

# Text Mining: Dealing with Multiple Orthographies

Anton Bryl  
University of Trento, Italy;  
Create-Net, Italy  
abryl@create-net.org

## ABSTRACT

For many languages the problem of plurality of orthographies exists in this or that form. Apart from the wider discussed problems of differences between modern and historic orthographies, or of the absence of a standard orthography in some languages, there are also languages for which several standards of orthography coexist at the same time and in the same territory for some reason. Evidently, it causes problems for the applications working with text data, in particular for web search engines: in order to find all the relevant information the user needs to repeat his query in all possible spellings. In this article we discuss the problem statement and propose a partial solution, discussing stemming-like orthographic normalizer, that frees a word, at least partly, from orthography-specific features and is able (though primitively) to adjust the balance between false positives and false negatives, which is required to meet the needs of practical application. A sample normalizer for the Belarusian language is presented and experimentally evaluated. The experiments show that the solution is quite efficient, though there is room for improvement.

## Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications—*Text Mining*; H.3.0 [Information Storage and Retrieval]: General—*Web Search*; H.3.1 [Information Storage and Retrieval]: Context Analysis and Indexing—*Linguistic Processing*

## 1. INTRODUCTION

The problem of plurality of orthographies, stated and studied in Chinese, Japanese, and Korean [2], is also present in this or that form in many languages with alphabet-based writing systems. It may take the form of minor differences of local spelling in different countries (for example British and American English), or differences between old and new orthographies after a reform, with the old orthography still used sometimes for old texts (for example in Russian after

the reform of 1918). Moreover, even within one orthography there may exist multiple possible spellings for certain words, especially for proper names of foreign origin. Finally, for some languages two or more quite different standards of spelling coexist in the same time and on the same territory (for example for Belarusian or Cornish), admirers of each unwilling to use the other variants because of various reasons. This last case is the main focus of our work. Such plurality of orthographies should be considered carefully in text mining applications, in particular in search engines, unless we agree to find in many cases only a quite random subset of the relevant data.

It is possible to treat different spellings of one word as a particular kind of synonyms or related words [1]. Thus a valid solution for the problem is listing different spellings of each words in a thesaurus. This approach is used to deal with unsystematic spelling in old texts [3]. However, in the case of several systematic orthographies this approach has a disadvantage of being seriously redundant: alternative spellings for each word have to be entered separately (which is a long and hard work), while the differences between several formalized orthographies often allow convenient generalization. Another problem is that no thesaurus can contain all possible words, at least because of enormous number of proper names that can potentially be used in texts. Also the problem of multiple orthographies is similar to the problems of intentional misspellings in junk e-mail [4] and inconsistently spelled proper names in automatic speech recognizer output [6], but there is at least one serious difference: in both these problems there can be a huge number of possible spellings of one word, instead of two or three in our case. Furthermore, in the case of intentional misspellings there is a need for considering possible reactivity, that is, attempts of spammers to overcome new techniques, while in the case of multiple orthographies no such phenomenon is present.

In this paper we discuss a simple algorithmic approach to the problem. The discussed algorithm, which can be called a *substring-replacement orthographic normalizer*, is based on an idea similar to that of affix-removal stemming [5], namely to use a transformation which brings words to some uniform spelling. This algorithm is able to adjust the balance between false positives and false negatives, which is needed for practical tasks, as it will be shown further. We also present a sample normalizer for the Belarusian language together with basic experimental evaluation.

The rest of the paper is organized as follows: in Section 2 we give a formal problem statement, in Section 3 we describe the proposed algorithmic solution to the problem, in Section

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission from the author.

DIR 2007, March 28-29, Leuven, Belgium  
Copyright 2007 author .

4 we present an example of a normalization algorithm for the Belarusian language, in Section 5 some experimental evaluation of this example is presented and discussed, Section 6 is a conclusion, and Section 7 contains acknowledgements.

## 2. PROBLEM STATEMENT

### 2.1 The Problem: Plurality of Spellings

Informally the problem stands as follows: we need a way to determine, given two sequences of letters, if they can be two different spellings of the same word. A more formal problem statement is provided below.

