

A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine

Ilya Segalovich

Yandex

iseg@yandex-team.ru

Abstract

This paper describes a {simple yet practical} algorithm of morphological analysis and synthesis that uses a limited dictionary to obtain {rather precise} morphology of a wide lexical coverage. It doesn't imply any particular morphological model of the dictionary and is implemented for Russian, Polish and English. The design of the algorithm fits the constraints of a modern web search engines.

Preamble: does a web search engine need morphology?

Some IR studies showed very little or no improvements in search effectiveness with morphological analysis applied [harman1991]. Though later results proved success even in languages that typically have few word forms such as English [hull1996] it is still under the question if a web search engine needs morphologic analysis at all. Google, the #1 ranked search engine in the world, claims that it intentionally doesn't use stemming to provide the most accurate results to its users [google1998]. Such an extreme position was a good revisionist point in the late 90s before the link analysis had improved the precision of web search. But is recall really unnecessary? Here are some arguments pro. Number of pages that a web search engines return are distributed according to a Zipf law, which means that though on popular queries it is very big, the amount of queries with only few found documents is substantial. For example more than 30% of responses received at www.yandex.ru on the 06.09.2001 contained less than 100

documents while only 5-10% of queries have misspelling.

The user behaviour, coordinate ranking and morphology

Another observation in favour of morphological processing on the web is the searchers behaviour and expectations. Users have their own mental models of how search engines should work [muramatsu2001] and they generally prefer coordinate-level ranking [heimstra2002]. To satisfy this model (please note that we don't mention precision and recall as the primary goals) Web search engines first look for the exact phrase matching. When it is impossible (more 50% of queries doesn't have a "phrase match" anywhere on the web) search engines rank results according to the term proximity [muramatsu2001] so the responses are as "similar" to queries as possible. One may notice looking at the typical web search engine response that documents are roughly listed in the decreasing order of the edit distance ("likelihood") between the query and *snippets* of the documents. In the same time 75% of web search queries consist from more than 1 word [onestat2003]. So why the "similarity" of responses to longer queries allows insertion of big spans of characters (words, sentences) between search terms but doesn't allow small changes of terms' suffixes when terms are close to each other?

Introduction & current trends

The state-of-the-art algorithms for morphological analysis are based on the Finite State Transducers of different flavours

([kimo1983], [kartunen1992], [mohri2000], [daciuk2000]), where phonological rules are described by hand and each word from the dictionary is assigned its corresponding (inflectional/derivational) model. These transducers allow both analysis and synthesis of the word forms; they are very fast and robust. Unfortunately number of unknown lexical items brought from the web with every crawl are very big and can be estimated by Heaps law $V_R = Kn^\beta$, where V_R is the portion of the vocabulary represented by the instance text of size n . K and β are free parameters determined empirically. On the web β is closer to 1.0 than to 0.5 due to (un) intentional misspellings, foreign words, privately invented words such as taken from *blogs* etc. This means that the dictionary grows sub linear with grow of index. One approach here is simply to ignore all the words that are not from main dictionary [www.aport.ru] hoping that the coverage will suffice.

Recently the problem of the morphological analysis of the unknown words draws more attention mostly from the success in unsupervised learning of morphology from the corpora itself. [Jackuemin1997] analysed statistics of *k-n conflatons* (where k – length of the hypothesized suffix, usually 3 letters, and n – the width of the context window – usually 2 words) and obtained decent results compared to Lovins and Porter famous hand-made stemmers ([lovins1968], [porter1980]). [gaussier1999] used similar approach (*p-similarity* – the number of common letters between two words) to learn derivational morphological families and even derivational trees. The more generalized approaches of corpus suffix analysis are presented in [goldsmith2001] and [snover2002]. They use MDL (Minimum Description Length) principle, known from theory of information, to produce the set of stems, signatures (paradigms) and affixes that can be encoded in the minimal number of bits [goldsmith2001] or that are most probable (Brent) thus making morphological description of corpus minimal in some sense. Linguistica [goldsmith2001] is also a publicly available tool that we will use later in this paper.

