PROBABILISTIC MATCHSIMILARITY MEASURE FOR DOCUMENT CLUSTERING

Selvi K. [1], **Suresh R.M.** [2]
[1,] Research Scholar, Sathyabama University, Chennai, India
[2] Member, IEEE, Director, Chennai Institute of Technology, Chennai, India
[1]ssi.cse@rmkec.ac.in

Abstract —

Machine Learning captures the intrinsic characteristics of natural language, synonymy and polysemy. Investigations indicate that Similarity Measure is fundamental to a variety of tasks such as Clustering. and Classification.Much work has been done by researchers on document clustering with the use of semantic properties. In this paper, we develop a Probabilistic match similarity measure that naturally extends the recently proposed Web-based kernel function which are trained and tested to cluster the documents effectively. We consider two approaches to learning (similarity metric and preference ordering) and both achieved higher precision scores as compared to all other similarity measures. This method works well for Web tasks such as query/keyword matching and search query suggestion that rely heavily on the quality of similarity measures between short text segments. We show that the learned measures are efficient at a wide range of scales and achieve better results than existing similarity measures.

Key words: Text Mining, Similarity Measure, Machine Learning, Natural Language Processing, Document Clustering.

I. INTRODUCTION

The problem of measuring the similarity between two very short text segments has become increasingly important for many Web-related tasks. Examples of such tasks include query reformulation (similarity between two queries) , search advertising (similarity between the user's query and advertiser's keywords), and product keyword recommendation (similarity between the given product name and suggested keyword). Measuring the semantic similarity between two texts has been studied extensively in the IR and NLP communities. However, the problem of assessing the similarity between two short text segments poses new challenges. Text segments commonly found in these tasks range from a single word to a dozen words. Because of the short length, the text segments do not provide enough contexts for surface matching methods such as computing the cosine score of the two text segments to be effective. On the other hand, because many text segments in these tasks contain more than one or two words, traditional corpus-based word similarity measures can fail too. These methods typically rely on the co-occurrences of the two compared text segments and, because of their lengths, they may not co-occur in any documents even when using the whole Web as the corpus. Finally, because of the diversity of the text segments used in these Web applications, linguistic thesauruses such as Word- Net do not cover a significant fraction of the input text segments. In order to overcome these difficulties, researchers have recently proposed several new methods for measuring similarity of short text segments (Sahami & Heilman 2006; Jones et al. 2006; Metzler, Dumais, & Meek 2007). In this paper, we study the problem of measuring similarity of short text segments. In a general query suggestion scenario: given a short text segment q and a list of suggestions {s1, s2, ..., sn}, we would like to rank suggestions based on their similarity to q or select a subset of suggestions that are similar to q. Our contributions are as follows. First, we introduce a probabilistic match similarity measure which improves the web-based kernel method (Sahami & Heilman 2006) through a new term weighting scheme. Instead of using the traditional TF×IDF score or its variations, we use the "relevancy" of the words to the document, estimated by a stateof-the-art keyword extractor (Yih, Goodman, & Carvalho 2006). Second, in order to leverage the strengths of different similarity measures, we propose to combine them using machine learning. In particular, we consider two learning approaches: one directly models the similarity between a query and a suggestion (q, si) and the other models the preference ordering between two suggestions si and sj, with respect to the same query q. Finally, we present an experimental comparison between existing approaches

for measuring similarity between short text segments and our enhanced similarity measures. The experiments indicate that our methods are significantly better than existing methods. The rest of the paper is organized as follows. We first review existing methods for measuring similarity of short text segments. We then introduce our Web-relevance similarity measure and the proposed learning approaches, followed by the experimental evaluation.

II. Related Work

Translation models, in a monolingual setting, have been used for document retrieval [1], question answering [10], and detecting text reuse [9]. The goal is to measure the likelihood that some candidate document or sentence is translation (or transformation) of the query. However, such models are less likely to be effective on very short segments of texts, such as queries, due to the difficulty involved in estimating reliable translation probabilities for such pieces of text. Query expansion is a common technique used to convert an initial, typically short, query into a richer representation of the information need [7,12,16].This is accomplished by adding terms that are likely to appear in relevant or pseudo-relevant documents to the original query representation. In our query-query matching work, we explore expanding both the original and candidate query representations. Sahami and Heilman proposed a method of enriching short text representations that can be constructed as a form of query expansion [13]. Their proposed method expands short segments of text using web search results. The similarity between two short segments of text can then computed in the expanded representation space. The expanded representation and DenseProb similarity measure that we present in Sections 3 and 4 are similar to this approach. However, we estimate term weights differently and analyze how such expansion approaches compare, in terms of efficiency and effectiveness, to other standard information retrieval measures. Finally, since we evaluate our techniques on a query-query similarity task, it should be noted that this problem, and the related problem of suggesting and identifying query-query reformulations has been investigated from a number of angles, ranging from machine learning approaches [4] to query session log analysis[2].These techniques are complimentary to the core representational and similarity ideas that we explore in our work.

