 1999. Advances in Automatic Text Summarization, MIT Press, Cambridge, MA, 1999. ISBN: 0262133598, Publisher: MIT Press

Megerdoomian, Karine and Rémi, Zajac 2000.
Processing Persian Text: Tokenization in the Shiraz Project. NMSU, CRL, Memoranda in Computer and Cognitive Science (MCCS-00-322).

Neto, L., Freitas, A. and Kaestner, C. 2002. In G Bittencourt and GL Ramalho, editors, Proc. 16th Brazilian Symp. on Artificial Intelligence (SBIA-2002). Lecture Notes in Artificial Intelligence 2507, pages 205-215. Springer-Verlag, November 2002.)

Wiemer-Hastings, P. 1999. How Latent is Latent Semantic Analysis? in Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99), Stockholm, 1999.

Appendix A: HTML, XML & XHTML

HTML

The basic layout of an HTML document is shown below:

HTML	Displayed in a browser
<!**DOCTYPE** HTML PUBLIC "-//IETF//DTD HTML//EN"> <**HTML**> <**HEAD**> <**TITLE**>Title of the web page </**TITLE**> </**HEAD**> <**BODY**> The <**B**>text.</**B**> is bolded<**BR**> The <**I**>text </**I**> is italicized.<**BR**> The <**U**>text</**U**> is underlined.<**BR**> </**BODY**> </**HTML**>	The **text** is bolded. The *text* is italicized. The text is underlined.

<! DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
Each HTML document starts with DOCTYPE (Document Type Declaration) that declares to the browser the version of the current HTML document.
<**HTML**> <**HTML**>

<**HEAD**> </**HEAD**>
Contains *header information* about the document such as, its title, keywords, description and style sheet.
<**TITLE**> </**TITLE**>
The title of the document.
<**BODY**></**BODY**>
Contains the document's content.
<**B**> </**B**>
Indicates bold text.
<**I**> </**I**>
Indicates italic text.
<**U**></**U**>
Indicates underlined text.
<**BR**>
Indicates line break.
There are some other tags that can be used in the summarization process:
<OL> **Ordered List**: Defines a list of items which are automatically ordered; typically, this is a sequential numbering from one to the number of list items.
<**UL**> **Unordered List**: Defines a list of items which are automatically marked with a "bullet"; typically, this is a solid disc or asterisk.

<**P**> **Paragraph**: Defines a section of text as being a paragraph. The closing tag (</P>) is technically optional, but its use is strongly recommended.
<**STRONG**> **Strong:** Causes text to be strongly emphasized. The display of this text varies by browser and user, but the suggested default is boldfaced text.
<**U**> **Underline**: Causes text to be underlined. Not supported by all browsers.

XML

E**X**tensible **M**arkup **L**anguage (XML) is a markup language for documents containing structured information. HTML is used to represent data while XML is designed to store and exchange data between applications.
Unlike HTML, XML does not have predefined tags. You must define your own tags.
XML uses a Document Type Definition (DTD) or an XML Schema to describe the data. The purpose of a Document Type Definition is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.
XML provides tagging capability for representation of text documents. The XML-tagged representation can be used for NLP applications such as text summarization, information extraction, and other analyses. The availability of XML-tagged representations of documents provides a significant set of opportunities for further improvements of NLP applications.

XHTML

XHTML stands for e**x**tensible **H**yper**T**ext **M**arkup **L**anguage and its goal is to replace HTML. It is very similar to HTML 4.0, but its syntax is stricter than HTML. XHTML was created for two main reasons:

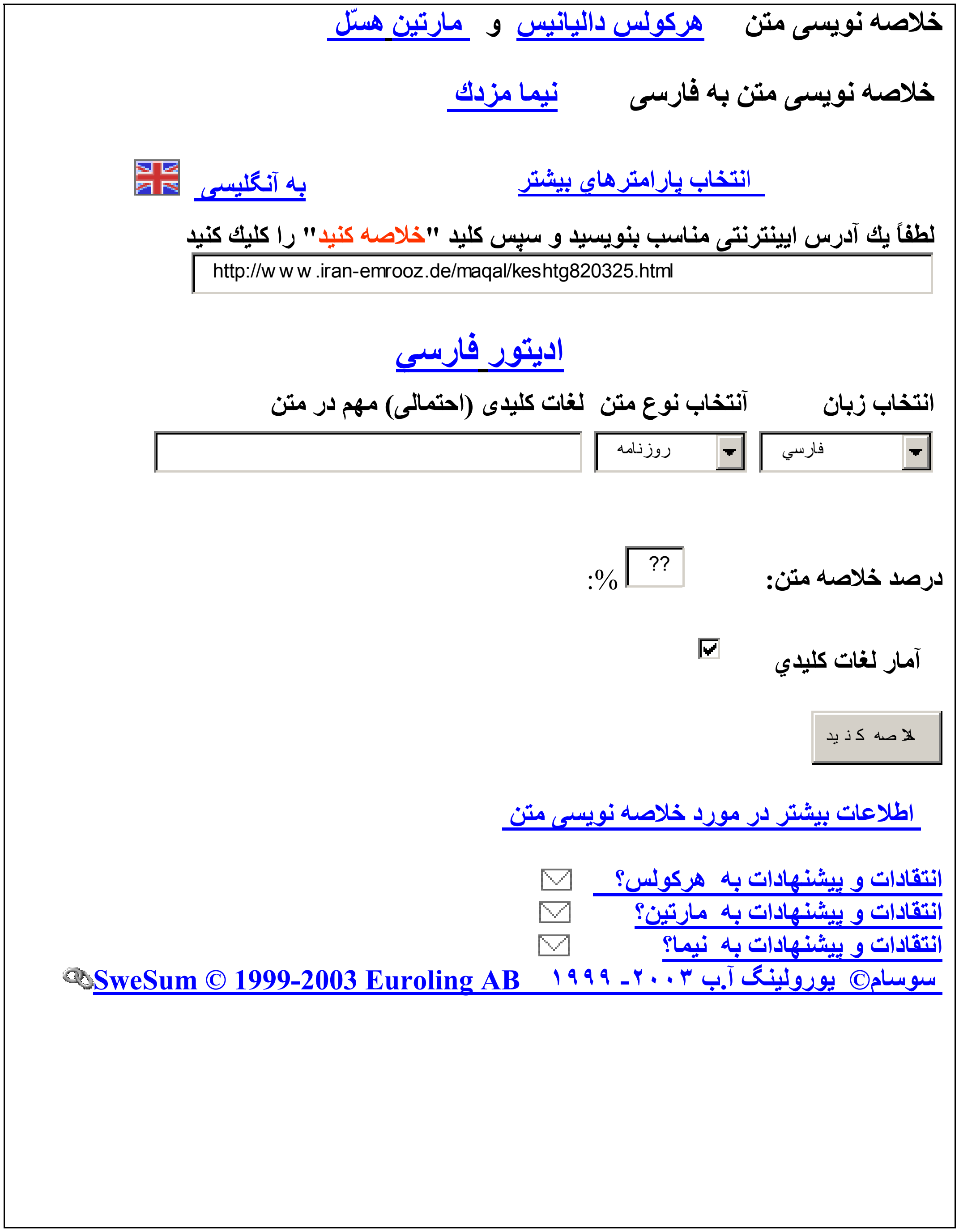
- To create a stricter standard for making web pages, reducing incompatibilities between browsers
- To create a standard that can be used on a variety of different devices without changes

There are several main changes in XHTML from HTML:

- All tags must be in lower case
- All documents must have a doctype (Document Type Declaration).
- All documents must be properly formed
- All tags must be closed
- All attributes must be added properly
- The name attribute has changed
- Attributes cannot be shortened
- All tags must be properly nested

An XHTML document optionally starts with an xml declaration:
<? xml version=”1.0” encoding=”ISO-8859-1”?>
The xml declaration is not required in all XHTML documents, unless the character encoding is other than UTF-8, or UTF-16, but it is good practice to include it.

