Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 331–335
DOI: 10.15439/2014F406
ACSIS, Vol. 2

Data Cleansing of the Fire & Rescue Text Corpus.
The Case Study of Correction of the Misspellings and Segmentation into Sentences

Karol Kreński*, Mateusz Fliszkiewicz†
*Section of Computer Science, The Main School of Fire Service
ul. Słowackiego 52/54, 01-629 Warsaw, Poland
krenski@inf.sgsp.edu.pl,
†Section of Computer Science, The Main School of Fire Service
ul. Słowackiego 52/54, 01-629 Warsaw, Poland
fliszkiewicz@inf.sgsp.edu.pl

***Abstract*—The article presents a case study of applying data cleansing methods and segmentation procedures in order to correct and enhance the structure of the domain corpus of fire service. During the study we present our approach and the results in the task of correcting the misspellings, as well as the method of segmenting the corpus into sentences.**

***Index Terms*—Fire Service, Data Cleansing, Text Corpus, Misspellings, Segmentation**

I. Introduction

OVER the years, the National Fire Service of Poland collected a large corpus of texts from about 6 million incidents. Unfortunately, there was little validation of the input which resulted not only in the (too much) free form of the texts which is difficult to automatically process, but also in lots of misspellings and lack of structure (sentences boundaries) which further impede computer analyses.

Our research was focused on finding the characteristics of the above problems and by using mostly regular expressions, n-gram analysis, spell checkers and databases of some entities (e.g. geographic locations) as well as reference domain texts (fire & rescue journal) we tried to cleanse the corpus.

The paper is structured as follows: In section II we introduce the fire & rescue text corpus named EWID. In section III we present our motivation and the context of our research. In sections IV and V we provide the details of how we corrected the corpus.

II. Characteristics of the EWID corpus

The national fire and rescue services (just like, e.g., police) are typically equipped with the incident data reporting systems (IDRS), which gather the information about conducted actions. Each of approximately 500 Fire and Rescue Units (JRG) of the State Fire Service of Poland (PSP) conducts around 3 fire & rescue actions per day. After every action a report is created in EWID – the internal computerized reporting system of PSP [1]. As of 2014, the total number of the reports in EWID is around 6 million, of which about 0.3 million records were available for the purposes of this research. Each record contains 560+ attributes (only a few dozens are usually set per record). Most of these attributes provide yes/no information about action parameters (binary), but there are also timestamps, quantities and short text entries. There is one attribute which we consider distinct: the natural language *description* of the action.

The collection of the 0.3 million EWID *descriptions* contains about 60 MB of texts, which is about 8 mln of words, written in semi-natural, technical language. Following is a sample passage from the corpus (awkward vocabulary and misspelled words are intentional):

"Affer arriving at the fire scene the undergrowth fire was observed. Two firefighting jets ware applied and suction line from the nearby lake was created. After putting out the fire, appliance crew came back to fire station".

The concern is that over the years this large corpus which contains valuable information has been collected with limited validation of the input. This situation is considered quite common in the real world data collections [2], [3]. The corpus in scope requires data cleansing followed by further processing in order to improve the semantics. This case study is focused on the data cleansing only. It may be beneficial for other text corpora, which are affected by typographic errors. We know of projects where data cleansing step was explicitly skipped, as the expected solution was no other than a laborious human work [4], [5]. In this work we propose a mostly automatic, iterative process supervised by domain experts. It is important to mention that we are more interested in having the entities in the text unified (disambiguated) rather than grammatically correct. We assume, that for the purpose of further operations on EWID corpus, such as clustering, statistical analysis, and so on, this unification may be beneficial.

III. Motivation

The motivation for the corpus reparation is to prepare the ground for further processing. In our particular case there was a need to have the data cleansed when working on a concept of a decision support system named CLEWID [6]. CLEWID is a proposition of a platform for fire & rescue data analyses. It is built of 4 layers, namely: 1) the raw data layer (EWID

978-83-60810-58-3/$25.00

and other sources), 2) the quality data layer, 3) the granular (semantic) layer, 4) the models layer. The scope of this article is to transform the raw data layer into the quality data layer to have the higher level layers operate on the more precise and strict data. The semantics are fixed on the granular layer, since granulation is about organising the data based on various aspects of their similarity.

IV. Details of corrections of the misspellings

The process was divided into stages. Each stage describes another approach and provides the information how much gain was achieved.

1) Removal of redundant characters: The lack of the validation for the *description section* in EWID database results in various characters being incorrectly inserted. This may result in creation of alternate forms of the same entities (e.g. GBA3 vs GBA-3). The selection of the characters which should be removed requires the input from the expert – this step can be done by searching for the words containing non-alphanumeric characters and deciding which of the characters should be dropped. In the case of EWID corpus most of non-alphanumeric characters were replaced by space. Additionally, digits at the beginning of a word boundary and hyphens within word boundaries after a letter and before a digit were removed.

2) Frequent words not recognized by the dictionary: At this stage there were 8,044,535 words including 309,036 words which were not recognized by the popular *aspell* spell checker [1] (3.8% error ratio). 500 most frequent (the reasonable number for a human to manually process) of the 309,036 not recognized words were extracted from the corpus. 200 of these entries proved to be valid words from domain vocabulary – it was reasonable that aspell didn't have them in its database. Automatic corrections by aspell were proposed for the remaining 300 and they were later manually adjusted by the expert. The knowledge from the domain expert was instrumental in achieving reasonable outcome, as some cases were not quite obvious. For example, aspell proposed 'dzielenie' (division) as the correction for the misspelled word 'dzialenie', which was overruled by expert's 'działanie' (action). The corrections were applied to the corpus and the spell checker was rerun. The error ratio dropped to 2.9%. This particular fix was an example of a huge gain with little effort.

3) The additional dictionaries: In order to extend the spell checker, we searched for collections of the domain vocabulary. There exist a number of texts collections which could serve as a reference in composing the domain vocabulary (domain knowledge). Ultimately, the expert decided that the domain journal "Przeglad Pozarniczy" (*PP*)[2] would contain the texts that are most relevant for the operational content of EWID corpus. PP publishes fire & rescue related articles, and by the fact that it is a journal it (hopefully) contains very small amount of misspellings. We spell checked the acquired PP corpus and all the misspellings reported by aspell (words not found in standard dictionary) were treated as candidates for domain vocabulary, thus domain dictionary was created. The extended spell checker reported error rate of 2.15%.

The idea to use a good quality domain corpus as an extension of a spell checker came after we already fixed the 500 most frequent words manually (as described in the previous section). The spell checker extended by journal-based domain vocabulary would likely have recognized most of the frequent words from EWID since they come from the same domain of fire & rescue. The lesson learned is that the domain vocabulary should be used as early as possible (if it is available).

By knowing the content of EWID the expert added more elements to dictionary. Geographical entities – streets, cities and districts were obtained from the external public sources (Polish governmental/administration organisations) and became another extension to the dictionary. The spell checker extended with the domain vocabulary and geographical entities was rerun and the error ratio dropped to 1.75%.

Another key step was the inclusion of surnames. Surnames in EWID are frequently reported as misspellings by aspell. Fortunately most first names are recognized by aspell. The public database of Polish surnames and first names was acquired and roughly checked for the completeness against our students surnames database (around 200 entries). 97% of the surnames were recognized, so the completeness of the surnames database was reasonable. However, this mechanism proved to be too greedy – too many actual misspellings were being forgiven as possible surnames. We needed to drop the surnames database. The spell checker extended with domain vocabulary, geographical entities and names reported 1.56% of errors.