Ling Zhuang Honghua Dai 2004 introduced the initial points as centers for k-means algorithm. However, k-means clustering is a completely unstructured approach, sensitive to noise that produces an unorganized collection of clusters not favorable to interpretation [8].

III. Text Representations

Text representations are an important part of any similarity measure. In this section, we describe three different ways of representing text. Although these representations can be applied to text of any length, we are primarily interested in using them to represent short segments of text.

A. Surface Representation

The most basic representation of a short segment of text is the surface representation (i.e. the text itself). Such a representation is very sparse. However, it is very high quality because no automatic or manual transformations (such as stemming) have been done to alter it. While it is possible that such transformations enhance the representation, it is also possible that they introduce noise.

B. Stemmed Representation

Stemming is one of the most obvious ways to generalize (normalize) text. For this reason, stemming is commonly used in information retrieval systems as a rudimentary device to overcome the vocabulary mismatch problem. Various stemmers exist, including rule-based stemmers [11] and statistical stemmers [5]. Although stemming can significantly improve matching coverage, it also introduces noise, which can lead to poor matches. Using the Porter stemmer, both “marine vegetation” and “marinated vegetables” stem to “marin veget”, which is undesirable. Overall, however, the number of meaningful matches introduced typically outweighs the number of spurious matches. Throughout the remainder of this paper, we use the Porter stemmer to generate all of our stemmed representations.

C. Expanded Representation

Although stemming helps overcome the vocabulary mismatch problem to a certain extent, it does not handle the contextual problem. It fails to discern the difference between the meaning of "bank" in "Bank of America" and "river bank". Therefore, it is desirable to build representations for the short text segments that include contextually relevant information. One approach is to enrich the representation using an external source of information related to the query terms. Possible sources of such information include web (or other) search results returned by issuing the short text segment as a query, relevant Wikipedia articles, and, if the short text segment is a query, query reformulation logs. Each of these sources provides a set of contextual text that can be used to expand the original sparse text representation. In our experiments, we use web search results to expand our short text representations. For each short segment of text, we run the query against a commercial search engine's index and retrieve the top 200 results. The titles and snippets associated with these results are then concatenated and used as our expanded representation. In Figure 1, we show a portion of the expanded representation for the short text segment "apple pie". As we see, this expanded representation contains a number of contextually relevant terms, such as "recipe", "food", and "cooking" that are not present in the surface representation. We note that this expanded representation is similar to the one proposed in [11].

<query>***apple pie***</query>
<title>Applie pie – Wikipedia, the free encyclopedia</title>
<snippet>
In cooking, an apple pie is a fruit pie (or tart) in which the principal filling ingredient is apples . Pastry is generally used top-and-bottom, making a double-crust pie, the upper crust of which
...</snippet>
<url>en.wikipedia.org/wiki/Apple_pie</url>
<title>All About Food – Apple Pies</title>
<snippet>
Apple Pie. Recipes. All-American Apple Pie. American Apple Pie. Amish Apple Pie .Apple Cream Pie. Apple Crumble Pie. Apple Pie . Apple Pie in a Brown Bag. Best Apple Pie</snippet>
<url>fp.enter.net/~rburk/pies/applepie/applepie.htm</url>
<title>Apple Pie Recipe</title>
<snippet>Apple Pie Recipe using apple peeler corer slicer ... Apple Pie Recipe. From Scratch to Oven in 20-Minutes. Start by preheating the oven. By the time it's ...</snippet>
<url>applesource.com/applepierecipe.htm</url>

Fig.1. Example expanded representation for the text "apple pie."

IV. OVERVIEW OF THE PROPOSED METHOD

In this section we describe three methods for measuring the similarity between short segments of text. These measures are motivated by, and make use of, the representations described in the previous section. We also propose a hybrid method of combining the ranking of the various similarity measures in order to exploit the strengths and weaknesses of each.