Let  $W$  be the set of all potentially possible words of the language, including personal names, in all possible grammatical forms. Let  $A$  be an alphabet, and  $A^+$  be the set of all finite-length sequences of characters from this alphabet. Then an orthography which uses this alphabet can be defined as a function that relates each word of the language to a set of its possible spellings (such set usually, though not necessarily, contains exactly one element):

$$\mathfrak{D} : W \rightarrow 2^{A^+},$$

Say, we want to consider  $N$  orthographies in our application,  $\mathfrak{D}_1, \mathfrak{D}_2, \dots, \mathfrak{D}_N$ , which use alphabets  $A_1, A_2, \dots, A_N$  (probably, though not necessarily, identical). Then we can define a set  $\mathfrak{W}$  of all possible written words:

$$\mathfrak{W} = \{\mathfrak{w} | \exists i \in \{1..N\}, \exists w \in W, \mathfrak{w} \in \mathfrak{D}_i(w)\},$$

$$\mathfrak{W} \subset A_1^+ \cup A_2^+ \cup \dots \cup A_N^+$$

Now we can formally state the problem in the following way: we need to develop an algorithm able to determine, if two written words are in the following relation:

$$S \subset \mathfrak{W}^2,$$

$$(\mathfrak{w}_1, \mathfrak{w}_2) \in S \Leftrightarrow \exists i, j \in \{1..N\}, \exists w \in W,$$

$$\mathfrak{w}_1 \in \mathfrak{D}_i(w), \mathfrak{w}_2 \in \mathfrak{D}_j(w).$$

This relation has the following properties:

- it is reflexive:

$$(\mathfrak{w}, \mathfrak{w}) \in S, \forall \mathfrak{w} \in \mathfrak{W};$$

- it is symmetric:

$$(\mathfrak{w}_1, \mathfrak{w}_2) \in S \Leftrightarrow (\mathfrak{w}_2, \mathfrak{w}_1) \in S, \forall \mathfrak{w}_1, \mathfrak{w}_2 \in \mathfrak{W};$$

- it is not necessarily transitive, in particular due to the fact that the words that are homographs (that is, are spelled in the same way) in one orthography are not always homographs in another orthography.

|                |  |
|----------------|--|
| <b>Example</b> | <p><i>Language:</i> Cornish.<br/> <i>Orthographies:</i> Kernewek Kemmyn &amp; Unified Cornish.</p> |
|----------------|--|

The words for “care” and “work” are spelled as “gwith” and “gweyth” in Kernewek Kemmyn, while in Unified Cornish both are spelled as “gwyth”. Therefore we have:

$$(gwith, gwyth) \in S,$$

$$(gwyth, gweyth) \in S,$$

but

$$(gwith, gweyth) \notin S$$

Hence, in this case the relation  $S$  is intransitive by definition.

Surely, it does not mean that the relation is necessarily intransitive for all languages.

In practice, unless the differences between the orthographies are very simple and completely formal, a real-world algorithm will implement not the relation  $S$  itself, but another relation, which we will further designate with  $S'$ , which is an approximation of  $S$ . Then  $S_{fp} = S' \setminus S$  is the set of false positives (that is,  $(\mathfrak{w}_1, \mathfrak{w}_2) \in S_{fp}$  if there exists no such word  $w$  that both  $\mathfrak{w}_1$  and  $\mathfrak{w}_2$  are possible spellings of it, but the algorithm decides that there exists such a word); and  $S_{fn} = S \setminus S'$  is the set of false negatives (that is,  $(\mathfrak{w}_1, \mathfrak{w}_2) \in S_{fn}$  if  $\mathfrak{w}_1$  and  $\mathfrak{w}_2$  are different spellings of one word, but the algorithm fails to recognize it).

### 2.2 Peculiarities of Problem Statement for Different Practical Tasks

The notion of what is the optimal balance between false positive rate and false negative rate may differ seriously depending on the application. Below we will discuss two kinds of applications, namely search engines and wiki engines.

In search engines there are at least two possible ways of dealing with error rates. Firstly, one can make the desired balance between false positive rate and false negative rate a parameter of search, with a reasonable default value and possibility to adjust it manually. Secondly, one can use the minimal false positive rate which must be allowed to find the given document to judge its relevance, so that the documents found with higher false positive rate allowed appear further in the list of results than those found with lower false positive rate allowed.

Another task where the problem of multiple spellings causes inconvenience is finding the article with a given name in wiki engines. Here the problem of multiple spellings is usually solved by explicit creation of redirection pages, which can be viewed as a kind of thesaurus-based approach. False positives are completely unacceptable here, while false negatives can be tolerated: they can be eliminated by the use of redirection pages as before. Automatic consideration of plurality of orthographies is not only more convenient than manual creation of redirections, but also more reliable, because the creation of redirections is easy to forget about. If the article with a given name is not found in the database, there may be two reasons for it: either it really does not exist, or the author forgot to create a redirection, and the algorithm produced a false negative. In such cases it makes sense to propose to the user a list of the articles the names of which are *possibly* the same as the one he is searching for. For this task, it is reasonable to tolerate high false positive rate, taking care rather of minimizing false negative rate.