4) The n-gram approach: The knowledge-based extensions of spell checker's dictionary exhausted the inventory of easy fixes. The remainder of the misspellings in the corpus required more extensive approach. The method that we have applied replaces (corrects) a misspelled words using their nearest correct neighbor. The neighbor(s) of a given word needs to be identified in a meaningful way. For this purpose, the list of all 3-grams (unique triplets of words in the corpus) was created. This list was spell checked with the use of the extended spell checker introduced above and, as a result, split in two. The `3-grams.correct` and `3-grams.errors` contain 3-grams recognized as correct and misspelled, respectively. Then the `3-grams.errors` list was iterated to find the nearest entry on the `3-grams.correct` list. The measure we use is the Levenshtein (editorial) distance [7]. The correction was applied if the distance between the misspelled trigram and correct trigram was less or equal 2. The threshold of 2 was set by the domain expert after his inspection of a sample of such corrections.

At this stage we faced a computational problem. The corpus is a collection of about 8 mln words. The building of the 3-grams.errors (about 0.3 mln entries) and 3-grams.correct (about 2 mln entries) databases proved to be unexpectedly quick. However, the performance of finding the closest match for each entry from 3-grams.errors in 3-grams.correct database

[1] GNU aspell, http://www.aspell.net/

[2] ISSN 0137-8910, http://www.ppoz.pl/

was very poor when choosing a simple approach: for each trigram in 3-grams.errors iterate over 3.grams.correct and calculate the Levenshtein distance between the two elements in each step. Database indexing of 3-grams.errors was not an options, since we didn't operate on exact matches, but needed to always calculate the difference.

Therefore we searched for a better method than scanning this large corpus and calculating the distance. The imaginary example below illustrates our solution:

Let us consider an example corpus of words

a-correct b-correct c-error d-correct e-error f-error.

For this corpus, there are 4 possible trigrams. Let us search for contexts contain the c-error:

(a) *a-correct b-correct c-error*

(b) *b-correct c-error d-correct*

(c) *c-error d-correct e-error*

A number of conclusions can be drawn from this observation: 1) the best context to fix the c-error is the (b)-trigram as it provides most likely the best (left and right) context for the misspelling, 2) the trigrams are redundant – it is enough to consider just one from the three above to have the c-error placed in the context, 3) the other two words in each trigram can be either correct or misspelled.

The third conclusion can be inspected further. At this stage the corpus contained around 2% of words with errors. The chances for two misspelled words occurring in one trigram seem low; assuming the misspellings are normally distributed across the corpus – there should be very few such trigrams. However, the assumption of normality proved to have a flaw, since in the population of humans, there are ones that tend to produce misspellings and others who do not. The result is that there occur trigrams with 2 misspelled words and less (but still) with all 3 words misspelled. Luckily, the prevailing majority of the trigrams were composed of one misspelling in the context of two correct words.

Considering the above, our approach proceeded as follows: the trigrams containing misspellings were split into two groups: i) a large group of trigrams with only one misspelled word, and ii) a small group of trigrams with two or three misspelled words.

Concerning the ii) group the plan was simple: the accepted Levenshtein distance was increased from 2 to $2 \cdot n$, where n is the number of misspelled words in the trigram. Then these trigrams were a subject to a linear scanning through the `3-grams.correct` database and because the small number of misspelled trigrams it proved not to be a computational issue.

In the group i) we started with our conclusion that trigrams are redundant. There are 3 setups for a misspelled word to be placed in the trigram, of which we choose the scenario (b) *b-correct c-error d-correct* (misspelling in the middle). The other two setups can be safely dropped since the goal of fixing the misspelled word can be achieved based on just a single context. The trigram was then reorganized into an associative array with the context as the key and the misspelled word as the value, i.e. *key="b-correct d-correct"* and *value="c-error"*. This structure later evolved: since between *b-correct* and *d-correct* more misspelled words may appear in the corpus, the value of the array should be a placeholder for more objects than just *c-error*. Therefore the final data structure has the form: *key="b-correct d-correct"*, *value="array('x1-error', 'x2-error', 'xN-error')"*. The same data structure was applied to the `3-grams.correct` database. The keys were hashed. The task has now become the searching for a hashed key of the misspelled trigram in the hashed keys of `3-grams.correct` database. Once the matching key is found, the Levenshtein distance between the given *xN-error* word and all the correct words (a small array) for the corresponding key in `3-grams.correct` database is calculated. This method proved to be very effective computationally and resolved the issue. The overall error rate dropped to under 1% after incorporating the ngrams method.