A. LEXICAL

The most basic similarity measures are purely lexical. That is, they rely solely on matching the terms present in the surface representations. Given two short segments of text, *Q* and *C*, treating *Q* as the query and *C* as the candidate we wish to measure the similarity of, we define the following lexical matching criteria:

Exact – *Q* and *C* are lexically equivalent. (*Q*: "seattle mariners tickets", *C*:
"seattle mariners tickets")

Phrase – *C* is a substring of *Q*. (*Q*: "seattle mariners tickets", *C*: "seattle
mariners")

Subset – The terms in *C* are a subset of the terms in *Q*. (*Q*: "seattle mariners
tickets", *C*: "tickets seattle")

These measures are binary. That is, two segments of text either match (are deemed 'similar') or they do not. There is no graded score associated with the match. However, if necessary, it is possible to impose such a score by looking at various characteristics of the match such as the length of *Q* and *C,* or the frequency of the terms in some collection. Any candidate *C* that contains a term that does not appear in the query *Q* will not match under any of these rules, which is very undesirable.

Therefore, we expect that matches generated using these lexical rules will be have high precision but poor recall.

B. PROBABILISTIC

As we just described, lexical matching alone is not enough to produce a large number of relevant matches. In order to improve recall, we must make use of the expanded text representations as shown in Fig.1. To do so, we use the language modeling framework to model query and candidate texts. To utilize the framework, we must estimate unigram language models for the query (θ*Q*) and each candidate (θ*C*). For ranking purposes, we use the negative KL divergence between the query and candidate model, which is commonly used in the language modeling framework [14]. This results in the following ranking function:

$$-KL(\theta_Q,\theta_C) = H(\theta_Q) - CE(\theta_Q,\theta_C)$$
$$\equiv \sum_{w \in V} P(w \mid \theta_Q) \log P(w \mid \theta_C)$$

where *V* is the vocabulary, *H* is entropy, *CE* is cross entropy, and ≡ denotes rank equivalence. The critical part of the ranking function is how the query and candidate language models are estimated. Different estimates can lead to radically different rankings. We now describe how we estimate these models using the representations available to us.

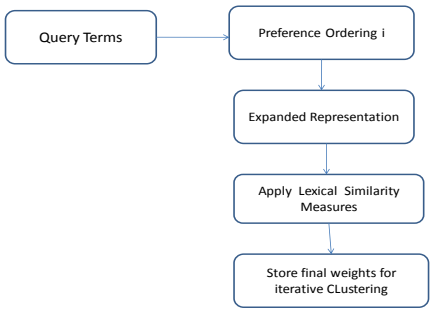

Fig.1. Proposed Architecture

A disadvantage of the above method is that it requires short text segments. A threshold on the similarity measure greatly affects the final clustering and therefore might impose a structure on the given data, instead of detecting any existing structure.

V. EXPERIMENTAL EVALUATION

In this section we evaluate the similarity measures proposed in Section 4. We begin by showing some illustrative examples of matches generated using our algorithms. We then formally evaluate the methods in the context of a query-query similarity task which captures all the characteristics before Clustering. Assuming that the documents already preprocessed and the vector representation exists, the word pair is converted into vector and processed with final weights obtained from an Probabilistic Similarity Measure. The outputs in the output layer are well interpreted whether the documents retrieved or clustered are relevant to the words in order to evaluate the quality of the implemented algorithms for document clustering. The results of the final weights with their similarity scores is shown in Figure 2.

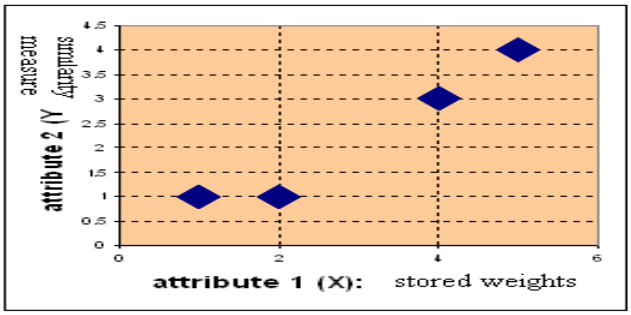

Fig.2. Similarity Measure with stored weights

We now describe our query-query similarity experiments. Here, we are interested in evaluating how well the various methods we described in Section 4 can be used to find queries that are similar to some target query. This task is a general task that is widely applicable. For example, such a query-query similarity system could be used to recommend alternative queries to users of a web search engine or for session boundary detection in query log analysis.