Another promising approach [schone2001] makes use of word contexts to distillate hypothesized models obtained from pure suffixal analysis. It uses the primitive of *PPMV* – *pair of potential morphological variants* and measures *semantic similarity* between them, which is understood as the normalized correlation between their vectors in the latent semantic space (which in fact is word-to-word-neighbours matrix factorised with the help of LSI).

All of the approaches mentioned above are unsupervised in a sense that they don't use any prior knowledge about a language and its morphological models. Unfortunately there are at least two obstacles that don't let direct use of the unsupervised methods in Web search engines as is. The first is the quality of the morphology of the most used words that must be as precise as possible. The second is that the complicating of index updates: index is distributed over hundred and thousands of computers, so the morphological interpretation of the new words that appear in daily crawls must not change too often or it may require daily patching of the index. It worth noting that if a language has a web corpus substantial enough to be of a public interest and of a commercial search engine to strive on, it usually has plenty of dictionaries that already accumulated some knowledge. For example widespread spelling tools that use compact description of all possible word forms such as [ispell] or [aspell] have dictionaries for 37 and 26 languages respectively. Of course the task of the compact listing of all allowed forms in a language is not exactly the task of describing all words and its respecting paradigms, nevertheless these dictionaries are successfully used in the web search engines ([mnogosearch], [glimpse], [aspseek] etc). Important observation about existing dictionaries that they always (and hopefully correct) describe the most frequently used words of the language. Here we suppose that all the complex and non-trivial [inflectional/derivational] models always belong to the most used words of the language -- which is in fact another law of Zipf -- "the greater the

frequency of a word, the greater the number of morphologically complex forms it will be used in” [manning1999]).

The idea of applying a relatively small dictionary to guess new words morphology is not new. For example [woods1999] used a set of aggressive rules together with a small lexicon to improve lexical coverage, [mikheev1995] presented a technique of automatic acquisition of unknown words morphology through a general-purpose lexicon and word frequencies, the classic Russian morphology algorithm [belonogov1985] used the same dictionary both for known and unknown words, also all the FST machines allow to write catch-all rules for the words that are not in the dictionary. While the process of describing particular word morphology seem to be deterministic, the catch-all rules are rather hard to write and debug. The example of applying a one-time learning procedure together with the existing morphological dictionary to the corpus in order to obtain morphological guesser is presented in [kovalenko2002].

The approach described here is based on [segalovich1998] and uses a very simple idea of morphological similarity. Unknown word morphology is taken from all the closest words from dictionary, where the closeness is the number of letters on its end. From the [belonogov1985] approach it differs in that it doesn't choose one variant, but rather take all possible variants that have the same common length, and unlike [kovalenko2002] it uses for the induction of the hypothesizes the whole dictionary not just N several last letters of stems. To build such dictionary we don't need to know the morpheme structure or any particular formalism of any particular morphology, the compiler accepts input in the form of full listing of all word paradigms from the dictionary. If the grammar tags are available they are accepted either. Such a dictionary is be able not only to analyse known words, but also to synthesize all forms both known and hypothesized. The algorithmic idea behind this algorithm is that after we analysed the word trying to find its exact form in the dictionary we already have

enough information (we already looked at all its letters) to find the most probable inexact equivalent. The FST approach also use one pre-compiled automata with exact and catch-all rules but unlike [Karttunen1992] we don't require catch-all rules, we rather consider them harmful and completely rely on the probabilistic nature of the unknown words analysis. This is the idea of the algorithm – use deterministic knowledge when possible (for all the ‘known’ words) and probabilistic approach when the exact interpretation is not available.

An interesting application of this algorithm is the combined dictionary of several languages which is useful in the situations when the language is ambiguous, so the combined dictionary will choose the most likely morphological interpretations based on the likelihood to all words of all languages. (Such a combined Polish-English dictionary is used in our experimental project at www.yandex.pl as a part of the query interpretation).

Design issues

There are some questions about morphological model and algorithms that must be answered before we describe the algorithm itself. All of them originate from the nature of the information retrieval or from the technical requirement of search engines. What kind of morphological analysis to perform: inflectional or derivational? Are grammar tags (or POS tags) required as a result of analysis? Do we need stems (roots) or the base form (in other words -- do we need stemming or lemmatisation)? How unknown words are processed? Shall we use the context of words to help analysis? Is the analysis corpus-dependent and if it is, to what degree? To fully answer to all of these questions we will need a much longer paper, in short the answers are: inflectional, no grammar tags, lemmatisation, no context built in morphology. Here is very a short reasoning.

Inflectional morphology. For the sake of precision we don't want derivational morphology, though the difference is often very subtle; consider for example *aspect, gender of nouns, diminutive forms*.

No grammar tags. Different languages have different morphology with different grammar systems and search engine often can't distinguish nor the language of the queries (people never choose appropriate switches) neither of the document (consider multilingual document, technical documents, etc). So the index keys should have purely textual representation without any grammar or language tags.

Lemmas not stem. Stemming is not only connects different parts of speech or derivational forms (that we don't want) but also produce spurious confluents. So search engines usually need the dictionary forms as a result of morphological analysis. That in turn opens the question of disambiguation between them even if we completely drop POS tags; consider *stocking* -> (*stock OR stocking*).

Context analysis. The morphological algorithm itself must not rely on the word context or on the assumption that all the words will be submitted strictly in the order of the text flow. The indexing speed requirements of modern search engine may not permit this (their speed is about 50-100 documents per second [najork2001]), the morphological module is often called in multi-threaded environment once per word form met anywhere in the portion of texts.

Ambiguity. It's important to note that ambiguity won't cost more than 10-20% of the index size. And the linguistic disambiguation is impossible in short queries (in fact non-linguistic disambiguation is much more successful by coordination ranking).

Algorithm

The dictionary is represented as a set of *tries* [cormen1990] of inverted stems and a *trie* of all possible suffixes. The boundary between stem and suffix is taken either from the input file if it is explicitly presented or computed automatically as the length of the common part of all word forms in the paradigm. Here is the description of the algorithm.

1. Sink into word from the right end using the *trie* of all possible suffixes. As a result we

have all possible positions of stem/suffix boundary in a single pass

2. Start with the deepest possible stem/suffix boundary and perform the steps 3-7

3. Use two last letters of a possible stem as an index to obtain an appropriate stem *trie*. If no stem has last two letters like this go to the next stem/suffix boundary

4. Sink into the appropriate stem trie and try to find the terminal leaf at the edge of the word collecting all the branching points in a single pass

5. *Dictionary word.* If there is a terminal leaf at the edge of the word compare the corresponding inflectional model (its identifier is stored in the stem trie as a special letter after the first letter of a stem) with the word suffix. If it matches it means we found the dictionary word

6. *Unknown word.* If there is no terminal at the edge of the word or the inflectional model doesn't match to the suffix then perform the next steps

7. Traverse all the previously collected branching point of the stem trie starting from the deepest one and find all the candidate stems which can be models for the given unknown word. Check their inflectional models against the given suffix

The algorithm supports 2 additional modes: check only dictionary words; find only one candidate (this is useful to conform one-to-one stemming).

Speed up. We cut traversing as early as we can with the use of 'best candidate' metrics i.e. the number of common letters at end of the word between the given word and a model candidate. All the branching points in a trie that have only one inflectional model below are marked, so it helps to limit traversing.

Learning. Some heuristics that was learned and now built into algorithm are: the resulted stem must contain a vowel, model stems must have a productive POS tag (when available), there must be minimum stem length, there must be minimal common part of an example and unknown word, the model paradigm must have

minimal frequency (e.g. met at least twice in the dictionary).

It's also quite useful [segalovich1998] to use the corpus statistics to do some basic disambiguation and eliminating of extra morphological variants: remove lemmas with paradigms fully 'nested' into paradigms of the other lemmas (if we met *shmocking*->(*shmock* | *shmocking*), *shmockings*-> (*shmocking*), but not *shmock* or *shmoked* we can drop the lemma *shmock*; or as an opposite if we met *shmocking*, *shmock* and *shmoked* but not *shmockings* we can drop the lemma *shmocking*), prefer paradigms with more frequent lemmas, prefer paradigms that contain lemma in the corpus, etc. These and other heuristics may be obtained through the learning by corpus. Anyway, in this paper we concentrate on the quality of the morphology itself, and don't consider disambiguation phase as a part of it.