Appendix B: User Interface



خلاصه نویسی متن   هرکولس دالیانیس و مارتین هسّل

خلاصه نویسی متن به فارسی   نیما مزدك

انتخاب پارامترهای بیشتر   به آنگلیسی

لطفاً يك آدرس ایینترنتی مناسب بنویسید و سپس کلید "خلاصه کنید" را کلیك کنید

http://w w w .iran-emrooz.de/maqal/keshtg820325.html

ادیتور فارسی

انتخاب زبان	آنتخاب نوع متن	لغات کلیدی (احتمالی) مهم در متن
فارسي	روزنامه	

درصد خلاصه متن: ?? %:

آمار لغات کلیدي ☑

لاصه کنید

اطلاعات بیشتر در مورد خلاصه نویسی متن

انتقادات و پیشنهادات به هرکولس؟

انتقادات و پیشنهادات به مارتین؟

انتقادات و پیشنهادات به نیما؟

سوسام© یورولینگ آ.ب ۱۹۹۹ -۲۰۰۳ SweSum © 1999-2003 Euroling AB

Appendix C: Results from the Field Test

P1…P7 → Participant 1-7
T1…T3 → Text 1-3
M1 → FarsiSum in the mode one (the stop-list enabled)
M2 → FarsiSum in the mode two (the stop-list disabled)
M3 → SweSum in the *Generic mode*

Table A: The best method

Text	P1	P2	P3	P4	P5	P6	P7	Method Result
T1	M1	M1	M1	M3	M2	M1	M3	M1(4), M2(1), M3(2)
T2	M1	M1	M3	M3	M3	M1	M1	M1(4), M2(0), M3(3)
T3	M1	M1	M1	M3	M2	M2	M3	M1(3), M2(2), M3(2)

Table B: The best method (score 1-5)

Text Method	P1	P2	P3	P4	P5	P6	P7	Method Result
T1 M1	**5**	**5**	**4**	**2**	**3**	**5**	**3**	**27**
T1 M2	**4**	**4**	**2**	**3**	**4**	**4**	**2**	**23**
T1 M3	**3**	**2**	**1**	**4**	**3**	**2**	**4**	**19**
T2 M1	**4**	**5**	**4**	**3**	**3**	**4**	**4**	**27**
T2 M2	**3**	**4**	**2**	**2**	**3**	**3**	**3**	**20**
T2 M3	**2**	**3**	**5**	**4**	**5**	**3**	**3**	**25**
T3 M1	**5**	**5**	**5**	**2**	**3**	**1**	**3**	**24**
T3 M2	**4**	**4**	**2**	**3**	**3**	**2**	**2**	**20**
T3 M3	**2**	**3**	**3**	**4**	**4**	**3**	**4**	**23**

Table C: Cohesion

Text	P1	P2	P3	P4	P5	P6	P7	Method Result
T1	M1,M2	M 1,M 2	M 1	M 3	M 2	M1	M1,M3	M1(5), M2(3), M3(2)
T2	M1,M2	M1, M 2	M 1,M3	M 3	M 3	M2	M1,M3	M1(4), M2(3), M3(4)
T3	M 1,M2	M1,M 2	M 1	M 3	M 3	M2	M1, M2 M3	M1(4), M2(4), M3(3)

Table D: Important information preserved

Text	P1	P2	P3	P4	P5	P6	P7	Method Result
T1	M1, M2	M1	M1, M2	M3	M2	M1	M3	M1(4),M2(3),M3(2)
T2	M1,M2	M1	M1, M3	M1	M1	M1	M1	M1(7),M2(1),M3(1)
T3	M1, M2	M1	M1,M2,M3	M3	M1	M2, M3	M1, M3	M1(5),M2(3),M3(4)

