The 22nd International Conference on Machine Learning

7-11 August 2005 in Bonn, Germany

W 4

Proceedings of the Workshop on

Learning in Web Search (LWS 2005)

Stephan Bloehdorn

Wray Buntine

Andreas Hotho

ICML 2005
BONN - GERMANY

Learning in Web Search (LWS 2005)

International Workshop located at
the 22nd International Conference on Machine Learning (ICML 2005)
7th August 2005 - Bonn, Germany

Supported by

ALVIS, KnowledgeWeb, Network of Excellence Open Source Search, SEKT, PASCAL Network of Excellence and SmartWeb

Workshop Chairs:
Stephan Bloehdorn
Wray Buntine
Andreas Hotho

Learning in Web Search (LWS 2005)

The emerging world of search we see is one which makes increasing use of information extraction, gradually blends in semantic web technology and peer to peer systems, and uses grid computing as part of resources for information extraction and learning. This workshop aims at exploring the theory and application of machine learning in this context for the internet, intranets, the emerging semantic web and peer to peer search.

We are happy to see that this workshop succeeded in attracting a large number of high quality paper submissions, 8 of which were selected by the program committee. Besides this, three invited speakers have agreed to complement the paper presentations.

In his invited talk *Large Margin Methods in Information Extraction and Content Categorization*, Thomas Hofmann gives insights on using Support Vector Machines for predicting structured output variables. The papers *A Web-based kernel Function for Matching Short Text Snippets* and *A Semantic Kernel to classify Texts with very few Training Examples* also contribute to the field of kernel methods. In using formalized background knowledge, the latter seamlessly matches with the contribution *Learning Word-to-Concept Mappings for Automatic Text Classification*. The task of automated knowledge markup for the Semantic Web is addressed by means of machine learning methods in the paper *Unsupervised Ontology-based Semantic Tagging for Knowledge Markup*.

The invited talk *Generating Accurate Training Data from Implicit Feedback* by Thorsten Joachims moves the focus to the behavior of users in Web Search. The contribution *Topic-Specific Scoring of Documents for Relevant Retrieval* explores ways to differentiate and bias web search results with respect to topical preferences. In the paper *Evaluating the Robustness of Learning from Implicit Feedback*, the authors present a new approach for simulating user behavior in a web search setting.

In the third invited talk, *Type-enabled Keyword Searches with Uncertain Schema*, Soumen Chakrabarti gives insights into future Search paradigms that integrate more complex entity and relationship annotations with type-enabled queries. A short infrastructure contribution presents *Pipelets: A Framework for Distributed Computation*. Finally, the paper *Sailing the Web with Captain Nemo: a Personalized Metasearch Engine* presents the implementation of a metasearch engine that exploits personal user search spaces.

We thank the members of our program committee for their efforts to ensure the quality of accepted papers. We kindly acknowledge the research projects that are supporting this workshop. We are looking forward to having interesting presentations and fruitful discussions.

August 2005

Your LWS2005 Team

Stephan Bloehdorn, *Wray Buntine* and *Andreas Hotho*

Workshop Chairs

Stephan Bloehdorn
University of Karlsruhe
Institute AIFB, Knowledge Management Research Group
D–76128 Karlsruhe, Germany
http://www.aifb.uni-karlsruhe.de/WBS/sbl
sbl@aifb.uni-karlsruhe.de

Wray Buntine
Helsinki Institute of Information Technology
Complex Systems Computation Group
FIN–00180 Helsinki, Finland
http://www.hiit.fi/u/buntine/
wray.buntine@hiit.fi

Andreas Hotho
University of Kassel
Knowledge and Data Engineering Group
D–34121 Kassel, Germany
http://www.kde.cs.uni-kassel.de/hotho
hotho@cs.uni-kassel.de

Program Committee

Further Reviewers

Jose Iria
University of Sheffield

Table of Contents

Large Margin Methods in Information Extraction and Content Categorization (Invited Talk)

Thomas Hofmann

Technical University of Darmstadt, Intelligent Systems Group, and
Fraunhofer Institute for Integrated Publication and Information Systems (IPSI),
D–64293 Darmstadt, Germany

Abstract: Support Vector Machines (SVMs) have been one of the major breakthroughs in machine learning, both in terms of their practical success as well as their learning-theoretic properties. This talk presents a generic extension of SVM classification to the case of structured classification, i.e. the task of predicting output variables with some meaningful internal structure. As we will show, this approach has many interesting applications in information extraction, information retrieval, document categorization and natural language processing, including supervised training of Markov Random Fields and probabilistic context-free grammars.

A Web-based Kernel Function for Matching Short Text Snippets

Mehran Sahami SAHAMI@GOOGLE.COM
Tim Heilman TDH@GOOGLE.COM
Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043 USA

Abstract

Determining the similarity of short text snippets, such as search queries, works poorly with traditional document similarity measures (e.g., cosine), since there are often few, if any, terms in common between two short text snippets. We address this problem by introducing a novel method for measuring the similarity between short text snippets (even those without any overlapping terms) by leveraging web search results to provide greater context for the short texts. In this paper, we define such a similarity kernel function and provide examples of its efficacy. We also show the use of this kernel function in a large-scale system for suggesting related queries to search engine users.

1. Introduction

In analyzing text, there are many situations in which we wish to determine how similar two short text snippets are. For example, there may be different ways to describe some concept or individual, such as "United Nations Secretary-General" and "Kofi Annan", and we would like to determine that there is a high degree of *semantic* similarity between these two text snippets. Similarly, the snippets "AI" and "Artificial Intelligence" are very similar with regard to their meaning, even though they may not share any actual terms in common.

Applying traditional document similarity measures, such as the widely used cosine coefficient (Salton et al., 1975; Salton & McGill, 1983), to such short text snippets often produces inadequate results, however. Indeed, in both the examples given previously, applying the cosine would yield a similarity of 0 since each given text pair contains no common terms. Even in cases where two snippets may share terms, they may be using the term in different contexts. Consider the snippets "graphical models" and "graphical interface". The first uses *graphical* in reference to graph structures whereas the second uses the term to refer to graphic displays. Thus, while the cosine score between these two snippets would be 0.5 due to the shared lexical term "graphical", at a semantic level the use of this shared term is not truly an indication of similarity between the snippets.

To address this problem, we would like to have a method for measuring the similarity between such short text snippets that captures more of the semantic context of the snippets rather than simply measuring their term-wise similarity. To help us achieve this goal, we can leverage the large volume of documents on the web to determine greater context for a short text snippet. By examining documents that contain the text snippet terms we can discover other contextual terms that help to provide a greater context for the original snippet and potentially resolve ambiguity in the use of terms with multiple meanings.

Our approach to this problem is relatively simple, but surprisingly quite powerful. We simply treat each snippet as a query to a web search engine in order to find a number of documents that contain the terms in the original snippets. We then use these returned documents to create a *context vector* for the original snippet, where such a context vector contains many words that tend to occur in context with the original snippet (i.e., query) terms. Such context vectors can now be much more robustly compared with a measure such as the cosine to determine the similarity between the original text snippets. Furthermore, since the cosine is a valid kernel, using this function in conjunction with the generated context vectors makes this similarity function applicable in any kernel-based learning algorithm where (short) text data is being processed.

While there are many cases where getting a robust measure of similarity between short texts is important,

Appearing in *W4: Learning in Web Search, at the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

one particularly useful application in the context of search is to suggest related queries to a user. In such an application, a user who issues a query to a search engine may find it helpful to be provided with a list of semantically related queries that he or she may consider to further explore the related information space. By employing our short text similarity kernel, we could match the user's initial query against a large repository of existing user queries to determine other similar queries to suggest to the user. Thus, the results of the similarity function can be directly employed in an end-user application.

The approach we take in constructing our similarity function has relations to previous work in both the Information Retrieval and Machine Learning communities. We explore these relations and put our work in the context of previous research in Section 2. We then formally define our similarity function in Section 3 and present examples of the results in Section 4. In Section 5 we present a system for related query suggestion using our similarity function, and then present its evaluation in Section 6. Finally, in Section 7 we provide some conclusions and directions for future work.

2. Related Work

The similarity function we present here is based on *query expansion* techniques (Buckley et al., 1994; Mitra et al., 1998) which have long been used in the Information Retrieval community. Such methods automatically augment a user query with additional terms based on documents that are retrieved in response to the initial user query or by using an available thesaurus. Our motivation for and usage of query expansion greatly differs from this previous work, however. First, the traditional goal of query expansion has been to improve recall (potentially at the expense of precision) in a retrieval task. Our focus, however, is on using such expansions to provide a richer representation for a short text in order to potentially compare it robustly with other short texts. Moreover, traditional expansion is focused on creating a new query for retrieval rather than doing pair-wise comparisons between short texts. Thus, the approach we take is quite different than the use of query expansion in a standard Information Retrieval context.

Alternatively, information retrieval researchers have previously proposed other means of determining query similarity. One early method proposed by Raghavan and Sever (Raghavan & Sever, 1995) attempts to measure the relatedness of two queries by determining differences in the ordering of documents retrieved in response to the two queries. This method requires a total ordering (ranking) of documents over the whole collection for each query. Thus, comparing the pairwise differences in rankings requires $O(N^2)$ time, where N is the number of documents in the collection. In the context of the web, where $N > 8$ billion, this algorithm quickly becomes intractable.

Later work by Fitzpatrick and Dent (Fitzpatrick & Dent, 1997) measures query similarity using the normalized set overlap (intersection) of the top 200 documents retrieved for each query. While this algorithm's runtime complexity easily scales to the web, it will likely not lead to very meaningful similarity results as the sheer number of documents in the web collection will often make the set overlap for returned results extremely small (or empty) for many related queries that are not nearly identical. We show that this is indeed the case in our experimental results later in the paper.

In the context of Machine Learning, there has been a great deal of work in using kernel methods, such as Support Vector Machines for text classification (Joachims, 1998; Dumais et al., 1998). Such work has recently extended to building specialized kernels aimed at measuring semantic similarity between documents. We outline some of these approaches below, and show how they differ from the work presented here.

One of the early approaches in this vein is *Latent Semantic Kernels* (Cristianini et al., 2002), which is a kernel-based extension to Latent Semantic Indexing (Deerwester et al., 1990). Here a kernel matrix is computed over text documents, and the eigen-decomposition of this matrix is used to compute a new (lower rank approximation) basis for the space. The dimensions of the new basis can intuitively be thought of as capturing "semantic concepts" (i.e., roughly corresponding to co-varying subsets of the dimensions in the original space). While there may be some superficial similarities, this approach differs in fundamental respects from our work. First, our method is aimed at constructing a new kernel function, not using an existing kernel matrix to infer "semantic dimensions". Also, our method takes a *lazy* approach in the sense that we need not compute an expansion for a given text snippet until we want to evaluate the kernel function. We never need to explicitly compute a full kernel matrix over some set of existing text snippets nor its eigen-decomposition. Indeed, the kernel we present here is entire complimentary to work on Latent Semantic Kernels, as our kernel could be used to construct the kernel matrix on which the eigen-decomposition is performed.

An approach more akin to that taken here is the work of Kandola *et al.* (Kandola et al., 2002) who define

a kernel for determining the similarity of individual terms based on the collection of documents that these terms appear in. In their work, they learn a *Semantic Proximity Matrix* that captures the relatedness of individual terms by essentially measuring the correlation in the documents that contain these terms. In our work, the kernel we consider is not attempting to just determine similarity between single terms, but entire text snippets. Moreover, our approach does not require performing an optimization over an entire collection of documents (as is required in the previous work), but rather the kernel between snippets can be computed on-line selectively, as needed.

Previous research has also tried to address learning a semantic representation for a document by using cross-lingual techniques (Vinokourov et al., 2002). Here, one starts with a corpus of document pairs, where each pair is the same document written in two different languages. A correlation analysis is then performed between the corpora in each language to determine combinations of related words in one language that correlate well with combinations of words in the other language, and thereby learn word relations within a given language. Obviously, the approach we take does not require such a paired corpora. And, again, we seek to not just learn relationships between single terms but between entire arbitrary short texts.

Thus, while there has been a good deal of work in determining semantic similarities between texts (which highlights the general importance of this problem), many of which use kernel methods, the approach we present has significant differences with that work. Moreover, our approach provides the compelling advantage that semantic similarity can be measured between multi-term short texts, where the entire text can be considered as a whole, rather than just determining similarity between individual terms. Furthermore, no expensive pre-processing of a corpora is required (e.g., eigen-decomposition), and the kernel can easily be computed for a given snippet pair as needed. We simply require access to a search engine (i.e., text index) over a corpora, which can be quite efficiently (linearly) constructed or can be obviated entirely by accessing a public search engine on the Web, such as the Google API (`http://www.google.com/apis`).

3. A New Similarity Function

Presently, we formalize our kernel function for semantic similarity. Let x represent a short text snippet[1]. Now, we compute the *query expansion* of x, denoted $QE(x)$, as follows:

1. Issue x as a query to a search engine S.
2. Let $R(x)$ be the set of (at most) n retrieved documents $d_1, d_2, \ldots, d_n$
3. Compute the TFIDF term vector v_i for each document $d_i \in R(x)$
4. Truncate each vector v_i to include its m highest weighted terms
5. Let C be the centroid of the L_2 normalized vectors v_i:
$$C = \frac{1}{n} \sum_{i=1}^{n} \frac{v_i}{\|v_i\|_2}$$
6. Let $QE(x)$ be the L_2 normalized centroid of C:
$$QE(x) = \frac{C}{\|C\|_2}$$

We note that to be precise, the computation of $QE(x)$ really should be parameterized by both the query x and the search engine S used. Since we assume that S remains constant in all computations, we omit this parameter for brevity.

There are several modifications that can be made to the above procedure, as appropriate for different document collections. Foremost among these is the term weighting scheme used in Step 3. Here, we consider a TFIDF vector weighting scheme (Salton & Buckley, 1988), where the weight $w_{i,j}$ associated with with term t_i in document d_j is defined to be:

$$w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i}),$$

where $tf_{i,j}$ is the frequency of t_i in d_j, N is the total number of documents in the corpus, and df_i is the total number of documents that contain t_i. Clearly, other weighting schemes are possible, but we choose TFIDF here since it is commonly used in the IR community and we have found it to empirically give good results in building representative query expansions. Also, in Step 4, we set the maximum number of terms in each vector $m = 50$, as we have found this value to give a good trade-off between representational robustness and efficiency.

Also, in Step 2, we need not choose to use the entirety of retrieved documents in order to produce vectors. We may choose to limit ourselves to create vectors using just the *contextually descriptive text snippet* for each document that is commonly generated by Web search engines. This would make our algorithm more efficient in terms of the amount of data processed, and allows us to make ready use of the results from public web search engines without having to even retrieve the full actual underlying documents. Of course, there remains the question of how large such descriptive texts

[1] While the real focus of our work is geared toward short text snippets, there is no technical reason why x must have limited length, and in fact x can be arbitrary text.

provided by search engines need to be in order to be particularly useful. Empirically, we have found that using 1000 characters (in a token delimited window centered on the original query terms in the original text) is sufficient to get accurate results, and increasing this number does not seem to provide much additional benefit.

Evaluating a variety of term weighting or text windowing schemes, however, is not the aim of this work and we do not explore it further here. Rather we simply seek to outline some of the issues that may be of interest to practitioners and provide some guidance on reasonable values to use that we have found work well empirically.

Finally, given that we have a means for computing the query expansion for a short text, it is a simple matter to define the semantic kernel function K as the inner product of the query expansions for two text snippets. More formally, given two short text snippets x and y, we define the semantic similarity kernel between them as:

$$K(x, y) = QE(x) \cdot QE(y).$$

Clearly, since $K(x, y)$ is an inner product with a bounded norm (given that each query expansion vector has norm 1.0), it is a valid kernel function.

4. Examples of Results With Kernel

To get a cursory evaluation for how well our semantic similarity kernel performs, we show results with the kernel on a number of text pairs, using the Google search engine as the underlying document retrieval mechanism. We attempt to highlight both the strengths and potential weaknesses of this kernel function.

We examined several text snippet pairs to determine the similarity score given by our new web-based kernel, the traditional cosine measure, and the set overlap measure proposed by Fitzpatrick and Dent. We specifically look at three genres of text snippet matching: (i) acronyms, (ii) individuals and their positions, and (iii) multi-faceted terms.[2] Examples of applying the kernel are shown in Table 1, which is segmented by the genre of matching examined.

[2]We prefer the term *multi-faceted* over *ambiguous*, since multi-faceted terms may have the same definition in two contexts, but the accepted semantics of that definition may vary in context. For example, the term "travel" has the same definition in both the phrases "space travel" and "vacation travel", so it is (strictly speaking) not ambiguous here, but the semantics of what is meant by *traveling* in those two cases is different.

The first section of the table deals with the identification of acronyms. In this genre, we find two notable effects using our kernel. First, from the relatively high similarity scores found between acronyms and their full name, it appears that our kernel is generally effective at capturing the semantic similarity between an acronym and its full name. Note that the kernel scores are not 1.0 since acronyms can often have multiple meanings. Related to this point, our second observation is that our kernel function (being based on contextual text usage on the web) tends to prefer more common usages of an acronym in determining semantic similarity. For example, the text "AI" is determined to be much more similar to "artificial intelligence" than "artificial insemination" (even though it is a valid acronym for both), since contextual usage of "AI" on the web tends to favor the former meaning. We see a similar effect when comparing "term frequency inverse document frequency" to "tf idf" and "tfidf". While the former acronym tends to be more commonly used (especially since the sub-acronyms "tf" and "idf" are separated), the still reasonable score over 0.5 for the acronym "tfidf" shows that the kernel function is still able to determine a solid level of semantic similarity. It is not surprising that the use of cosine similarity is entirely inappropriate for such a task (since the full name of an acronym virtually never contains the acronym itself). Moreover, we find, as expected, that the set overlap measure leads to very low (and not very robust) similarity values.

Next, we examined the use of our kernel in identifying different characterizations of individuals. Specifically, we considered determining the similarity of the name of a notable individual with his prominent role description. The results of these examples are shown in the second section of Table 1.

In order to assess the strengths and weakness of the kernel function we intentionally applied the kernel to both correct pairs of descriptions and individuals as well looking at pairs involving an individual and a *close*, but incorrect, description. For example, while Kofi Annan and George W. Bush are both prominent world political figures, the kernel is effective at determining the correct role matches and assigning them appropriately high scores.

In the realm of business figures, we find that the kernel is able to distinguish Steve Ballmer as the current CEO of Microsoft (and not Bill Gates). Bill Gates still gets a non-trivial semantic similarity with the role "Microsoft CEO" since he was indeed the *former* CEO, but he is much more strongly (by a over a factor of 2) associated correctly with the text "Microsoft founder". Sim-

Text 1	Text 2	Kernel	Cosine	Set Overlap
Acronyms				
support vector machine	SVM	0.812	0.0	0.110
International Conference on Machine Learning	ICML	0.762	0.0	0.085
portable document format	PDF	0.732	0.0	0.060
artificial intelligence	AI	0.831	0.0	0.255
artificial insemination	AI	0.391	0.0	0.000
term frequency inverse document frequency	tf idf	0.831	0.0	0.125
term frequency inverse document frequency	tfidf	0.507	0.0	0.060
Individuals and their positions				
UN Secretary-General	Kofi Annan	0.825	0.0	0.065
UN Secretary-General	George W. Bush	0.110	0.0	0.000
US President	George W. Bush	0.688	0.0	0.045
Microsoft CEO	Steve Ballmer	0.838	0.0	0.090
Microsoft CEO	Bill Gates	0.317	0.0	0.000
Microsoft Founder	Bill Gates	0.677	0.0	0.010
Google CEO	Eric Schmidt	0.845	0.0	0.105
Google CEO	Larry Page	0.450	0.0	0.040
Google Founder	Larry Page	0.770	0.0	0.050
Microsoft Founder	Larry Page	0.189	0.0	0.000
Google Founder	Bill Gates	0.096	0.0	0.000
web page	Larry Page	0.123	0.5	0.000
Multi-faceted terms				
space exploration	NASA	0.691	0.0	0.070
space exploration	space travel	0.592	0.5	0.005
vacation travel	space travel	0.321	0.5	0.000
machine learning	ICML	0.586	0.0	0.065
machine learning	machine tooling	0.197	0.5	0.000
graphical UI	graphical models	0.275	0.5	0.000
graphical UI	graphical interface	0.643	0.5	0.000
java island	Indonesia	0.454	0.0	0.000
java programming	Indonesia	0.020	0.0	0.000
java programming	applet development	0.563	0.0	0.010
java island	java programming	0.280	0.5	0.000

Table 1. Examples of web-based kernel applied to short text snippet pairs.

ilarly, the kernel is successful at correctly identifying the current Google CEO (Eric Schmidt) from Larry Page (Google's founder and *former* CEO).

We also attempted to test how easily the kernel function gave back high scores for inappropriate matches by trying to pair Bill Gates as the founder of Google and Larry Page as the founder of Microsoft. The low similarity scores given by the kernel show that it does indeed find little semantic similarity between these inappropriate pairs. Once again, the kernel value is nonzero since each of the individuals is indeed the founder of *some* company, so the texts compared are not entirely devoid of some semantic similarity. Finally, we show that even though Larry Page has a very common surname, the kernel does a good job of not confusing him with a "web page" (although the cosine gives a inappropriately high similarity due to the match on the term "page").

Lastly, we examined the efficacy of the kernel when applied to texts with multi-faceted terms – a case where we expect the raw cosine and set overlap to once again do quite poorly. As expected, the kernel does a reasonable job of determining the different facets of terms, such as identifying "space exploration" with "NASA" (even though they share no tokens), but finding that the similarity between "vacation travel" and "space travel" is indeed less than the cosine might otherwise lead us to believe. Similar effects are seen in looking at terms used in context, such as "machine", "graphical", and "java". We note that in many cases, the

similarity values here are not as extreme as in the previous instances. This has to do with the fact that we are trying to measure the rather fuzzy notion of *aboutness* between semantic concepts rather than trying to identify an acronym or individual (which tend to be much more specific matches). Still, the kernel does a respectable job (in most cases) of providing a score above 0.5 when two concepts are very related and less than 0.3 when the concepts are generally thought of as distinct.

Once again, the low similarity scores given by the set overlap method show that in the context of a large document collection such as the web, this measure is not very robust. As a side note, we also measured the set overlap using the top 500 and top 1000 documents retrieved for each query (in addition to the results reported here which looked at the top 200 documents as suggested in the original paper), and found qualitatively very similar results thus indicating that the method itself, and not merely the parameter settings, led to the poor results in the context of the web.

5. Related Query Suggestion

Armed with promising anecdotal results that argue in favor of using this kernel when comparing short texts, we turn our attention to the task of developing a simple application based on this kernel. The application we choose is query suggestion—that is, to suggest potentially related queries to the users of a search engine to give them additional options for information finding. We note that there is a long history of work in query refinement, including the previously mentioned work in query expansion (Buckley et al., 1994; Mitra et al., 1998), harnessing relevance feedback for query modification (Harman, 1992), using pre-computed term similarities for suggestions (Vlez et al., 1997), linguistically mining documents retrieved in response to a search for related terms and phrases (Xu & Croft, 1996; Anick & Tipirneni, 1999), and even simply finding related queries in a thesaurus. While this is certainly an active area of work in information retrieval, we note that improving query suggestion is not the focus of this work. Thus, we intentionally do not compare our system with others. Rather, we use query suggestion as a means of showing the potential utility of our kernel function in just one, of potentially many, real-world applications. We provide a user evaluation of the results in this application to get a more objective measure of the efficacy of our kernel.

At a high-level, our query expansion system can be described as starting with an initial repository Q of previously issued user queries (for example, culled from search engine logs). Now, for any newly issued user query u, we can compute our kernel function $K(u, q_i)$ for all $q_i \in Q$ and suggest related queries q_i which have the highest kernel score with u (subject to some post-filtering to eliminate related queries that are too linguistically similar to each other).

More specifically, we begin by pre-computing the query expansions for a repository Q of approximately 116 million popular user queries issued in 2003, determined by sampling anonymized web search logs from the Google search engine. After generating these query expansions, we index the resulting vectors for fast retrieval in a retrieval system R. Now, for any newly observed user query u, we can generate its query expansion $QE(u)$ and use this entire expansion as a disjunctive query to R, finding all existing query expansions $QE(q_i)$ in the repository that potentially match $QE(u)$. Note that if a query expansion $QE(q)$ indexed in R does not match $QE(u)$ in at least one term (i.e., it is not retrieved), then we know $K(u, q) = 0$ since there are no common terms in $QE(u)$ and $QE(q)$. For each retrieved query expansion $QE(q_i)$, we can then compute the inner product $QE(u) \cdot QE(q_i) = K(u, q_i)$.

To actually determine which of the matched queries from the repository to suggest to the user, we use the following algorithm, where the constant MAX is set to the maximum number of suggestions that we would like to obtain:

Given: user query u, and
list of matched queries from repository
Output: list Z of queries to suggest
1. Initialize suggestion list $Z = \emptyset$
2. Sort kernel scores $K(u, q_i)$ in descending order to produce an ordered list $L = (q_1, q_2, \ldots, q_k)$ of corresponding queries q_i.
3. $j = 1$
4. **While** ($j \leq k$ and size(Z) $<$ MAX) **do**
4.1 **If** ($|q_j| - |q_j \cap z| > 0.5|z| \quad \forall z \in (Z \cup u)$) **then**
4.1.1 $Z = Z \cup q_j$
4.2 $j = j + 1$
5. **Return** suggestion list Z

Here $|q|$ denotes the number of terms in query q. Thus, the test in Step 4.1 above is our post-filter to only add another suggested query q_j if it differs by more than half as many terms from any other query already in the suggestion list Z (as well as the original user query u). This helps promote linguistic diversity in the set of suggested queries. The outputted list of suggestions Z can be presented to the search engine user to guide them in conducting follow-up searches.

6. Evaluation of Query Suggestions

In order to evaluate our kernel within the context of this query suggestion system, we enlisted nine human raters who are computer scientists familiar with information retrieval technologies. