Improving IR performance from OCRed text using cooccurrence

Kripabandhu Ghosh and Anirban Chakraborty

Indian Statistical Institute, Kolkata, West Bengal, India

1 Introduction

Since the original version of the corpus is not always available, we propose an algorithm which can be used find the error patterns without the original version. We look to group the related words in the OCRed text and map them to the query words. Thus, we make an effort to find the erroneous variants of the query words using co-occurrence information from the OCRed corpus.

We say that two words co-occur if they appear in a window of certain number of words of each other in a document. To find similar words with respect to string match, we have considered Longest Common Subsequence (LCS). For the two words *industry* and *industrial*, the LCS is *industr*. We use a normalized similarity score, LCS_similarity to measure string match between two words. We define LCS_similarity between words w_1 and w_2 as (length of LCS(w_1, w_2))/(maximum length of w_1 and w_2)). So, LCS_similarity(*industy*, *industrial*) is 0.7. However, LCS_similarity(*industrious*, *industrial*) is 0.73. This is alarming! So, blind dependence on string matching can bring unrelated words with high string matching similarity together. The use of context information can help in adding another level of filtration in such situations. One popular way of capturing context match between words is co-occurrence statistics. The hypothesis is : words with high co-occurrence over the whole corpus share the same context and if such words have high string match, then they are very likely to be *variants* of each other. The *variants* can be of different types. In this work, we are looking to cluster all the erroneous (caused by OCR misrecognition) and inflectional *variants*. Our method, therefore, is purely statistical and language independent.

2 Our approach

2.1 Clustering algorithm

For each word w in the OCRed corpus:
Let S_{w1} and S_{w2} be empty sets. Let $S_w = S_{w1} \cup S_{w2}$

1. For word w_1 co-occurring with w, calculate LCS_similarity between w and w_1. Store w_1 in S_{w1} if LCS_similarity(w, w_1) > some threshold T.
2. For each w' in S_{w1}, find the words w_2 co-occurring with w' such that LCS_similarity(w, w_2) > T. Include all these words in S_{w1}.
3. Repeat step (2) until no new word is added to S_{w1}.
4. Consider top m (in terms of frequency in corpus) words co-occurring with w. For each such word w_3, find the words w_4 cooccurring with w_3 such that LCS_similarity(w, w_4) > T. Include all these words in S_{w2}.
5. For each w'' in S_{w2}, find the words w_5 co-occurring with w'' such that LCS_similarity(w, w_5) > T. Include all these words in S_{w2}.
6. Repeat step (5) until no new word is added to S_{w2}.

The first step is aimed at finding the variants (e.g., *tobacc*, *1obacco*, etc.) of w (e.g., *tobacco*) which co-occur with w. The next two steps are to capture those variants which co-occur with the variants that have entered S_{w1} in the previous pass. The next three steps are an attempt to capture the variants of w which co-occur neither with w and its variants obtained by (1)-(3). This is based on the assumption that the words co-occurring with w co-occur with the missed variants. For example, a variant *tobac1* of *tobacco*

may co-occur with *cigarette*, which co-occurs with *tobacco*. Note that, *tobac1* does not co-occur with *tobacco*, but it does with *cigarette*. But *cigarette* in turn co-occurs with *tobacco*. So, we get *tobac1* through *cigarette*. This is an example of one level of indirection. We look to capture multiple levels of indirect relationships in our algorithm.

2.2 Mapping of query words with the appropriate cluster

Next, we map the query words with the appropriate clusters. By appropriate cluster of a query word w_q, we mean the cluster containing words of OCRed corpus that contain the *variants* of w_q. We use the following algorithm:

For each query word w_q,

1. calculate LCS_similarity(w_q, w_C), where w_C is a word in cluster C, for each word in the corpus
2. Choose all the clusters C for which LCS_similarity(w_q, w_C) is greater than a high threshold
3. For each cluster C obtained from step (2) define $C' = C \cup \{w_q\}$
4. Create complete-linkage clusters from each C' of step (3) and keep those clusters containing w_q
5. For a cluster C, let us consider LCS_similarity between each pair of words in it. Let GM_C denote the Geometric Mean of LCS_similarity of all the pairs. Then, compute GM_C for each cluster given by step (4)
6. Select the cluster C with maximum GM_C as the appropriate cluster for w_q

We expand w_q with the words in the *appropriate cluster* obtained above.

3 Results

Run	**MAP**	**P5**
Original text	0.2567	0.3485
OCRed text	0.1791	0.2738
Proposed method on OCRed text	0.1974	0.2831

Table 1. Results

Table 1 shows that our method yields improvement over the run on OCRed corpus but is not as good as the retrieval result from original corpus.