V. The Segmentation into Sentences and the Abbreviations

Another step in enhancing the nature of the date was the segmentation of corpus into sentences. It is important to note that the standard procedures of segmentation into sentences assume that the corpus is rather free from misspellings, that upper/lower case and other language rules are strictly obeyed – for such pure corpora the approaches like [8] could be more easily applied.

There is a couple of aspects related to sentences: 1) it is not proper to treat a dot as a terminator of a sentence since dots also appear in abbreviations 2) words which end with a dot may be not recognized by aspell either because they are misspelled or because they are correct domain abbreviations not known to aspell 3) for the n-grams analysis: trigrams should not cross the sentence boundaries 4) having the corpus segmented into sentences allows for enhanced further processing in more abstract layers. In many applications the sentences can be the smallest building blocks, e.g. in the Computer Aided Translation systems such as OmegaT [3] the sentences are atoms.

For the sake of simplifying our further considerations, let us introduce the terms *a sentence terminator* meaning *the last word of the sentence* and *a dotted word* meaning *word ending with a dot*.

We tried to automatically extract the abbreviations from the corpus. First we found all dotted words and sorted them by the number of the occurrences in the corpus. Table I is the header of the resulting list.

As this list extends there are less and less abbreviations, but we can not make any assumptions that after a certain position of this list there won't be any abbreviations. This is particularly true if we realize that the distribution of words in a text corpora is a Zipf distribution [9]:

"In human languages, word frequencies have a very heavy-tailed distribution, and can therefore be modeled reasonably well by a Zipf distribution (...)" [10].

[3] OmegaT, The free (GPL) translation memory tool, http://www.omegat.org

TABLE I
OCCURRENCES OF DOTTED WORDS

word	en translation	occurrences	abbreviation?
st.	fireman	54716	Y
ul.	street	40162	Y
C.	Celsius	39772	N
sprawny.	operating	26733	N
ok.	around	22184	Y
temp.	temperature	18295	Y
p.	floor	17087	Y
ůmieci.	garbage	13662	N
zdarzenia.	incident	12464	N
wody.	water	9706	N
budynku.	building	8097	N
lasu.	forest	7159	N
zach.	west	6368	Y
...			

The result from being a heavy-tailed distribution is that most words in the corpus (say 80%) appear relatively seldom (say 3 times). Taken the large number of the words, that means that we should be aware of the weakness of any manual action against the corpus, as we will only process a small portion of all the entities. Therefore, we look for a more automatic way of distinguishing the abbreviations from the sentence terminators.

We assumed that any sentence terminator may also appear in other position than at the end of the sentence, thus not end with a dot. Then we inspected the fraction: $f = w_d/(w_d+w)$, where w_d is *the frequency of occurrence a dotted word* and w is a *the frequency of occurrence of the same word without a dot*. f should return higher values for abbreviations. By inspecting the table II we can expect that the good threshold should be somewhere around 0.50 – higher values would be the abbreviations, lower values would be sentence terminators.

TABLE II
OCCURRENCES OF DOTTED WORDS AS A FRACTION OF $dotted/(dotted + notdotted)$. HIGH VALUES SHOULD INDICATE ABBREVIATIONS

word	en translation	fraction	abbreviation?
st.	fireman	0.99	Y
ul.	street	0.95	Y
C.	Celsius	0.10	N
sprawny.	operating	0.28	N
ok.	around	0.84	Y
temp.	temperature	0.89	Y
p.	floor	0.77	Y
ůmieci.	garbage	0.16	N
zdarzenia.	incident	0.09	N
wody.	water	0.14	N
budynku.	building	0.20	N
lasu.	forest	0.34	N
zach.	west	0.52	Y
...			