A. DATA DESCRIPTION

The following data resources were used in our experimental evaluation. A sample of 363,822 popular queries drawn from a 2005 MSN Search query log was used as our candidate pool of queries to match against. For each query, we generated an expanded representation, as described in Section 3.3. In our

experiments, we set μQ to 0 and μC to 2500. To handle this amount of data, we built an index out of the expanded representations using the Indri search system [14]. We also randomly sampled a set of 120 queries from the same log to use as target queries. These target queries were then matched against the full set of 363k queries. For each of these target queries, we ran the methods described in Section 4 and pooled the results down to a depth of 25 per method. A single human assessor then judged the relevance of each candidate result with respect to the target query using a 4-point judgment scale. Table 3 provides a description and examples of each type of judgment. The result of this assessment was 5231 judged target/candidate pairs. Of these judgments, 317 (6%) were Excellent, 600 (11%) were Good, 2537 (49%) were Fair, and 1777 (34%) were Bad. In order to determine the reliability of the judgments, four assessors judged 10 target queries. The inter-annotator agreement was then computed for these queries and was found to be 60%. However, when Excellent and Good judgments were binned and Fair and Bad judgments were binned, the agreement increased to 80%. This indicates the boundary between Fair and Bad is interpreted differently among users. For this reason, we will primarily focus our attention on the boundary between Excellent and Good and between Good and Fair. In addition, the Excellent and Good matches are the most interesting for many practical applications including query suggestion and sponsored search.

B EVALUATION

We are interested in understanding how our matching methods compare to each other across various relevance criteria. Since we are interested in using standard information retrieval metrics, such as precision and recall, we must binarize the relevance judgments. For each experiment, we state the relevance criteria used.

Judgment	***Description***
Excellent	The candidate is *semantically equivalent*
Good	The candidate is related to (but not identical to) the query intent and it is likely the *user would be interested in the candidate*.
Fair	The candidate is related to the query intent, but in an overly vague or specific manner that results in the *user having little, if any, interest in the candidate*.
Bad	The candidate is *unrelated* to the query intent.

Table.1 Judgment Description Examples (Query / Candidate)

The candidate is *semantically equivalent* to the user query.atlanta ga / atlanta Georgia - Good. The candidate is related to (but not identical to) the query intent and it is likely the *user would be interested in the candidate*. seattle mariners / seattle baseball tickets –Fair. The candidate is *related* to the query intent, but in an overly vague or specific manner that results in the user having little, if any, interest in the candidate. hyundia azera / new york car show – Bad The candidate is *unrelated* to the query intent.

We first evaluate the methods using precision-recall graphs using two different relevance criteria. The results are given in Figure 2. For the case when Excellent matches are considered relevant (left panel), we see that the Lexical and Stemming methods outperform the probabilistic methods, especially at lower recall levels. This is not surprising, since we expect lexical matches to easily find most of the Excellent matches. In addition, we see that Stemming consistently outperforms the Lexical method. However, the Back-off method dominates the other methods at all recall levels. This results from backing off from stricter matches to less strict matches. For example, for the query “atlanta ga”, the Lexical method will match “atlanta ga”, but neither the Lexical nor the Stemming methods will match “atlanta georgia", which is actually an Excellent match that is found using the Dense Prob. method. When we relax the relevance criteria and consider both Excellent and Good judgments to be relevant (right panel), we see an interesting shift in the graph. Here, the probabilistic methods, Sparse-Prob and Dense-Prob, outperform the Lexical and Stemming methods at all recall levels, except very low levels. We further test this hypothesis later in this section. However, once again, we see that the Back-off method outperforms all of the methods at all recall levels. One reason why the Back-off method is superior to the non-hybrid probabilistic methods is the fact that the Sparse-Prob and Dense-Prob

methods often fail to return exact matches high in the ranked list. This is caused by truncating the expanded query distribution before computing the KL divergence. By forcing the exact and exact stems matches to occur first, we are 'stacking the deck' and promoting matches that are likely to be high precision. This combined with the high recall of the Dense-Prob method, results in a superior matching method. It is clear that exact matches are very likely to result in excellent matches. However, it is not clear how phrase and subset lexical matches compare to stemming and probabilistic matches. To measure this, we compute the precision at *k* for the Lexical and Back-off methods, where *k* is the number of results returned by the query.

k	Queries	Lexical	Back-off
1	40	0.7500	0.8125
2	38	0.3235	0.4853
3	31	0.2688	0.4194

Table 2. Precision at k, where k is the number of matches returned using the Lexical method.

In this table, the evaluation set of queries was stratified according to *k*. Queries indicates the number of queries associated with each *k*. Only values of *k* associated with 10 or more. Table.2. Interpolated, 11-point precision-recall curves for the five matching methods described in Section 4. On the left, candidates judged 'Excellent' are considered relevant. On the right, candidates judged 'Excellent' or 'Good' are considered relevant. Lexical method. This evaluation allows us to quantify the improvement achieved by replacing the low precision phrase and subset matches with the high precision exact stems matches and high recall Dense Prob matches. We stratify the queries with respect to *k*, the number of Lexical method matches for the query, and compute precision at depth *k* over these queries. We only include values of *k* associated with 10 or more queries, since it misleading to compute and compare means over smaller samples. As the results show, the Back-off method is superior in every case. This suggests that the stemming and probabilistic matches (used in the Back-off method) are considerably better at finding both Excellent and Good matches compared to the phrase and subset matches (used in the Lexical method).