To sum up, we may say that the fully functional solution to the problem must provide the possibility to adjust the balance between false positives and false negatives, allowing to make any of this very low with the other still not too high.

### 3. POSSIBLE SOLUTION: RULE-BASED SUBSTRING REPLACEMENT

#### 3.1 Core Idea

A simple solution can be obtained if we have a non-thesaurus based algorithm that performs orthographic normalization, that is, brings all words to a kind of a “unified orthography”, not necessarily identical to one of the existing. Formally, we define a function

$$N : \mathfrak{W} \rightarrow X,$$

where  $\mathfrak{W}$  is the set of all possible written words, and  $X$  is any nonempty set. Then we define  $S'$  in the following way:

$$(\mathfrak{w}_1, \mathfrak{w}_2) \in S' \text{ iff } N(\mathfrak{w}_1) = N(\mathfrak{w}_2)$$

Here we must mention, that the relation  $S'$  defined in this way will always be transitive, and so in case of intransitive relation  $S$  such approximation cannot be perfectly accurate. Nevertheless, if the cases of intransitivity are not very frequent for a given language, this solution can be acceptable. Particular cases, if not too numerous, can be processed separately by means of a thesaurus.

The normalization algorithm which we propose in this paper rests on the same principle as affix-removal stemming [5], namely on removing and/or replacing certain character sequences under certain (sometimes quite complex) conditions in order to bring words to a uniform spelling. Obviously, such normalizers can cause both false negatives (when two spellings of a particular word remain different even after applying the transformation) and false positives (when the algorithm makes different but somewhat similar words equal).

**Example** | *Language: English.*  
| *Orthographies: British & American.*

*Say, we want to cope with the difference in the spellings of the words ‘colour/color’, ‘honour/honor’, etc. in British and American English. The obvious way is to replace ‘-our’ at the end of the word by ‘-or’ or visa versa. Such a rule will solve the problem, but it will also cause false positives, for example the words ‘tour’ and ‘tor’ will become the same, and so will do ‘four’ and ‘for’, etc. We can overcome this disadvantage by setting a constraint on the word length. Thus the final rule will be the following: if the word is more than four characters long and ends with ‘-our’, then this ending is replaced by ‘-or’. Another possible rule for this pair of orthographies is replacing ‘-ize’ with ‘-ise’ at the end of the word to eliminate the differences in the spellings of such words as ‘customise/customize’, ‘realise/realize’, etc.*

We must mention that if an orthographic normalizer and a stemmer are used together in the same system the occurrence of false positives can be caused not only by any of the two, but also by their combination. More formally, if we have two character sequences  $\mathfrak{w}_1$  and  $\mathfrak{w}_2$ , which cannot be neither different spellings nor different grammatical forms of the same word, and orthographic normalizer  $N$  and stemmer  $St$  are used in the system, it may happen that

$$N(\mathfrak{w}_1) \neq N(\mathfrak{w}_2)$$

$$St(\mathfrak{w}_1) \neq St(\mathfrak{w}_2)$$

but

$$N(St(\mathfrak{w}_1)) = N(St(\mathfrak{w}_2))$$

Therefore it is necessary to consider the false positives caused by stemmer and orthographic normalizer as one indivisible problem.

#### 3.2 Application on Practical Tasks: Grouping the Rules into Levels

The discussed solution by itself gives no possibility to adjust the balance between the false positive rate and the false negative rate. Nevertheless, this possibility can be achieved by grouping the rules into levels according to their potential ability to cause false positives. Then, the more levels are used, the more false positives and the less false negatives appear during the search. In this way the normalizer takes the form  $N(\mathfrak{w}, t)$ , where  $t$  is a parameter responsible for the balance between the two types of errors.

Modified in this way, for search engines the solution allows to use any of the two approaches described in Section 2.2. Deploying automatic cross-spelling redirections in a wiki engine is also easy enough: one just needs to implement an intermediate level of transformed titles in the database, as shown in Figure 1.