Appendix D: The Stop-list

Pronoun	من آقا آقای آقایان آن آنان آنکه آنها او این ایشان اینکه این برخي تو خود خودم خودمان خویش شما ما
Conjunction& quantifier	آیا اما اگر البته اول اولین ای چند چه دوم که می و ولی ها هم هر یا یعنی
Adverb	آنجا اکنون امروز اینجا بسیار بسیاری بطور بیش تمامی جا چنان چنین حقیقتا علیرغم فقط همان هیچ هنوز
Preposition	از با بدون بجز بر برای به بی پس پیش تا توی توسط جز داخل در درباره درین را روی سوی علیه غیر کنار میان
Verb	افزود است باشد باشید باشیم بدهید بکنید بگذاریم بگوییم بماند بود بودند بوده خواهد شود | خواهند داد دادم دادند داده دارد دارند داریم داشت داشته دانند رسیده شد شدند شده باشد |کرد کردم کردن کردند کرده کند کنید کنیم گردند گرفت گرفته گفت گفتم گفتند گفته می |کنم ندارد ندارم ندارند نداشته نمی|کند می|کنند می|شود می|کنم می|رسد می|داند می|تواند می| خواهیم نماید نموده نیست نیستند هست هستیم هستند

Appendix E: Persian Arabic Unicode

Arabic Unicode: http://www.alanwood.net/unicode/arabic.html

Character	Decimal	Hex	Name
،	1548	060C	ARABIC COMMA
؛	1563	061B	ARABIC SEMICOLON
؟	1567	061F	ARABIC QUESTION MARK
ء	1569	0621	ARABIC LETTER HAMZA
آ	1570	0622	ARABIC LETTER ALEF WITH MADDA ABOVE
أ	1571	0623	ARABIC LETTER ALEF WITH HAMZA ABOVE
ؤ	1572	0624	ARABIC LETTER WAW WITH HAMZA ABOVE
إ	1573	0625	ARABIC LETTER ALEF WITH HAMZA BELOW
ئ	1574	0626	ARABIC LETTER YEH WITH HAMZA ABOVE
ا	1575	0627	ARABIC LETTER ALEF
ب	1576	0628	ARABIC LETTER BEH

ة	1577	0629	ARABIC LETTER TEH MARBUTA
ت	1578	062A	ARABIC LETTER TEH
ث	1579	062B	ARABIC LETTER THEH
ج	1580	062C	ARABIC LETTER JEEM
ح	1581	062D	ARABIC LETTER HAH
خ	1582	062E	ARABIC LETTER KHAH
د	1583	062F	ARABIC LETTER DAL
ذ	1584	0630	ARABIC LETTER THAL
ر	1585	0631	ARABIC LETTER REH
ز	1586	0632	ARABIC LETTER ZAIN
س	1587	0633	ARABIC LETTER SEEN
ش	1588	0634	ARABIC LETTER SHEEN
ص	1589	0635	ARABIC LETTER SAD
ض	1590	0636	ARABIC LETTER DAD
ط	1591	0637	ARABIC LETTER TAH
ظ	1592	0638	ARABIC LETTER ZAH
ع	1593	0639	ARABIC LETTER AIN
غ	1594	063A	ARABIC LETTER GHAIN
ـ	1600	0640	ARABIC TATWEEL
ف	1601	0641	ARABIC LETTER FEH
ق	1602	0642	ARABIC LETTER QAF
ك	1603	0643	ARABIC LETTER KAF
ل	1604	0644	ARABIC LETTER LAM
م	1605	0645	ARABIC LETTER MEEM
ن	1606	0646	ARABIC LETTER NOON
ه	1607	0647	ARABIC LETTER HEH

