Building Stemmers for the Polish Language

Nikitas N. Karanikolas
Technological Educational Institute of Athens
Ag. Spyridonos street
Aigaleo 12243
+30-210-5385737
nnk@teiath.gr

ABSTRACT

In this paper we examine the applicability of our "supervised learning methodology that builds stemmers given minor target language knowledge" (shortly: stemmer builder) for building a stemmer for the Polish language. We examine if the existing features of our stemmer builder can support the necessities of the Polish language for building an acceptable stemmer.

CCS Concepts

• **Information systems~Data encoding and canonicalization**
• **Information systems~Dictionaries**

Keywords

Stemming Algorithms; Information Retrieval; Natural Language Processing

1. INTRODUCTION

Information Retrieval (IR) is the activity of obtaining information resources relevant to an information need (a query). The retrieval is conducted usually against a collection of information resources. Searches can be based on features of the collection. Whenever the collection of resources is a collection of processable electronic documents (documents having text) and the features used in queries are words, we talk about ***Text Retrieval***. The indexing process of the processable electronic documents is named ***Full Text Indexing***. The way that words are combined for forming queries discriminates between ***Boolean Text Retrieval*** (using Boolean operators) and ***Free Text Retrieval*** (no operators between words). ***Search Engines*** are usually Free Text Retrieval Systems that (virtually) combine a lot of processable electronic document collections.

Stemmers are modules used in various text processing tasks, including search engines, document/text summarizers, document/text classifiers, etc. Stemmers provide normalized forms of words in order to handle as one attribute (of the documents' collection) all inflected word-forms existing in the collection for the same word.

Because languages have different inflectional systems, we have to create different stemmers for each target language that the Text Retrieval system should support. Polish language is not an exception.

We have proposed a learning methodology that can build stemmers without the need of IR experts that manually define stemming rules according to the grammar and the inflectional system of the target language. Our methodology uses only basic knowledge of the target language and learns from examples (arguments of experts). Our methodology is supervised because it does not learn a stemmer from scratch but it learns the adaptation of a (two level of granularity) stemmer skeleton.

In the next sections we will discuss further about stemming and especially about approaches for stemming words of the Polish language. Then we are going to explain our supervised learning methodology that builds stemmers. We hope that our methodology is universal or at least international. Next section examines if our methodology can apply also for the Polish language. The last two sections provide the conclusions regarding the feet of our stemmer builder for the Polish language and present some ideas for future work.

PCI '16, November 10 - 12, 2016, Patras, Greece
Copyright is held by the owner/author(s).
Publication rights licensed to ACM.
ACM 978-1-4503-4789-1/16/11…$15.00
DOI: http://dx.doi.org/10.1145/3003733.3003792

2. POLISH STEMMERS

There are two main approaches for building stemmers: Rule-Based and Dictionary-Based stemmers. A study of the different Stemming algorithms that also describes the differences with Lemmatizers (that conflate a set of words in their etymological root) has been presented by Anjali Ganesh Jivani [1].

Rule-Based stemmers are algorithmic approaches, using minimal resources, that elaborate suffix replacement (or suffix removal) and conflate all the inflected forms of words (usually) in a single stem. Stems are not (almost never) well formed words but, for the mentioned applications, this is not a problem.

Dictionary-Based stemmers match every document's (or query's) word against the entries of a digitalized dictionary. The digitalized dictionary contains words in their inflected form together with the corresponding normalized form. This direct method of word normalization seems adequate for restricted domains (using small subsets of a language). However, it is inadequate to deal with unrestricted domains (using the whole repertoire of a language's words) and this is worst when dealing with heavily inflected languages.

The Polish language is a highly inflectional language. In order to highlight the highly inflectional form of the Polish language we can mention that verbs are inflected according to voice, tense, mood, gender, number and person. We can compare this with another highly inflected language. In the Greek language, verbs are inflected according to voice, tense, mood, number and person. Gender does not affect the formation of a verb in the Greek language. Consequently, there is increased interest for defining methods that conflate the inflected Polish word forms in a single stem or lemma. In accordance with the need, there is an important number of attempts for building stemmers for the Polish Language.

Błażej Kubiński's stemmer is a rule based stemmer that remove endings (suffixes) according to simple rules (defined by human experts') and in some cases removes prefixes (in some adjectives). It is implemented in python programming language and the idea is based on Porter's Algorithm. However, this stemmer does not use replacements. It is freely available through GitHub [2].

