Extraction of Folksonomies from Noisy Texts

Wim De Smet
K.U.Leuven - ICRI-LIIR
Celestijnenlaan 200A
3001 Heverlee
Belgium
wim.desmet@law.kuleuven.be

Marie-Francine Moens
K.U.Leuven - ICRI-LIIR
Tiensestraat 41
3000 Leuven
Belgium
marie-france.moens@law.kuleuven.be

Abstract

We built a system for the automatic creation of a text-based topic hierarchy, meant to be used in a geographically defined community. This poses two main problems. First, the appearance of both standard language and a community-related dialect, demanding that dialect words should be as much as possible corrected to standard words, and second, the automatic hierarchic clustering of texts by their topic.

The problem of correcting dialect words is dealt with by performing a nearest neighbor search over a dynamic set of known words, using a set of transition rules from dialect to standard words, which are learned from a pair-wise lexicon. We tackle the clustering problem by implementing a hierarchical co-clustering algorithm that automatically generates a topic hierarchy of the collection and simultaneously groups documents and words into clusters.

Keywords

Dialect, phonology, edit-distance, folksonomy, co-clustering

1. Handling dialect words

Dialects are variations of a language spoken by a larger community, bound to a particular geographic area. In our research, this area consisted of the Belgian city of Hasselt. The main difference between a standard language and dialects is the consensus on spelling. A standard language is required to have an official written equivalent. Dialects on the other hand, usually have only a spoken component. Written versions appear however when a touch of "couleur locale" is wanted, as in a city-based folksonomy. Because of the lack of consensus on spelling, dialect words are not apparent in standard lexicons. A first category of dialect words have the same origin of standard words, but are pronounced differently and their written equivalent mimics the dialect pronunciation. In other cases, completely new words appear in dialects that do not necessarily have a one-to-one translation with standard words. Both situations make that the texts contain much more noise than standard texts.

Because in our clustering model we rely on the words of the texts, it is vital that words are as much as possible normalized to standard language. The algorithm starts from the dictionary D, an initial list of known words spelled correctly.

When processing a text, our goal is to resolve unknown words and expand D with words we assume spelled correctly. An unknown word (i.e., $\notin D$), can belong to one of three categories. First, a standard word that does not yet occur in D. Second, a misspelled version of one of D's words. Third, a dialect word. Since dialect words do not have an official spelling, we will not distinguish between "correct" and "misspelled" dialect words.

Dialect texts are processed and upon sufficient similarity of an unknown word with a known word, expressed by its *dialect edit distance* or *ded* (see further), it is normalized to this standard word.

If the unknown it cannot be normalized to a known word (*ded* is too large), it is added to D in one of the following two cases. First, if the unknown word occurs frequently enough in the texts, we rule out the possibility that it is coincidentally mistyped, and assume it correctly spelled. This is the *frequency assumption*. If the smallest *ded* to D exceeds a threshold, we assume it unlikely that a simple spelling error occured. This is the *distance assumption*.

1.1 Dialect edit distance

In case the dialect and standard word have the same roots, the dialect word has evolved to a different spelling by following a set of rules that determine the pronunciation. These rules comprise, for example, contractions or alterations of vowels, and are often (but not always) dependent on the context of letters they appear in. They are however not exclusive: the same phonetic entity can be expressed by several combinations of characters. For example, the Dutch vowel "o", would in the Hasselt dialect be written as "eu" or "ö", both resulting in the same vowel (different from the Dutch "o").

To be able to incorporate these rules, we adapt the edit-distance algorithm, as developed by Levenshtein [3]. The Levenshtein-distance calculates the minimal number of character operations, necessary to transform one word into another. In the original paper, three different character operations are considered: *deletion*, *insertion* and *substitution*. With each operation adding a cost of 1, the algorithm creates a matrix to derive which order of operations generates the least total cost.

Our adaptation gives each operation a dynamic cost, depending on the type of operation and the characters involved. This allows for alterations, typical for the dialect, to cause only a small difference between a dialect word and the standard word.

To learn the cost of each different operation, we make use of a pair-wise lexicon of dialect words and their standard equivalent.

After calculating the standard edit-distance matrix between every pair, we remove the pairs where $\frac{edit-distance}{word\ length}$ exceeds a threshold, as they are not likely to be phonetically related.

For the remaining words, the distance matrix provides the alignment between characters on which the two words concur, and which operations transform the dialect word to the standard word. For each operation O, we store its type (substitution s, insertion i or deletion d), its parameter letters (only x for insertion and deletion, x and y for substitution), and the context C(an n-letter two-sided window around the *dialect* letter). If the beginning or the ending of the word falls inside the context, then this is explicitly stored. We represent it as a following vector: $[x_{i-n}, \ldots, x_i, \ldots, x_{i+n}]$. Because the width of our context window is arbitrarely chosen, we might "overencode" information. If an operation typically occurs after one specific letter, we may lose information by storing two letters extra before, and two letters after. Therefore we also store the following subsets: $\forall v, w \in [0, n] : [x_{i-v}, \ldots, x_i, \ldots, x_{i+w}]$.