This method allows for pretty good results and abbreviations can be easily separated. However, a quick inspection of the full list reveals that false positives (not abbreviations) happen for values of above even 0.80. Therefore a few more features are added:

1. The number of characters in the word. Abbreviations should be short, that is an implicit part of their definition. The dotted words were getting benefit/penalty points for being short/long.

2. Position at the end of a paragraph indicates towards a sentence terminator. We added benefit points for each dotted word ending any paragraph.

3. Similarly, position directly before the beginning of a sentence indicates towards a sentence terminator. How to define the beginning of a sentence? The first idea was to treat any word beginning with an upper case as a likely beginning of a sentence. However, "kpt. John Snow" phrase quickly proves it is not entirely true. Instead, we built a list of bigrams starting with an upper case. Then we selected just the bigrams that occur often, more often than bigrams containing Names and Surnames – "John Snow" is not a frequent phrase in the corpus mentioning probably thousands of humans. Such frequent bigrams should very likely be the beginnings of the sentences. We added benefit points for the dotted word if it occurred before any beginning of a sentence (one occurrence is sufficient as it proves that such a word is a proper sentence terminator).

Finally we constructed the classifier based on the above features 1) the ratio of occurrence the dotted word with/without the trailing dot 2) the number of characters in the word 3) position at the end of a paragraph 4) position before the beginning of a sentence. The classifier was simply the sum of the 4 indicators, each of them normalized to <0,1> range. The list of sentence terminators was obtained and the corpus was segmented at each point were the sentence terminator with a trailing dot occurred. Manual browsing proved that this method was correct in about 97% cases, which seems a good score.

A. An example of the segmentation of a block of text into sentences

Let us illustrate our approach with the segmentation of an imaginary block of text: *Today temp. was 10 C. Strong wind from east. The fire was successfully put out.*

There are following dotted words to consider: temp, C, east, out. According to all of our considerations, the following would happen:

a) temp seldom appears without a dot in the EWID corpus. Some humans tend to write it without a dot (which is a mistake), but most write it properly and the statistics suggest it is an abbreviation. Our classifier correctly identified it as an abbreviation.

b) C is very short which suggest an abbreviation. Some firemen do follow C by a dot: "10 C." while it should be "10 °C". However, C is often spotted without the trailing dot, also spotted before the beginning of a sentence or even at the end of paragraphs which is a strong premise for a sentence terminator. Our classifier identified it as a sentence terminator.

c) east appears at the end of paragraphs and often without a dot. Our classifier identified it as a sentence terminator.

d) out appears at the end of paragraphs and often without a dot. Our classifier identified it as a sentence terminator.

The above block of text was therefore split into 3 sentences, which is correct.

B. Discussion

The knowledge-based data (text) correction method that we propose makes it possible to reduce error (typo) ratio from 4% down to below 1% (four-fold) in the EWID corpus. The cleansing/correction methods described above may also be tweaked for clearing the corpus from sensitive and private data. For example, there is an issue with sharing EWID corpus because it contains personal data (names, addresses, etc.) These sensitive data are not always easy to pinpoint, and the presented methods may help in this task, making anonymization of the text corpus feasible.

We also managed to quite successfully segment the corpus into sentences – the algorithm correctly proposed the endings of the sentences with about 97% accuracy. EWID system was hopefully carefully designed, but life often proves to find shortcomings in many designs, once these designs start to operate in the real world. The issue is that the designers seem to have lost their control over the content – there are difficulties in finding certain, fuzzy information (e.g. finding the information about all the accidents with the buses). Over the years EWID became the collection of lots of information in form of unstructured texts and it became a playground for researches like this one. One of our future ideas is to semantically inspect the content of EWID and sentences seem to be the proper building blocks for such an analysis. Once we have the sentences correctly defined we can cluster the whole corpus based on the sentences and then inspect the meaning (semantics) of each cluster. We believe that the system could improve the validation of the input by checking the input against it's knowledge base and then tag/correct/propose or otherwise interact with the human introducing the data.