و	1608	0648	ARABIC LETTER WAW
ى	1609	0649	ARABIC LETTER ALEF MAKSURA
ي	1610	064A	ARABIC LETTER YEH
٠	1632	0660	ARABIC-INDIC DIGIT ZERO
١	1633	0661	ARABIC-INDIC DIGIT ONE
٢	1634	0662	ARABIC-INDIC DIGIT TWO
٣	1635	0663	ARABIC-INDIC DIGIT THREE
٤	1636	0664	ARABIC-INDIC DIGIT FOUR
٥	1637	0665	ARABIC-INDIC DIGIT FIVE
٦	1638	0666	ARABIC-INDIC DIGIT SIX
٧	1639	0667	ARABIC-INDIC DIGIT SEVEN
٨	1640	0668	ARABIC-INDIC DIGIT EIGHT
٩	1641	0669	ARABIC-INDIC DIGIT NINE

Appendix F: A Summary sample created by FarsiSum

The Summary (30%)

آثار و نتايج اعتراضات اخير دانشجويان

- جنبش دانشجويی اگر در مضمون قاطع و در روش مسالمت آميز بماند شکست ناپذير می‌شود.
- دشمنان آزادی فقط زمانی می‌توانند مجموعه گسترده نيروهای مسلح را برای سرکوب، هماهنگ و منسجم کنند که اين جنبش به خشونت روی آورد

علی کشتگر
يکشنبه ٢٥ خرداد ١٣٨٢

اما اعتراضات دانشجويان تمامی اين نقشه را نقش برآب کرد. اما نتايج اعتراضات اخير دانشجويی به اين جا ختم نمی‌شود.

شرط گسترش اين جنبش در حدی که اکثريت دانشجويان را حول دفاع از آزادی فعال کند، پيوند با جنبش عمومی را ممکن سازد، و احتمال سرکوب جنبش دانشجويی را به حداقل برساند آن است که:

١- جنبش دانشجويی به هيچ وجه از روش‌های مسالمت آميز فاصله نگيرد و در دام درگيری‌های خشونت آميز درنغلتد. دشمنان آزادی فقط زمانی می‌توانند مجموعه گسترده نيروهای مسلح را برای سرکوب، هماهنگ و منسجم کنند که جنبش دانشجويی به خشونت روی آورد. برعکس هرچه جنبش دانشجويی درعين قاطعيت در مضمون و خواسته‌ها، بيشتر به روش‌های مسالمت آميز مبارزه عليه استبداد وفادار بماند، طيف عظيم تری از دانشجويان را به حرکت درمی آورد و سرکوب آن، هم به دليل ناتوانی دشمنان آزادی در توجيه نيروهای مسلح به تهاجم مسلحانه عليه اقدامات مسالمت جويانه دانشجويان و هم به

دلیل گسترده بودن جنبش دشوارتر می‌شود.

۲- فعالان جنبش دانشجویی باید راههای حرکات اعتراضی هماهنگ و سازمان یافته را بررسی کنند و هرچه زودتر تشکیلات لازم برای سراسری کردن و همزمان کردن اعتراضات را ایجاد نمایند. ایجاد سلولها و کمیته‌های هماهنگ کننده فعالیت دانشجویی مهمترین بخش چنین تشکیلاتی است. اگر جنبش دانشجویی بتواند در پی افت و خیزهای خود به يك جنبش سراسری وسازمان یافته تبدیل شود و صدها هزار دانشجو را در حرکات همزمان و هماهنگ به فعالیت علیه استبداد بکشاند بی‌تردید تحول بزرگی در راه شکل گیری جنبش همگانی ملی علیه استبداد در ایران پدید می‌آید، تحولی که می‌تواند تا يك قیام ملی ارتقاء یابد.

شنبه ٢٤ خردادماه
علی کشتگر

The Original Text

آثار و نتایج اعتراضات اخیر دانشجویان

- جنبش دانشجویی اگر در مضمون قاطع و در روش مسالمت آمیز بماند شکست ناپذیر می‌شود.
- دشمنان آزادی فقط زمانی می‌توانند مجموعه گسترده نیروهای مسلح را برای سرکوب، هماهنگ و منسجم کنند که این جنبش به خشونت روی آورد