There are several options to calculate the operation's cost. This cost must lie within $[0, 1]$, and be inversely proportionate to the confidence. One way to estimate this confidence is by the formula $conf(O_C) = \frac{|O_C|}{|C|}$, $|C|$ = being the number of times context C appears in the corpus. The cost-function $f(|O_C|, |C|)$ can then be expressed as $1 - conf(O_C)$.

We feel however that this relative frequency does not capture all information. Consider, for example, two operations with a relative frequency of 50%. The first however has an $|O_C|$ of 2 and a $|C|$ of 4, while the other one has an $|O_C|$ of 20 and a $|C|$ of 40. Because operation 2 occurs 10 times more than operation 1, we wish to give it a higher confidence, and a lower cost. The cost-function we designed has for this purpose has the following characteristics. If $|O_C| = 0$, the operation did not occur in the training data, and its cost equals 1. If $|O_C| = |C|$ ($|O_C|$ cannot exceed $|C|$), it receives the minimum cost, associated with the value of $|O_C|$. This cost decreases monotonally on increasing $|C|$. For values of $|O_C|$ between 0 and $|C|$, we calculate the quadratic interpolation between 1 and the minimum cost.

We defined the monotonically decreasing function for ($|O_C| = |C|$) as $g(|C|) = 1 - \frac{1}{1+\log(1+|C|)}$, which leads, together with the quadratic interpolation to $f(|O_C|, |C|) = g(|C|) * |O_C|^2 - 2 * g(|C|) * |O_C| + 1$.

When presented with a dialect word, the *dialect edit distance* between this word and all known standard words in the dictionary list D is calculated. For every entry in the distance matrix, the cost for each type of operation is determined. This requires the parameter letters, and the letter context of the operation. Again, all subsets of the context are generated, and for each subset the cost is calculated. The minimum cost among these is selected as the operation cost.

2. Topic hierarchy generation

Instead of explicitely extracting tags or keywords from documents, we try to build the topic hierarchy by grouping documents related by their topics into clusters, and create a hierarchy of these clusters. We associate representative words to each cluster. This way, the cluster hierarchy is translated into a topic hierarchy.

The method we used for this is a co-clustering algorithm, developed by Dhillon in [2], and adapted for topic hierarchy generation and associated clustering by Xu and Ma [4]. This algorithm transforms documents and words into a comparable format, and hierarchically clusters related documents together with the words relevant to that cluster.

# Training data	Edit Distance	Dialect Edit Distance
500	26.4%	39.6%
1000	28.2%	43.2%
2000	27.4%	46.0%
5000	27.2%	41.2%

Table 1: *Accuracy of resolving 500 dialect words*

3. Evaluation

We focus on the evaluation of the dialect edit distance, as the topic hierarchy generation failed to generate satisfying result on our current corpus. This is caused by a low overlap in the documents' topics and paraphrasing, as well as the lack of NLP tools (parsers etc.) capable of processing dialect texts.

To evaluate the dialect edit distance, we used an aligned corpus of dialect and standard words, consisting of 10,000 pairs [1] called T, and a lexicon D of standard words with correct spelling (among which the standard words from T). The particular dialect-standard corpus used in this evaluation was composed by a folklore organization, dedicated to the preservation of the Hasselt (and related) dialect. We evaluated our algorithm by training it on different sizes of subsets of the corpus, discovering the dialect's rules.

Using these rules, for every dialect word d from a test subset with fixed size, we determined the word from D with the smallest dialect edit distance and compared it to the standard word associated with d. To provide comparison with a baseline approach, we repeated the experiments using the standard edit distance. Results can be seen in table 1.

As apparant from the table, the Dialect Edit Distance has an average improvement of 15% accuracy over the standard Edit Distance. Not every word in our test data is a phonetic transformation of their standard word however, some words are (phonetically) unrelated. Those words can not be resolved to the standard word using the Dialect Edit Distance. When selecting a random sample of our dialect dictionary as our test data, we did not filter out these words, as they also occur in the texts we wish to process. If we are to remove them from our test data, we expect our results to improve.

References

[1] Agl: `http://www.limburghuis.nl/interact/menu_wdb.html`, 2001.

[2] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[3] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[4] Gu Xu and Wei-Ying Ma. Building implicit links from content for forum search. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 300–307, New York, NY, USA, 2006. ACM Press.