### 4. ORTHOGRAPHIC NORMALIZER FOR THE BELARUSIAN LANGUAGE

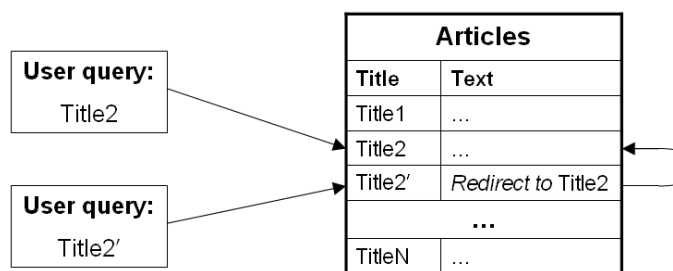
Our goal was to consider the two most popular variants of the Belarusian orthography, namely the reformed variant of 1933 and the so-called “tarashkevitsa” [7]<sup>1</sup>, both widely used on Belarusian websites. Our orthographic normalizer, which consists of 4 levels, is presented in Figure 2. Level L1 consists of four rules that represent some general and easily formalizable differences between the two orthographies and are supposed to cause no false positives. Level L2 consists of five more rules that can cause some false positives, but also must decrease the false negative rate seriously. Level L3a consists of three rules that can cause serious amount of false positives decreasing at the same time the amount of false negatives. The five rules of level L3b consider the differences in spelling of several frequently used roots of foreign origin (such as *kilo-*, etc.), and so are supposed to decrease the false negative rate without increasing the false positive rate much. All the five rules of this level are particular cases of the first rule of level L3a. Thus, level L1 is to be used where no false positives are allowed, levels L1, L2 and L3a together are to be used when false negatives are highly undesirable, and levels L1 and L2 together, as well as levels L1, L2 and L3b together, provide intermediate trade-offs between the two types of errors.

### 5. EXPERIMENTAL EVALUATION

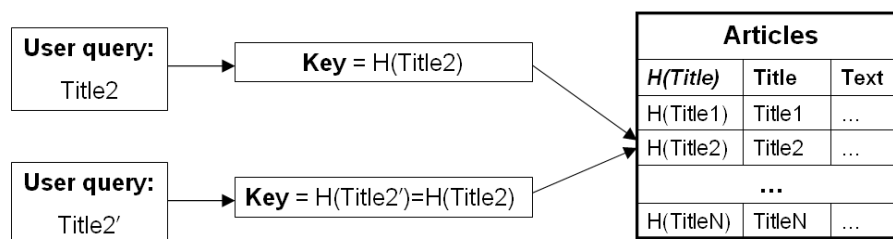
To perform preliminary evaluation of the performance of the proposed algorithm we established three experiments that are described below.

**Experiment 1.** The goal of the first experiment was to estimate the false negative rate of the orthographic normalizer. The data for this experiment was prepared as follows:

<sup>1</sup>This variants of the language differ not only in orthography, but also slightly in grammar, but the difference in orthography is much more noticeable.



(a) Usual variant: a wiki engine without an orthographic normalizer. The problem of multiple spelling solved through redirections.



(b) Proposed variant: a wiki engine with an intermediate layer of normalized titles in the database.

Figure 1: Using an orthographic normalizer in a wiki engine.

1. **Level L1**
  - (a) replace *ʹ* with *z*
  - (b) replace *дзвдз* with *ддз*
  - (c) delete *ь* if followed by a consonant (including *дз*) followed by *е, ё, і, ю, я, or в*
  - (d) replace *эь* with *эʹ* before *е, ё, і, ю, or я*
2. **Level L2**
  - (a) delete *ж,ш,з,х* if followed by *сх*
  - (b) replace *і* with *ы* after *з* or *с*
  - (c) replace *е* with *э* after *з, с, в, ф, м, н, б, or п*
  - (d) replace *ар* with *р* at the ends of the words if preceded by *д* or *т*
  - (e) replace *е* with *э* at the beginnings of the words
3. **Level L3a**
  - (a) replace *я,ё* with *а,о* after *л*
  - (b) delete *ь* after *л* if not followed by *е, ё, і, ю, or я*.
  - (c) replace *х* with *z*
4. **Level L3b**
  - (a) replace *лэгі* with *логі*
  - (b) replace *лягіч* with *лагіч*
  - (c) replace *оляг* with *олаг*, *ёляг* with *ёлаг*
  - (d) replace *ланд* with *лянд*
  - (e) replace *кіля* with *кіла* at the beginnings of the words

Figure 2: Orthographic normalizer for the Belarusian language.

2000 random entries were retrieved from an electronic orthographic dictionary for the “tarashkevitsa” orthography<sup>2</sup>, and then the corresponding spellings in the other orthography were entered manually. Of this 2000 words 1579 are written in the same way in both orthographies, and for 8 there are differences that probably cannot be resolved in other way than thesaurus (this are proper names that differ slightly in the two traditions). Thus, there are 413 entries that have differences that can potentially be eliminated by an orthographic normalizer. Of this 413 entries level L1 of the normalizer succeeded on 259 (63%), levels L1 and L2 together succeeded on 357 (86%), levels L1, L2 and L3a together succeeded on 388 (94%), and levels L1, L2 and L3b together succeeded on 365 (88%).

**Experiment 2.** The goal of the second experiment was to estimate the false positive rate of the orthographic normalizer. For this purpose the whole amount of words from the same orthographic dictionary as in Experiment 1 was used. It must be mentioned here, that such evaluation is not enough to estimate the real false negative rate because the dictionary contains only one form for each word. The result is the number of pairs of dictionary entries that represent different words but become undistinguishable after the transformation. This number is equal to 0 for level L1; to 2 for levels L1 and L2 together; to 60 for levels L1, L2 and 3a together; and to 2 for levels L1, L2 and L3b together.

**Experiment 3.** The goal of the third experiment was to evaluate the performance of level L1 of the orthographic normalizer on titles of Wikipedia articles. For this experiment the list of redirections on the Belarusian Wikipedia was used<sup>3</sup>. The redirections are 1141 in total, many of them

<sup>2</sup>The dictionary was downloaded from the page <http://pravapis.org/download.asp>

<sup>3</sup>The data for the experiment was downloaded from the

needed only because of the problem of multiple orthographies<sup>4</sup>. Experiment showed that level L1 of the normalizer makes 247 of the redirections unnecessary. Being combined with a thesaurus of three entries, containing three popular male names that are traditionally spelled differently in the two considered orthographies (Аляксандр vs. Аляксандар, Казімір vs. Казімер, Уладзімір vs. Уладзімер), the algorithm is able to cope already with 278 of the redirections.

In general, the achieved results are good, with the only problem of having nonzero false negative rate even with all the levels of the orthographic normalizer used.

## 6. CONCLUSION

In this paper we discussed the problem of multiple spellings, which may be an obstacle for efficient search in text data, paying attention to peculiarities of the problem statement for search engines and wiki engines. As a possible solution we discussed a substring-replacement orthographic normalizer, a simple stemming-like string processing algorithm which brings all words to a kind of a uniform spelling. A normalizer for the Belarusian language is presented and evaluated. The experiments prove that the designed algorithm is able to produce quite good results, though some improvement is still needed. The nearest future work in this direction includes more experiments for the evaluation of the proposed normalizer and deeper analysis of the differences between the considered orthographies in order to have a more efficient algorithm.

## 7. ACKNOWLEDGEMENTS

I would like to thank Ray Edwards, Symon Harner, and Julyan Holmes for their kind help in finding an appropriate example for Section 2.1 in the Cornish language.

## 8. REFERENCES

- [1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.
- [2] J. Halpern. Lexicon-based orthographic disambiguation in CJK intelligent information retrieval. In *Proceedings of the 3rd workshop on Asian language resources and international standardization*, 2002.
- [3] R. Hickey. Applications of software in the compilation of corpora. In M. Kytö, M. Rissanen, and S. Wrigg, editors, *Corpora across the centuries. Proceedings of the First International Colloquium on English Diachronic Corpora*, pages 165–186. Amsterdam & Atlanta: Rodopi, 1994.
- [4] H. Lee and A. Ng. Spam deobfuscation using a hidden markov model. In *Proceedings of Second Conference on Email and Anti-Spam, CEAS'2005*, 2005.
- [5] J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, pages 22–31, 1968.
- [6] H. Raghavan and J. Allan. Matching inconsistently spelled names in automatic speech recognizer output for information retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 451–458, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.
- [7] Wikipedia. Belarusian language. *Article accessed at 21.12.06*, 2006.

page <http://be.wikipedia.org/wiki/Special:Listredirects> at 08.01.2007.

<sup>4</sup>It is not possible to name the exact percentage, because with proper names, which constitute a large part of the titles, it is often hard to distinguish between the difference of the orthographies and the difference of the corresponding naming traditions.