Experiment

Though our dictionary is implemented for several languages (theoretically for all that have full paradigm dictionaries, in practice – only Russian, Polish, and English) we have chosen to measure its quality for Russian. First of all there are very few comparisons available for Russian morphology, especially for the guessers. In the last years at least 3 new tools appeared that cope with Russian morphology. The new version of Porter stemmer [snowball] uses hand written set of rules in a special 'snowball' language and is available for Russian. The Linguistica [goldsmith2001] learns morphology from the corpus. The 'stemka' [kovalenko2002] is a Russian and Ukrainian morphological guesser that was built with the help of the full-fledged morphological module based on [Zaliznak1980] and the collected corpus statistics. Because it produces several variants of stems for each word we consider two extreme variants it produces: the deepest or the *aggressive*, and the shallowest or the *conservative*. The 'mystem' is the algorithm presented here (it was originally designed in 1996 as a part of Yandex search engine development), it uses the full dictionary

[Zaliznak1980] as a guessing machine in a sense described above.

Another part of the experiment is the corpus with the canonical mappings between word forms. For this purpose we took recently appeared www.ruscorpora.ru [ruscorpora2003] that is currently 1302329-words morphologically annotated corpus of Russian. Like CELEX or Brown corpus it contains lemmas, POS and other grammar tags for each word etc. Currently it contains 130823 different words {and consists from two parts: from which the second part (834255 words) is obtained with the help of the independent morphosyntactic tool, while the smaller one (468074 words) was pre-processed with mystem, so we didn't consider it in the following study}.

It's a very vague subject how to compare the quality of morphological processors for IR purposes. The obvious measures are overstemming and understemming indexes (UI and OI) that should be obtained as a result of such experiment [Paice1996]. The traditional approaches apply different morphology to the same search engine and try to estimate the gain or loss on the precision and recall. It is hardly relevant mostly because of no-sense meaning of the precision and recall on the web and of the different way of how search engines incorporate morphology. Another approach is to compare the mappings (PPMVs [Shone2000]) to some canonical mappings for example hand-coded in the tagged corpus. This approach heavily depends on the grammatical model chosen by corpus editors – what they consider derivation and what they consider inflection for example (in Russian the common misjudgements between linguists are participles vs. verbs, different aspects, adverb vs. adjective, passive voice etc). The ideal measure would be the semantic similarity between variants of PPMV. The semantic similarity is often seen as a contextual interchangeability [manning1999] and as such was used in [Shone2000] to distillate spurious PPMVs. Unfortunately in richly inflected languages the context itself is syntactically dependent on the studied form so it's impossible to use the context words as is

without morphological processing. Another issue for this approach is that the corpus however big it is does not contain enough statistical information to expose the semantics scalable to the Web.

We propose here the following approach to measure the semantic similarity between variants of PPMV. If two different forms are close enough they must be close in terms of search results of a search engine (or even “The Search Engine” i.e. Google which is a good universal measure also because it doesn’t use morphological processing). Modern search engines make heavy use from word statistics; generally they try to catch semantics as close as possible also through socio-metrics methods (e.g. link analysis) and all other possible sources of information. Also it seems a good help for a search engine to discover that the other forms of the submitted word produce surprisingly close results. Let us see for example how it can be applied to English. Say we have two stemmers: one collates *stockings* to *stock*, while other collates it to *stocking* and *stocks* to *stock*. So we take the first N (30) results from the Google listing and see that *stock* and *stocks* have a lot of common hosts (for example www.nasdaq.com), so do the *stocking/stockings* PPMV (for example www.stockingstore.com). The first PPMV (*stockings/stock*) doesn’t have any common hosts of course. So let us call a semantic similarity of two different requests a number of hosts that are common in the first 30 items of both listings. But what requests we should use? Ideally we should have found out the most popular (say 3) requests containing the particular word form and compare them to the same requests with the PPMV variant instead. In this case we would see a realistic degree of similarity and would also catch more exactly what people mean by that or this word form (its “meaning”). Here for simplicity reasons we made only the single word requests.

We applied all available stemmers to www.ruscorpora.ru corpus and compared the obtained PPMVs from each stemmer to the canonical PPMVs from the corpus. The PPMV in our sense are all pairs that collate to the same

textual representation, a lemma or a stem. To lessen the sample set we left only word forms that are met in the corpus with frequencies from 10 to 100, which is close to best term specificity (10-100ppm).

Module	Total PPMVs	PPMVs with 10-100 freq.
Canonical	484462	70028
Stemka (aggressive)	628154	82895
Stemka (conservative)	108248	18014
Mystem	718216	93863
Snowball	519262	72568
Linguistica	575960	75515

Then we removed all the canonical PPMVs, i.e. those, that were taken directly from the www.ruscorpora.ru mappings. Thus we received separate PPMV listing both for added PPMVs and for deleted PPMVs for each stemmer. The idea here is to measure maximum effect that stemming has on the search engine, both positive and negative. In fact we see the corpus as a source of the new words and PPMVs (such as a daily crawl) and use the *semantic similarity* on whole index to find out how bad the PPMVs were that the particular stemmer added and how good those PPMVs were that the particular stemmer lost comparing to canonical PPMVs. So we took 30 most frequent forms and their PPMVs from each listing of added and lost PPMVs for each stemmer. Frequency was taken according to www.yandex.ru search log (51387 words of the July 2002 that have frequency more than 100 per a month).

Module	Added PPMV	Hosts	Lost PPMV	Hosts
stemka conserv.	136	46	1384	997
stemka aggres.	787	210	493	269
snowball	643	219	487	308
linguistica	953	175	648	585
mystem	778	403	41	7

Notes on Speed

All the observed algorithms are written in C++ without use of virtual functions and templates. For the 1600 MHz Pentium IV we measured speed on the 1 million of unique word forms. The results are presented in the following table:

Algorithm	Thousands words per second
Snowball	180-190
Mystem	60-65
Stemka	380-390

Discussion

We tried to show here that the use of the whole available dictionary as a guessing rule outperforms both purely corpus inducted morphology and hand-written set of guessing rules. Though the number of extra PPMVs is bigger than in other algorithms, they are much more precise and rarely bring wrong associations. Also the number of lost associations is minimal. The purely corpus inducted morphology (Linguistica) showed the worst performance in terms of extra PPMVs and low number of its semantic associations. Deepest variant of stemka and snowball showed very similar results, despite the fact, that snowball is a hand-made set of rules and stemka is the learned by corpus suffix-stripping algorithm.

References

- [1] [aspell], Aspell, <http://aspell.sourceforge.net/>, <http://ftp.gnu.org/gnu/aspell/dict/>
- [2] [belonogov1985]

- [3] [bosch1999], Memory-Based Morphological Analysis, A. v. d. Bosch, W. Daelemans, ACL'99, 1999
- [4] [cormen1990], Introduction to algorithms, Cormen, TH, Leiserson, CE, Rivest, RL, London: MIT Press, 1990
- [5] [daciuk2000], Incremental Construction of Minimal Acyclic Finite-State Automata, J. Daciuk; B. W. Watson; S. Mihov; R.E. Watson Computational Linguistics, Volume 26, Number 1, March 2000
- [6] [furnas1988], Information retrieval using a singular value decomposition model of latent semantic structure, G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, K. E. Lochbaum, 11th ACM SIGIR, 1988, Grenoble, France
- [7] [gaussier1999], Unsupervised learning of derivational morphology from inflectional lexicons, E. Gaussier, ACL'99 Workshop: Unsupervised Learning in Natural Language Processing, 1999
- [8] [goldsmith2001], Unsupervised learning of the morphology of a natural language, J. Goldsmith, Comp. Ling. 2001
- [9] [Google1998], Google Help: The Basics of Google Search, Google, <http://www.google.com/help/basics.html>
- [10] [gubin2002], Design of a Morphological Dictionary for a Search Engine, M. Gubin, Personal page, December 2002
- [11] [harman1991] How effective is suffixing? D. Harman, JASIS vol. 42, 1991
- [12] [heimstra2002], Term-Specific Smoothing for the Language Modelling Approach to Information Retrieval: The Importance of a Query Term, D. Hiemstra, 25th SIGIR, 2002
- [13] [Hull1996], Stemming Algorithms A Case Study for Detailed Evaluation, D. A. Hull (Rank Xerox Research Centre), JASIS vol. 47, 1996
- [14] [ispell], Ispell Dictionaries, <http://fmg-www.cs.ucla.edu/fmg-members/geoff/ispell-dictionaries.html>
- [15] [jacquemin1997] Guessing morphology from terms and corpora, C. Jacquemin, SIGIR'97