C *Effectiveness vs Efficiency*

One important practical aspect of the techniques developed is efficiency. Generating lexical and stemming matches is very efficient. The probabilistic methods are slower, but not unreasonable. Generating matches against our collection of 363,822 candidates using a modern single CPU machine takes 0.15 seconds per query using the Sparse-Prob method and 3 seconds per query using the Dense-Prob method. The Dense-Prob method requires, *apriori*, an index of expanded representations for both the candidates and the incoming queries. If we are asked to generate Dense-Prob matches for a query that is not in our index, then we must generate this representation on the fly. However, the Sparse-Prob method does not exhibit this behavior and can be used to efficiently generate matches for *any* incoming query. Therefore, Sparse-Prob is the best choice in terms of speed and coverage. However, if speed is not an issue, and high quality results are important, then Dense-Prob is the better choice.

VI.CONCLUSION

Web tasks such as query/keyword matching and search query suggestion rely heavily on the quality of similarity measures between short text segments. We consider two learning approaches: one directly models the similarity between a query and a suggestion (q, si) and the other models the preference ordering between two suggestions si and sj, with respect to the same query q. Finally, we present an experimental comparison between existing approaches for measuring similarity between short text segments and our enhanced similarity measures. The experiments indicate that our methods are significantly better than existing methods

ACKNOWLEDGMENT

I owe my sincere thanks to my Co-author and guide Dr. R.M. Suresh for his valuable mentoring and guidance in preparing this paper and helping me with valid suggestions and directing me in the correct path.

REFERENCES

[1] Berger, A. and Lafferty, J. Information retrieval as statistical translation. In Proceedings of SIGIR '99, pages 222-229, 1999.

[2] Cucerzan, S. and Brill, E. Extracting semantically related queries by exploiting user session information. Technical Report, Microsoft Research, 2005.

[3] Deerwester, S., Dumais, S., Landauer, T., Furnas, G. and Harshman, R. Indexing by latent semantic analysis. In JASIST, 41(6), pages 391-407, 1990.

[4] Jones, R. Generating query substitutions. In Proceedings of WWW 2006, pages 387-396,2006.

[5] Krovetz, R. Viewing morphology as an inference process. In Proceedings of SIGIR '93, pages 191-202, 1993.

[6] K.Selvi, R.M.Suresh "Context Similarity Measure Using Fuzzy Formal Concept Analysis" In Proc. of The Second Int'l conference On Computer Science and Engineering and Information Technology CCSEIT-2012, Pages 416-423,2012.

[7] Lavrenko, V. and Croft, W.B. Relevance based language models. In Proceedings of SIGIR '01, pages 120-127, 2001.

[8]. Ling Zhuang, Honghua Dai., 2004, A Maximal Frequent Item Set Approach for Web Document Clustering In Proceedings of the IEEE Fourth International Conference on Computer and Information Technology

[9] Metzler, D., Bernstein, Y., Croft, W.B., Moffat, A., and Zobel, J. Similarity measures for tracking information flow. In Proceedings of CIKM '05, pages 517-524, 2005.

[10] Murdock, V. and Croft, W.B. A Translation Model for Sentence Retrieval. In Proceedings of HLT/EMNLP '05, pages 684-691, 2005.

[11] Porter, M. F. An algorithm for suffix stripping. Program, 14(3), pages 130-137, 1980.

[12] Rocchio, J. J. Relevance Feedback in Information Retrieval, pages 313-323. Prentice-Hall, 1971.

[13] Sahami, M. and Heilman, T. A web-based kernel function for measuring the similarity of short text snippets. In Proceedings of WWW 2006, pages 377-386, 2006.

[14] Strohman, T., Metzler, D., Turtle, H., Croft, W. B. Indri: A language model-based search engine for complex queries. In Proceedings of the International Conference on Intelligence Analysis, 2005.

[15] Zhai, C. and Lafferty, J. A study of smoothing methods for language models applied to ad hoc information retrieval. In Proceedings of SIGIR '01, pages 334-342, 2001.

[16] Zhai, C. and Lafferty, J. Model-based feedback in the language modeling approach to information retrieval. In Proceedings of CIKM '01, pages 403-410, 2001

.

.