An Efficient Mechanism for Stemming and Tagging: The Case of Greek Language

Giorgos Adam, Konstantinos Asimakis, Christos Bouras, and Vassilis Poulopoulos

Research Academic Computer Technology Institute, Greece
and
Computer Engineering and Informatics Department, University of Patras, Greece
{adam,asimakis,bouras,poulop}@ceid.upatras.gr

Abstract. In an era that, searching the WWW for information becomes a tedious task, it is obvious that mainly search engines and other data mining mechanisms need to be enhanced with characteristics such as NLP in order to better analyze and recognize user queries and fetch data. We present an efficient mechanism for stemming and tagging for the Greek language. Our system is constructed in such a way that can be easily adapted to any existing system and support it with recognition and analysis of Greek words. We examine the accuracy of the system and its ability to support peRSSonal a medium constructed for offering meta-portal news services to internet users. We present experimental evaluation of the system compared to already existing stemmers and taggers of the Greek language and we prove the higher efficiency and quality of results of our system.

Keywords: Greek stemmer, natural language processing, information tagging, context, keyword extraction.

1 Introduction

As the Internet expands dramatically more and more people are concerned about the difficulty in searching information on the WWW. The task of searching for useful information involves a large amount of procedures that are transparent to the end-user, who just needs to locate the information needed each time. One of the many procedures that are executed during the data processing is the stemming of words and root extraction. It remains an open issue of the academia whether an NLP procedure of a specific language should contain a stemmer or a lemmatizer. It seems that the solution is not as simple as it seems to be from a first analysis. In languages with large morphological linguistic variety a lemmatizer is more accurate while a stemmer may categorize words with extremely different meaning.

Another huge problem that exists, and is produced by the expansion of the web, is the fact that nowadays almost everybody is searching for information to their own language and not in English as it used to be some years ago. This implies that the data that exist on the Internet are written in many different languages. The Greek language, like many others, uses extensive inflection and therefore semantic relations between words cannot be detected unless those words are stemmed. Stemming Greek words is much harder than stemming words of other European languages because of the large

R. Setchi et al. (Eds.): KES 2010, Part III, LNAI 6278, pp. 389–397, 2010.

number of possible suffixes, many of which cannot be properly separated from the word stem without knowing the grammatical type of the word. Moreover, stemming of words with different meaning may lead to the same stem.

Many steps have been made towards the issue of construction of stemmers for many languages. The first stemmer that was ever presented was the effort made by Lovins. The Lovins stemming algorithm [1] was first presented in 1968 by Julie Beth Lovins. It is a single pass, context sensitive stemmer, which removes endings based on the longest-match principle. The stemmer was the first to be published and was extremely well developed considering the date of its release and has been the main influence on a large amount of the future work in the area. One of the most important efforts that became the basis of many other stemmers is Porter Stemmer [2]. The Porter Stemmer is a conflation Stemmer developed by Martin Porter at the University of Cambridge in 1980. The Stemmer is based on the idea that the suffixes in the English language are mostly made up of a combination of smaller and simpler suffixes. This Stemmer is a linear step Stemmer. Specifically it has five steps applying rules within each step. Within each step, if a suffix rule matched to a word, then the conditions attached to that rule are tested on what would be the resulting stem, if that suffix was removed, in the way defined by the rule. For example such a condition may be, the number of vowel characters, which are followed be a consonant character in the stem (Measure), must be greater than one for the rule to be applied. Finally, another commonly used stemmer is Paice-Husk stemmer [3] implemented at the University of Lancaster. The stemmer is a conflation based iterative stemmer. The stemmer, although remaining efficient and easily implemented, is known to be very strong and aggressive. A very common usage of the stemmers is the search engines and their query analysis and enhancement subsystems. The stemming procedure is applied to the input query in order to increase the recall rate, when it is necessary. It is important to note that the major search engines (such as Google[1] or Yahoo[2]) utilize an analyzer for the languages that they support. Despite the fact that such service is not referenced to, as it is transparent to the end user, it is obvious that it exists from the results presented to the end user. A Greek search engine that was constructed recently does utilize a stemmer; though, no information is publicly available except for a reference to the complete work that was done for the specific search engine [11].

The porter stemmer is the root of many stemmers that were produced for many European languages. The similarities on the construction of words in many European languages make it easy to construct stemmers starting from the basic Porter Stemmer. The basic categories to which we can possibly put a stemmer are three: Stemmers that are based (a) on dictionaries, (b) on algorithms, or (c) on hybrid algorithms that are based on both (a) and (b). We are putting the focus on the stemmers constructed for the Greek language. The first two are the TZK algorithm [4] by Kalamboukis and Nikolaidis in 1995 and the Automated Morphological Processor (AMP) [5] by Tambouratzis and Carayanis in 2001. The latest system presented was the work of George Ntais [6] while Spyridon Saroukos [7] has presented in 2008 an enhanced version of Ntais' stemmer.

[1] http://www.google.com. Google Search Engine
[2] http://www.yahoo.com. Yahoo! Search Engine

The first suffix stripping algorithm for the Greek language was presented in 1995 from Kalamboukis and Niloaidis. Design for information retrieval from Greek corpora the algorithm deals with inflections of the Greek language. They make extensive usage of suffix lists for inflection while in parallel; at a second level they remove derivational suffixes. Only 6 years later Tambouratzis and Carayannis presented a system for automated morphological categorization (AMP). Their work is based on extraction of stems through matching and masking with an initial set of correct stems and suffixes. A basic assumption for the usage of the system is that each word consists of a stem and a suffix. Every other word cannot be stemmed by their system. Nevertheless, because of the fact that in both the aforementioned effort no code or corpus was ever revealed it is impossible to make a direct comparison of a new stemmer.

The latest stemming algorithm developed for the Greek language is presented by George Ntais in 2006. The algorithm is based on Porter stemmer and an online implementation is available on the web. According to the author, the algorithm can handle a large number of suffixes of the Greek language, clearly outperforming the first two algorithms presented and aforementioned. A clear disadvantage of the system is the inability to manipulate with any other form of a word apart from the word in capital letters. Due to the fact that the morphology of the Greek language implies that a minor change to the accent mark of a word can change its meaning Ntais algorithm does have some weak points.

We propose a novel effort towards stemming for the Greek language. Our system is a hybrid system that is able to apply stemming on branches of texts without any limitation on the way text is written. The novelty of the system compared to the older efforts for the creation of a Greek stemmer is the fact that we are enhancing the stemming procedure by word tagging techniques. Tagging is as hard as stemming unless words are viewed in context. Greek syntax may be almost free but still some few useful rules apply which can be used to increase the accuracy of the grammatical tagging. Additionally, proper tagging enables us to ignore words based on their grammatical type instead of ignoring words using word length thresholds or stopwords lists.

The rest of the paper is structured as follows. In the next section we present our motivation: peRSSonal, a meta-portal; while in section 3 we present the architecture of our system. In section 4 we describe the algorithmic aspects of our system while the following section contains the results of the experimental evaluation of our system as well as a comparison to already existing mechanisms. The paper concludes with remarks on our system and what feature enhancements we consider intresting and perhaps useful.

2 Motivation: peRSSonal Meta-portal

The prevalent idea that motivated the peRSSonal [10] mechanism has its roots in the change of our web life, which has turned every single corner of the Web in a potential source of valuable information. However, benefit never comes without cost: locating the desired piece of information among irrelevant data has become a difficult and tedious task, even for the more experienced users, as everyone has different special needs from the medium that is called World Wide Web. Major search engines

(e.g. Google[1], Yahoo[2]) are trying to refine search. In parallel, attempts for the creation of WWW ontologies (e.g. DMOZ) look forward to resolve these issues. Our experience made us realize that among these facts lies another huge problem that is of primary concern for millions of users all over the world. As the Internet expands and acts as a form of "digital newspaper", more and more people come to realize that they are able to read and stay informed by articles in real time. This leads to a problematic situation where users have to visit a big number of news portals to read the news from the categories they are concerned. The problem is partially answered by RSS feeds and personalized micro sites. In the first case, the user does not have to browse to every single website but, as a forfeit, they must undergo data filtering due to the fact that there is no specialization of these feeds on his needs. In the second case, focus on each user's preferences can be guaranteed but he or she still has to visit each one or several of these sites in order to track down all the information on a specific subject.

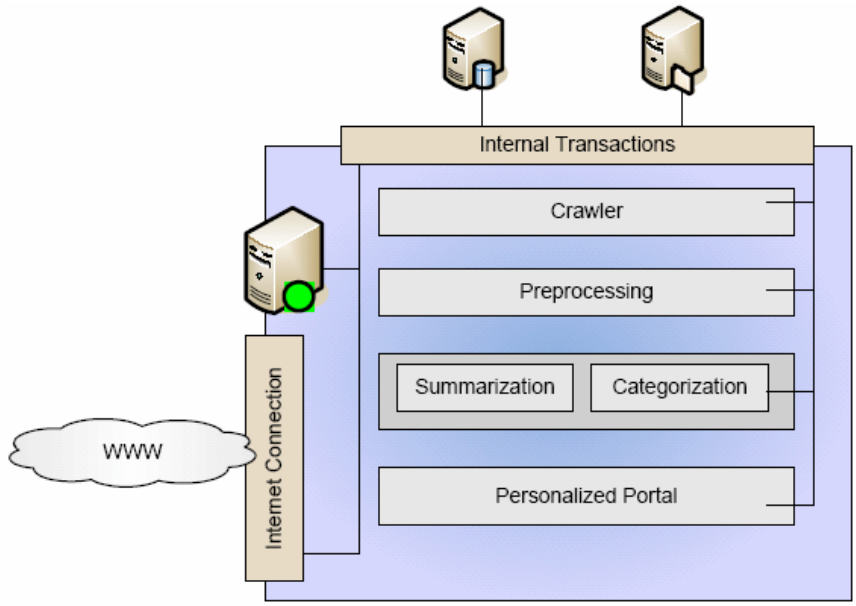

Fig. 1. peRSSonal architecture

3 Architecture and Algorithm Analysis

The architecture of the proposed system is similar to a context sensitive stemmer that applies a suffix stripping algorithm to produce the stem. It takes as input complete sentences and takes advantage of the POS tagging process in order to limit the possible suffixes that are going to be removed. The suffix stripping algorithm is rule-based and utilizes a table with possible suffixes for every part of speech tag. The output is a list of the input words, their stem and their grammatical type.

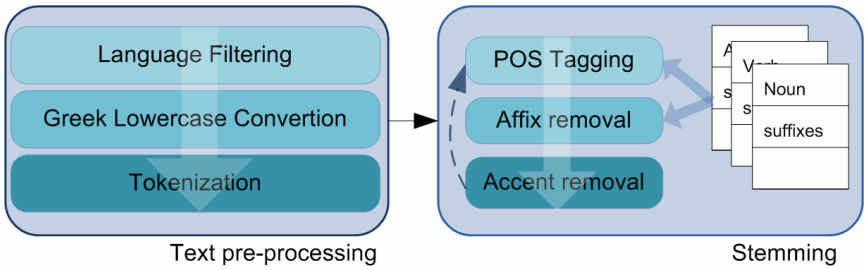

Fig. 2. G.I.C.S. architecture

The stemmer was initially designed in order to support the Greek language in the peRSSonal system. The procedure of this system is: (a) capture pages from the www and extract the useful text, (b) parse the extracted text, (c) summarize and categorize the text, and (d) present the personalized results to the end user. The capture is done using a crawler that takes as input a list of RSS feeds from news portals. The crawler extracts the articles and stores the html pages without any other element of the web page, like CSS and JavaScript files. At the next step it triggers a useful text extractor mechanism in order to be able to get the real text of the article and to omit undesired content (advertisements, etc.). The useful text can be defined as the title and the main body of the article.

The stemmer is invoked at the second analysis level, parsing the whole useful text and extracting the keywords (stems) of the article. This level receives as input XML files that include the title and body of articles. Its main scope is to apply preprocessing algorithms on this text and provide as output keywords, their location into the text and the frequency of their appearance in the text. These results are necessary in order to proceed to the third analysis level. The core of peRSSonal mechanism is located in the third analysis level, where the summarization and categorization subsystems are located. Their main scope is to characterize the article with a label (category) and produce a summary of it. All these results are then presented back to the end users of our personalized portal. The portal can feed each user only with articles that the user may find them interesting, according to his/her dynamically created profile.

The proposed context sensitive stemmer has a time and space complexity of O(n) and it can be considered as the combination of two different procedures: stemming and tagging. Although the main scope of the system is to provide stemming functionality, we incorporated a grammatical tagger in the algorithm for two reasons. First of all, viewing words in the context of the grammatical types of nearby words helps resolve some ambiguous cases where different suffixes could be removed. Secondly, because discarding words belonging to certain grammatical categories (that usually carry no significant meaning for our purposes, e.g. pronouns) produces better results than using stop-word lists or discarding words based on their size.

Firstly, the algorithm accepts an array encoded in ISO-8859-7 and discards all characters except for English and Greek letters. The remaining characters are converted to lower case and any dieresis diacritics are removed. Accenting is left intact since it provides extra information that can be used in tagging and stemming. The last step of the preprocessing is the tokenization process which creates a linked list of structures, each of which hold a word along with space for storing the tag, the stem,

and information used internally by the algorithm. After the preprocessing step, the tagging process starts which consists of two rounds.

In the first round each word is tested for fitness in different grammatical categories, using tables with known suffixes and words of the Greek language [8]. Some of the categories cause the testing process to stop if the word is found fit for them while other let it continue. In cases where the word fits in more than one category, the grammatical type of the previous word is used to determine which category is chosen. While the words are matched to categories, information about the size of the affixes is stored in the linked structure list. During the second round the same steps are followed for the words that haven't been recognized during the first round. This time the accenting is ignored as a last resort, since in many cases words are either misaccented or unaccented.

Next, the stemming procedure starts. During that, words are trimmed based on information stored in the previous step. For verbs, additional conversions are applied in an attempt to make the stems of different tenses match each other.

Finally, after the stemming process is finished. The algorithm will return the linked structure list, which now contains the stems and the tags, to the caller.

4 Experimental Evaluation

In contrast to other stemming algorithms, G.I.C.S. is not trying to extract the grammatically correct stem of words, although in most cases it does so. Instead we consider a stem correct as long as grammatically similar words are assigned to the same stem.

Specifically, for verbs we consider a stem correct if it is matching that of the first person singular of the same verb in present tense. For nouns/adjectives and pronouns we consider a stem correct if it is matching that of the masculine singular of the nominative case of the same noun. If there is no masculine singular then any singular of the nominative case is used instead. Additionally, although not necessary for considering the stemming correct, we tried to assign the noun to the same stem of the verb from which it derives (if any). For verb-derived proverbs, we consider the stem correct if it matches that of the verb which the proverb derives from. For other type of words stemming is trivial and not of any importance so we only rated the tagging.

We evaluated the algorithm using two different sets of text. The first set was composed of news articles like those that will be used by peRSSonal, which totaled in around one thousand words. The second similarly sized set was composed of both formal and informal emails. The execution time of the mechanism for these two sets was insignificant, as the average stemming speed has been estimated, through extra testing, to be eabout 163 thousand characters per second. We first evaluated G.I.C.S tagging precision on all the words in the dataset. Finally we evaluated the stemming precision of both G.I.C.S and Ntais stemmer ignoring all words except nouns, adjectives, verbs and proverbs. Words in other categories are known, though our experiments, to carry no significant meaning. Additionally these words appear frequently and their stem can be easily extracted so they affect the results positively, making the useful precision of the stemmer harder to be measured.

98.6% of the words of the first set were tagged correctly by G.I.C.S.; while 96.7% of the useful words were stemmed correctly (Fig.3) 12.5% of the erroneous stems

were the result of over-stemming, meaning that the algorithm removed more letters than it should. The rest 87.5% of the words were either under-stemmed or they were irregular and the stemmer wasn't able to convert the stem correctly to match the desired one, based on our criteria. Specifically, 75% of the errors were made when an irregular verb was stemmed. (Fig.5) By comparison, using the same words and the same criteria, the Ntais stemmer stemmed correctly 91.1% of the words of the first set (the articles).

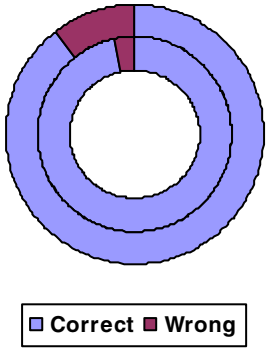

Fig. 3. G.I.C.S. and Ntais stemmer comparison. G.I.C.S. stemming precision on the inside. Ntais stemming precision on the outside. (articles dataset)

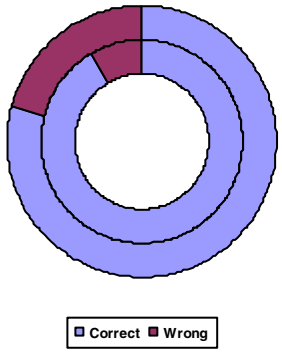

Fig. 4. G.I.C.S. and Ntais stemmer comparison. G.I.C.S. stemming precision on the inside. Ntais stemming precision on the outside.(emails dataset)

Regarding the second set which included emails, our algorithm achieved 96.7% correctly stemmed words while the tagging was successful for 91.7% of the useful words (Fig.4). Of the errors, 6.65% was because of under-stemmed words while 93.3% was of over-stemmed words including the irregular verbs. 80% of the errors were made during the stemming of irregular verbs (Fig. 5). By comparison, the Ntais stemmer achieved 88.15% of correctly stemmed words.