- [16] [karttunen1992], Two-Level Rule Compiler, L. Karttunen, and K. R. Beesley, Xerox Palo Alto Research Centre, Palo Alto, 1992
- [17] [koskonniemi1983], Two-level Morphology: A General Computational Model for Word-Form Recognition and Production, K. Koskeniemi, Publication No. 11, University of Helsinki.
- [18] [lovins1968] Development of a Stemming Algorithm, Lovins, J.B., .Mechanical Translation and Computational Linguistics, Vol. 11
- [19] [manning1999] Foundations of Statistical Natural Language Processing, C.D. Manning, H. Schutze, 1999
- [20] [manber1989], Suffix arrays: A new method for on-line string searches, U. Manber, G. Myers, Proceedings of the First Annual ACM-SIAM, May 1989
- [21] [mikheev1997], Automatic Rule Induction for Unknown Word Guessing, A. Mikheev 1997 University of Edinburgh
- [22] [mohri2000], The Design Principles of a Weighted Finite-State Transducer Library, M. Mohri, Fernando Pereira, Michael Riley, Theoretical Computer Science, 2000
- [23] [Muramatsu2001], Transparent Queries: investigation users' mental models of search engines, J. Muramatsu, Wanda Pratt, 24th ACM SIGIR, 2001
- [24] [najork2001], High-Performance Web Crawling, M. Najork, A. Heydon, Compaq System Research Center Report, 2001
- [25] [onestat2003], Onestat.com News Release, <http://www.onestat.com>, <http://www.pressi.com/int/release/65369.html>
- [26] [Paice1996], Method for evaluation of stemming algorithms based on error counting, C. D. Paice, JASIS, Vol.47, Issue 8, August 1996
- [27] [porter1980], An algorithm for suffix stripping, M. F. Porter, Program, 14:130—137, July 1980
- [28] [segalovich98], Russian Morphological Analysis and Synthesis With Automatic Generation of Inflection Models For Unknown Words, I. Segalovich, M. Maslov, Dialog'98 (in Russian) <http://company.yandex.ru/articles/article1.html>
- [29] [Shone2000], Knowledge-Free Induction of Morphology Using Latent Semantic Analysis, P. Schone, D. Jurafsky, 4th Conference on Computational Natural Language Learning and 2nd Learning Language in Logic Workshop, Lisbon, 2000
- [30] [snover2002], A Probabilistic Model for Learning Concatenative Morphology, M. G. Snover, M. R. Brent Proceedings of NIPS-2002
- [31] [snowball] Snowball, <http://snowball.tartarus.org/>
- [32] [woods1999], Aggressive Morphology for Robust Lexical Coverage, W.A. Woods, Sun Technical Report http://research.sun.com/techrep/1999/smli_tr-99-82.pdf
- [33] [zaliznak1980]. Grammatical Dictionary of the Russian Language. A.A. Zalizniak, Published 1980
- [34] [kovalenko2002] <http://linguist.nm.ru/stemka/stemka.html>
- [35] [ruscorpora2003] Russian Morphological Corpus <http://corpora.narod.ru/article.html>