علی کشتگر
یکشنبه ۲۵ خرداد ۱۳۸۲

نخستین اثر اعتراضات اخیر دانشجویان، بی‌اثر کردن تلاشهای اقتدارگرایان دردستیابی به توافق پنهان و یا آشکار با آمریکا است. چرا که هرچه مخالفت‌ها علیه استبداد آشکارتر ابراز شود،شانس اقتدارگرایان در قبولاندن خود به آمریکا کمتر می‌شود.
آقایان خامنه‌ای و رفسنجانی امیدواربودند که با کنترل اوضاع داخلی و به حاشیه راندن رقبای اصلاح طلب خود به آمریکائیها بفهمانند که همه کاره ایران زمین ما هستیم و در این سرزمین محکم میخ خود را کوبیده ایم. پس اگر رابطه با ایران می‌خواهید بفرمائید با ما تفاهم کنید. البته سران حاکمیت برای ماندن بر سریر قدرت اهل معامله و امتیاز دادن هم هستند بویژه آن که مامور عادی سازی رابطه ایران با آمریکا آقای‌هاشمی رفسنجانی باشد. و اگر تا امروز هم چنین توافقی با آمریکا حاصل نشده به آن دلیل است که دولت آمریکا تا به امروز نپذیرفته است که با رژیم بی‌آینده کنونی توافق کند.

پیشتر نوشته بودم اگر اقتدارگرایان بتوانند نشان دهند که بر اوضاع مسلط هستند و با چالش‌ها و اعتراضات داخلی دست به گریبان نیستند، آن وقت شاید فرصت پیدا کنند با دادن امتیاز و پذیرش شرایط آمریکا، با تنها ابرقدرت جهان که نقش عظیمی در تحولات منطقه پیدا کرده است به توافق برسند و دشمنی آن را به دوستی با خود بدل کنند و از این راه عمر استبداد را درایران طولانی‌تر کنند. طراحان این سیاست خوب می‌دانستند که عادی سازی رابطه با آمریکا در شرایط کنونی برکات زیادی دارد. که تبدیل تمایلات دوستانه نسل جوان ایران نسبت به آمریکا به گرایش دشمنانه(بخوانید محروم کردن آمریکا از مهمترین ذخیره استراتژيك خود در منطقه) ایجاد روحیه انفعالی در این نسل و بازشدن دست آنها در سرکوب داخلی از جمله این برکات است. اما اعتراضات دانشجویان تمامی این نقشه را نقش برآب کرد. پس به هوشیاری دانشجویان و جوانان که می‌دانند چه می‌خواهند و می‌دانند چه کارمی کنند باید آفرین گفت. اما نتایج اعتراضات اخیر دانشجویی به این جا ختم نمی‌شود. حرکات اعتراضی چندروز گذشته دانشجویی نشان می‌دهد که نقشه فاشیستهایی که طرح تهاجم خونین به کوی دانشگاهها را ریخته و پس از آن نیز فعالان جنبش دانشجوئی را زندانی و مجریان آن حرکت جنایتکارانه را تشویق کردند تا بدینوسیله جنبش دانشجویی را منفعل کنند، شکست خورده است. دشمنان آزادی پس از آن همه بیرحمی و حق کشی نسبت به جنبش دانشجویی و آن همه مصلحت جویی‌های ذلیلانه اصلاح طلبان در برابر تهاجم به حقوق ملت و سرکوب دانشجویان امیدوار بودند که با غیرسیاسی کردن دانشجو و دانشگاه، جنبش دانشجویی، یعنی موتور به حرکت درآورنده جنبش آزادیخواهانه ملی را خاموش کنند. نتیجه کار آنها اما برعکس شده است. جنبش دانشجویی با توهم زدایی نسبت به همه جناحهای حکومتی و هماهنگی و همدلی بی‌سابقه در دفاع از آزادی و حقوق بشر ظرف دو سال گذشته به همانجائی رسیده است که برای تبدیل شدن به يك جنبش مستقل و نیرومند ضد استبدادی باید می‌رسید. آزادی يك امرطبقاتی نیست. بلکه همه طبقات بجز گروه اندك خودکامگان حاکم به آن نیازمندند. آزادی‌های فردی و سیاسی با سرنوشت ایران و

منافع و مصالح حال و آینده ملت ایران گره خورده است. دانشجویان نیز از يك طبقه معین نیستند از همه طبقات و اقشار جامعه‌ای هستند که می‌خواهد به هرقیمتی شده دوران طولانی و دردناك استبداد را پشت سر گذارد. دانشجویان وقتی بر سر يك خواست ملی هماهنگ و همدل شوند، همه مردم را همراه و هماهنگ می‌کنند. چنین کاری در ایران از هیچ حزب و گروه و طبقه‌ای بجز سپاه میلیونی دانشجویان که متعلق به همه ملت و فرزندان همه مردم ایران‌اند ساخته نیست.

جنبش دانشجویی ایران حالا دیگر بر سر خواست مبرم جامعه همدل و هماهنگ است، اما روش‌هایی که در مبارزه علیه استبداد به کار خواهد گرفت ونیز ظرفیت آن در انجام اعتراضات سازمان یافته، نامعلوم و هنوز مورد سوال است. شرط گسترش این جنبش در حدی که اکثریت دانشجویان را حول دفاع از آزادی فعال کند، پیوند با جنبش عمومی را ممکن سازد، و احتمال سرکوب جنبش دانشجویی را به حداقل برساند آن است که:

١- جنبش دانشجویی به هیچ وجه از روش‌های مسالمت آمیز فاصله نگیرد و در دام درگیری‌های خشونت آمیز درنغلتد. دشمنان آزادی فقط زمانی می‌توانند مجموعه گسترده نیروهای مسلح را برای سرکوب، هماهنگ و منسجم کنند که جنبش دانشجویی به خشونت روی آورد. چرا که در این صورت اولا بخش عظیمی از دانشجویان از این حرکات فاصله می‌گیرند و در نتیجه حرکات اعتراضی ضعیف و قابل سرکوب می‌شوند و ثانیا آن که استبداد نیز برای سرکوب توجیه پیدا می‌کند. برعکس هرچه جنبش دانشجویی درعین قاطعیت در مضمون و خواسته‌ها، بیشتر به روش‌های مسالمت آمیز مبارزه علیه استبداد وفادار بماند، طیف عظیم تری از دانشجویان را به حرکت درمی آورد و سرکوب آن، هم به دلیل ناتوانی دشمنان آزادی در توجیه نیروهای مسلح به تهاجم مسلحانه علیه اقدامات مسالمت جویانه دانشجویان و هم به دلیل گسترده بودن جنبش دشوارتر می‌شود. فراموش نکنیم که نیروهای مسلح نیز همگی دنباله رو مافیای سیاسی حاکم برایران نیستند و چه بسا که بسیاری از افراد نیروهای مسلح با دانشجویان درد مشترك و خواست مشترك دارند. دانشجویان باید حساب آنان را از حساب سران رژیم و گروه کوچك انصار حزب الله جدا کنند.

٢- فعالان جنبش دانشجویی باید راههای حرکات اعتراضی هماهنگ و سازمان یافته را بررسی کنند و هرچه زودتر تشکیلات لازم برای سراسری کردن و همزمان کردن اعتراضات را ایجاد نمایند. ایجاد سلولها و کمیته‌های هماهنگ کننده فعالیت دانشجویی مهمترین بخش چنین تشکیلاتی است. اگر جنبش دانشجویی بتواند در پی افت و خیزهای خود به يك جنبش سراسری وسازمان یافته تبدیل شود و صدها هزار دانشجو را در حرکات همزمان و هماهنگ به فعالیت علیه استبداد بکشاند بی‌تردید تحول بزرگی در راه شکل گیری جنبش همگانی ملی علیه استبداد در ایران پدید می‌آید، تحولی که می‌تواند تا يك قیام ملی ارتقاء یابد.

شنبه ٢٤ خردادماه
علی کشتگر