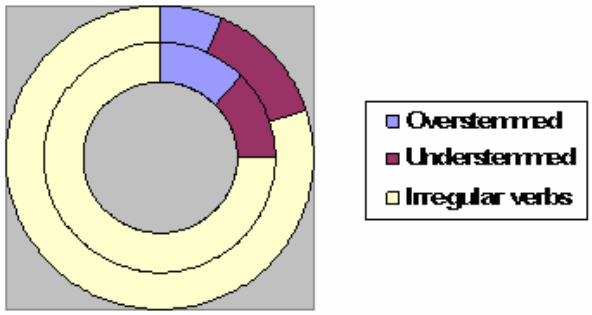

Fig. 5. G.I.C.S. types of error. Articles dataset (inside), Emails dataset (outside)

At the next experiment we utilize three different metrics [9] in order to compare the proposed stemming algorithm to Ntais implementation. The first metric (index compression factor) indicates the index reduction that can be achieved through stemming.

$$ICS = \frac{n - s}{n} \ . \tag{1}$$

Where n is the number of words in the corpus and s is the number of produced stems. The other two metric are the mean and median Hamming distance. The Hamming distance is defined as the number of characters in the two strings that are different at the same position. For unequal length strings, the difference of their length is added. The dataset for this experiment consists of 10981 words that have been retrieved from major Greek news portals. The results are shown in table 1.

Table 1. Comparison of G.I.C.S. and Ntais stemming algorithms

	Ntais stemmer	G.I.C.S.
Index compression factor	76.8%	80.9%
Mean modified Hamming Distance	1.95	2.73
Median modified Hamming Distance	2	2

5 Conclusion and Future Work

In this paper we have presented an efficient stemmer and tagger for the Greek language. The stemmer and tagger is created in order to support an existing meta-portal peRSSonal which intends to present news articles collected from the WWW to the users in a personalized manner. The stemming is performed using a rule-based algorithm that removes suffixes. It bases most of its procedures on the fact that all the words of the text are grammatically tagged by a tagger which works as part of the stemmer. The mechanism was compared to the latest known Greek stemmer implementation and the experimental evaluation showed that it can achieve higher stemming precision. Despite the fact that we just wanted to create groups of similar words as input for the peRSSonal system we managed to construct a Greek stemmer that can possibly achieve very high scores of stemming on the text that we have as input. Our input texts are news articles which are usually short in length and they use compact language with many forms of the same word used across the text.

For the future we would like to enhance the current system in order to improve the tagging process using the punctuation information. Moreover, when processing a specific word, the mechanism could take into account the tags of more words that it currently does (e.g. the tags of all the words in the current sentence). This information will lead to better POS tagging and thus, better stemming precision.

References

1. Lovins, J.B.: Development of a stemming algorithm. Mechanical Translation and Computational Linguistics 11, 22–31 (1968)
2. Porter, M.F.: An algorithm for suffix stripping. Program; automated library and information systems 14(3), 130–137 (1980)

3. Paice, D.: Another stemmer. ACM SIGIR Forum 24(3), 56–61 (1990)
4. Kalamboukis, T.Z.: Suffix stripping with modern Greek. Program 29(3), 313–321 (1995)
5. Tambouratzis, G., Carayannis, C.: Automatic corpora-based stemming in Greek. Literacy and Linguistic Computing 16, 445–466 (2001)
6. Ntais, G.: Development of a stemmer for the Greek language, MSc Thesis, Stockholm University (2006)
7. Saroukos, S.: Enhancing a Greek Language Stemmer, MSc Thesis, University of Tampere (2008)
8. Triantafillidis, M.: Modern Greek Grammar (Dimotiki) (in Greek). Reprint with corrections 1978. Institute of Modern Greek Studies, Thessaloniki (1941)
9. Frakes, W.B., Fox, C.J.: Strength and similarity of affix removal stemming algorithms. SIGIR Forum 37(1), 26–30 (2003)
10. Bouras, C., Poulopoulos, V., Tsogkas, V.: PeRSSonal's core functionality evaluation: Enhancing text labeling through personalized summaries. Data and Knowledge Engineering Journal 64(1), 330–345 (2008)
11. Papadakos, P., Vasiliadis, G., Theoharis, Y., Armenatzoglou, N., Kopidaki, S., Marketakis, Y., Daskalakis, M., Karamaroudis, K., Linardakis, G., Makrydakis, G., Papathanasiou, V., Sardis, L., Tsialiamanis, P., Troullinou, G., Vandikas, K., Velegrakis, D., Tzitzikas, Y.: The Anatomy of Mitos Web Search Engine. CoRR, Information Retrieval, abs/0803.2220. CoRR Technical Report (March 2008)