Andrzej Białecki's Stempel stemmer [3] is another rule based stemmer that separates the basic algorithm from the data that adapt the execution flow. The basic algorithm is result of the Egothor project that developed a universal stemmer [4] that is able to process any language. Data are transformation rules (patch commands) defined separately for each language/stemmer. In this case the transformation rules are not defined by human experts but they are learned from a training corpus and they are stored in data tables. Thus, the Stempel stemmer is a compilation of the Egothor universal stemmer with patterns extracted by learning from Polish corpuses.

Dawid Weiss's Lametyzator [5] is a dictionary-based stemmer that is (or used to be) available in public domain. Internally, Lametyzator uses pairs (inflected form – stem). The data (pairs) of Lametyzator comes from another project (Polish dictionary for ispell). An efficient representation of a huge number of such pairs is based in a finite state automaton. Thus, for any word (inflected form) that Lametyzator has in its database, the corresponding stem can be returned.

The article of Weiss in 2005 [6] contains details for some other trials for building freely available or open source stemmers for Polish.

3. SUPERVISED LEARNING FOR BUILDING STEMMERS

In our supervised learning methodology that builds stemmers, the development team is not experts in the target language. (Target language is the language of documents where the stemmer is going to be applied.) Thus, our approach removes the barrier of finding Information Retrieval experts with deep knowledge of the target language. Moreover, in our approach, we can create stemmers for whatever language we want.

Our approach requires two resources: a) a list of available suffixes used in the target language and b) a test set of words in the target language with their translations in the native language of the experts. These resources can be easily prepared by speakers of both languages (target language and IR experts' native language) having high school education level. It is easier to find high school level speakers of both languages (target and experts') than to find IR experts with in depth knowledge of the target language. The approach assumes a primary stemmer that provides stems by simply removing the longest suffix that match with a given word (in the target language). Consequently, experts express their arguments regarding the results of the primary stemmer. Experts' arguments are declarations of disagreement or verification declarations for the results of the primary stemmer. One feature of our methodology is a function (a utility) that measures the harmonization between a stemmer's results and some expert's (or group of experts') arguments. Thus, the IR expert can use this function (utility) to measure the harmonization between the primary stemmer and the (whole set of) experts' arguments. Details of this measure (measuring function) are given in [7].

Rule-Based stemmers elaborate a set of rules that remove (or replace) suffixes under some conditions. However, there are different levels of granularity (steps) for applying the rules. Our approach builds two granularity levels (two steps) rule-based stemmers. The expert can easily (through GUI) decide the available suffixes for each step and also configure some other options. In this way, the experts can configure different (trial) stemmers. Thus, the IR expert can use the previously described measuring function in order to measure the harmonization between some trial stemmer and the experts' arguments. In this way the IR expert can decide which trial stemmer to adopt.

However, the expert's arguments are valuable information that could be transformed to knowledge. The knowledge that can be extracted from the experts' arguments is which suffixes are adequate and which ones are inadequate for each step in order to provide a better stemmer (a stemmer that presents increased harmonization with the given experts' arguments). To achieve this goal (decide which suffixes are adequate for each step) we have constructed a heuristic process (a facility) that does this work. Details of how this facility works are given in [8]. Figure 1 depicts the interface of our stemmer builder methodology (system).

In order to create different (trial) stemmers our methodology (system) permit the user to configure through the GUI (figure 1) various parameters. User can configure:

– Which suffixes are available (can be removed) in each step.
– The value of parameter "At least remains letters". It defines the number of letters that should remain after a matching suffix removal.
– The value of parameter "Minimum word letters to apply stemming". It defines the length that a word should have in order to apply any suffix removal.
– The value (enable/disable) of option Split couples. It defines if letter couples (digraphs) can be split through suffix removal.

More than the previous configuration options, our methodology offers the previously mentioned facility (Wizard) that suggests the activation/deactivation of suffixes for each step. The wizard tries to find and activate a combination of (possibly different for each step) suffixes that maximizes the harmonization between the resulting stemmer and the selected (group of experts') arguments. See the list in the upper – right corner of figure 1 for selecting a group of experts.