What was learned from the experiment is a confirmation of [11]: "Usually the process of data cleansing cannot be performed without the involvement of a domain expert, because the detection and correction of anomalies requires detailed domain knowledge. Data cleansing is therefore described as semi-automatic but it should be as automatic as possible because of the large amount of data that usually is be processed and because of the time required for an expert to cleanse it manually. The ability for comprehensive and successful data cleansing is limited by the available knowledge and information necessary to detect and correct anomalies in data.".

The process of data cleansing has an iterative nature. Different aspects appear after the nature of data is better know, new thresholds must be checked, then parameters tweaked and then the whole process must be rerun. There is a difficulty with the order of the undertakings. On one hand we would like to start segmenting into sentences very early in the whole process of data cleansing. But at this time we would like to have the misspellings fixed already. In order to fix misspellings on the other hand, we use n-grams analysis which should not cross the sentences boundaries, but the sentences boundaries are not yet defined. We therefore need to run the analyses simultaneously and iteratively, as stated before. There is also the question whether bothering with data cleansing is worthwhile – the alternative is to accept that there is noise in the data (google and other search engines accept such noise after all). Answering the question of how much gain we achieve by cleansing the data would be possible after performing specific researches in higher level layers, e.g. the proposed CLEWID platform, where models operate on these lower level data. However, we didn't conduct such experiments.

ACKNOWLEDGMENT

Supported by Polish National Centre for Research and Development (NCBiR) – Grant No. O ROB/0010/03/001 in the frame of Defence and Security Programmes and Projects: "Modern engineering tools for decision support for commanders of the State Fire Service of Poland during Fire&Rescue operations in the buildings".

REFERENCES

[1] C. work, "Ewidencja zdarzeń - EWID99," Abacus, http://www.ewid.pl/, Tech. Rep., [Access: 23.04.2014].

[2] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data mining and knowledge discovery*, vol. 2, no. 1, pp. 9–37, 1998.

[3] M. L. Lee, H. Lu, T. W. Ling, and Y. T. Ko, "Cleansing data for mining and warehousing," in *Database and Expert Systems Applications*. Springer, 1999, pp. 751–760.

[4] P. Elzinga, J. Poelmans, S. Viaene, G. Dedene, and S. Morsing, "Terrorist threat assessment with formal concept analysis," in *Intelligence and Security Informatics (ISI), 2010 IEEE International Conference on*. IEEE, 2010, pp. 77–82.

[5] J. Poelmans, P. Elzinga, G. Dedene, S. Viaene, and S. Kuznetsov, "A concept discovery approach for fighting human trafficking and forced prostitution," *Conceptual Structures for Discovering Knowledge*, pp. 201–214, 2011.

[6] A. Krasuski, K. Kreński, P. Wasilewski, and S. Łazowy, "Granular approach in knowledge discovery," in *Rough Sets and Knowledge Technology*. Springer, 2012, pp. 416–421.

[7] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet physics doklady*, vol. 10, pp. 707–710, 1966.

[8] M. Rudolf and M. Świdziński, "Automatic utterance boundaries recognition in large polish text corpora," in *Intelligent Information Processing and Web Mining*. Springer, 2004, pp. 247–256.

[9] G. K. Zipf, "Selected studies of the principle of relative frequency in language." 1932.

[10] Wikipedia, "Zipf's law," http://en.wikipedia.org/wiki/Zipf's_law, [Access: 23.04.2014].

[11] H. Müller and J.-C. Freytag, *Problems, methods, and challenges in comprehensive data cleansing*. Professoren des Inst. Für Informatik, 2005.