4. FEET

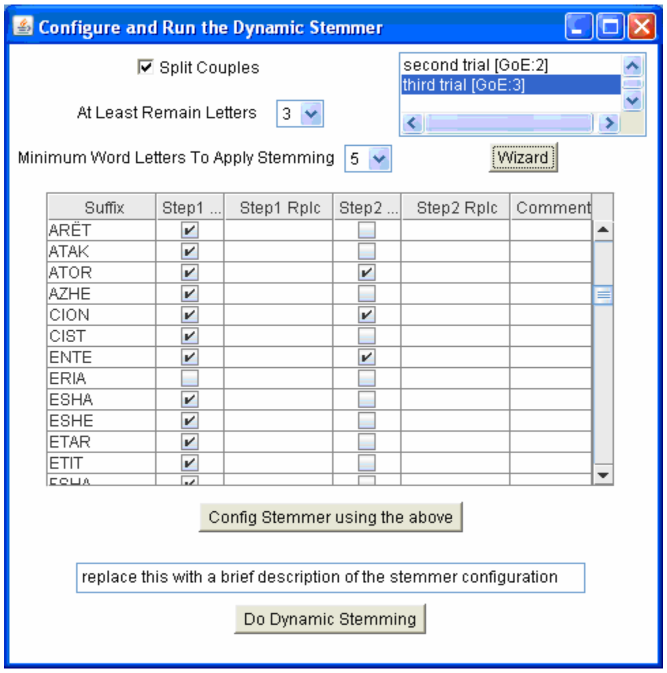
Our methodology builds two granularity-level stemmers (two steps of suffix removals). It differentiates the suffixes that are available in each granularity-level.

The longest suffix that matches is removed in each step (granularity-level).

In order to evaluate different (trial) stemmers our methodology is based on:

– A primary stemmer that has a single step that removes the longest matching suffix.
– A number of arguments that IR experts express against the results of the primary stemmer. The arguments can be negative (express disagreement of expert with the primary stemmer) or positive (express verification of expert). There are three types of arguments: CS, DS, DS/CS.

– A function that measures the harmonization between a stemmer's results and some expert's (or group of experts') arguments.


Configure and Run the Dynamic Stemmer

☑ Split Couples

At Least Remain Letters: 3

Minimum Word Letters To Apply Stemming: 5

second trial [GoE:2]
third trial [GoE:3]

Suffix	Step1 ...	Step1 Rplc	Step2 ...	Step2 Rplc	Comment
ARËT	✓				
ATAK	✓				
ATOR	✓		✓		
AZHE	✓				
CION	✓		✓		
CIST	✓				
ENTE	✓		✓		
ERIA					
ESHA	✓				
ESHE	✓				
ETAR	✓				
ETIT	✓				

replace this with a brief description of the stemmer configuration

Figure 1. The stemmer builder interface.

Polish language has 32 letters (table 1) and 7 digraphs (table 2) [9]. Each Polish digraph corresponds to a single sound and actually to a single consonant (digraph consonants). In our methodology (system) we use the term couples because it is more general and can apply for digraph vowels, digraph consonants, and diphthongs.

Table 1. Polish alphabet.

a	ą	b	c	ć	d	e	ę
f	g	h	i	j	k	l	ł
m	n	ń	o	ó	p	r	s
ś	t	u	w	y	z	ź	ż

Table 2. Polish digraphs.

ch	cz	dz	dź	dż	rz	sz

Previous work [8], for building stemmers according our methodology for some other language (Albanian), drives to the conclusion that the value False to "split couples" parameter produces better stemmers. However we kept the "split couples" as a configurable parameter because other languages can support splitting digraphs during the application of inflectional rules that produce words. For example the Polish words "koszony" (participle, passive, present perfect, male, singular, nominative) and "kosić" (verb, active, present, subjunctive, singular, second person) are inflected forms of verb barber (to cut someone's hairs). The first one contains the digraph "sz" while the second one has only the letter "s".

All features of the methodology can apply for Polish. In the beginning a list of available for the target language suffixes should be imported in our system (methodology). These suffixes can be extracted from grammar books of the target language by target language speakers having high school knowledge of language (not IR experts). An external primary single-step stemmer that simply removes the longest matching suffix can also be easily programmed. IR experts can declare their arguments against the primary stemmer's results. The Wizard can facilitate the user (IR expert) to create easily good trial stemmers. Configuration options ("At least remains letters", "Minimum word letters to apply stemming", "Split couples") can be used for fine tuning trial stemmers. The Harmonization measurement function is built in the system. Thus, this methodology can apply for building a stemmer for the Polish Language. In order to make more self contained the present article, we will provide some Arguments given by experts for the results of the primary stemmer applied in a small Polish collection of documents.

Table 3. CS argument example for Polish.

Ref. No	Word	Stem	Explanation	Argument
1118	głębiej	głęb	βαθύ	CS(głęb)
1119	głęboką	głębok	βαθιά	
1120	głębokie	głębok	βαθιά	
1121	głęboko	głębok	βαθιά	
1122	głębszym	głęb	βαθύτερο	

Table 4. DS/CS argument example for Polish.

Ref. No	Word	Stem	Explanation	Argument	
1289	Idealne	ide	τέλειο	DS	CS_1
1290	Idealnego	ide	τέλειου		
1291	Idealnych	ide	τέλειους		
1292	Ideał	ide	τελειότητα		
1293	Ideą	ide	ιδέα		CS_2
1294	Idee	ide	ιδέα		
1295	Ideę	ide	ιδέα		
1296	ideologii	ideolog	ιδεολογία		CS_3

5. CONCLUSIONS

It seems that our methodology offers the proper facilities for building stemmers for the Polish language without having advanced knowledge of the Polish language. We need only basic knowledge of the Polish language (alphabet, digraphs, a list of suffixes, few documents) and volunteers to translate some Polish words to the language that IR experts speak.

6. FUTURE WORK

Till now, the Wizard considers only the CS arguments. In the future we plan to extend the Wizard in order to also consider the DS/CS arguments.

Till now our methodology supposes that words are in the same case (either only capital or only lower letters). Regarding the ascent, the system is depended on the user input. That means that the system does not offer any facility to remove accents and considers C as different from Ć. Thus the user has to apply, before data insertion, some filter that removes accents (e.g. replace Ć with C) in order to consider accented and not accented forms of letters as equals. In some languages this (accent removal before insertion to our system) is enough. For example, in Greek language accents are used only to stress some letter and consequently can be completely removed before data insertion (in the system) and consequently before applying any stemming. However in some other languages, accents and other diacritics are not used for stressing but for changing completely the phoneme (the sound of letter). For example in Polish O sounds differently than Ó. The same applies for graphemes L and Ł, and graphemes A and Ą. But in the same language (Polish) and for some other letters the accent is used only for orthographic reasons (e.g. Ć, Ń, Ś and Ź are used at the end of words and before consonants and never before vowels). Thus, an internal to the system Alphabet Reduction should be very interesting. In such case not any accent or diacritics removals should be conducted outside the system (before data insertion). The user will be provided with internal to the system handlers, in order to selectively enable/disable the differentiation of similar graphemes (letters) and build different trial stemmers accordingly.

Another future plan we have is to build a stemmer for Polish using our system but now with a big set of words (more that 5000 words) and compare the resulting stemmer with another existing rule based stemmer for Polish.

7. REFERENCES

[1] [1] Anjali Ganesh Jivani, A Comparative Study of Stemming Algorithms. International Journal of Computer Technology Applications, Vol 2 (6), 2011.

[2] Błażej Kubiński, Polish stemmer https://github.com/Tutanchamon/pl_stemmer

[3] Andrzej Białecki, Stempel – Algorithmic Stemmer for the Polish Language, 2004. http://getopt.org/stempel/

[4] Leo Galambos, Semi-automatic stemmer evaluation. In Proceedings of the International IIS: Intelligent Information Processing and Web Mining Conference, Zakopane, Poland, 2004. Advances in Soft Computing, vol. 25, pp. 209-218. http://link.springer.com/chapter/10.1007/978-3-540-39985-8_22?no-access=true

[5] Dawid Weiss, Lametyzator – Stemming engine for Polish, 2003. http://www.cs.put.poznan.pl/dweiss/xml/projects/lametyzator/index.xml

[6] Dawid Weiss, A Survey of Freely Available Polish Stemmers and Evaluation of Their Applicability in Information Retrieval. 2nd Language and Technology Conference, Poznań, Poland, 2005, pp. 216-221. http://www.cs.put.poznan.pl/dweiss/site/publications/download/ltc_092_weiss_2.pdf

[7] Nikitas N. Karanikolas, A methodology for building simple but robust stemmers without language knowledge: Overview, data model and ranking algorithm. CompSysTech'2013: 14th International Conference on Computer Systems and Technologies, June 2013, Ruse, Bulgaria. ACM ICPS, doi:10.1145/2516775.2516783.

[8] Nikitas N. Karanikolas. Supervised learning for building stemmers. Journal of Information Science, Vol. 41 (3), pp. 315-328, 2015, doi:10.1177/0165551515572528.

[9] Grzegorz Jagodziński, A Grammar of the Polish Language. http://grzegorj.interiowo.pl/gram/en/gram00.html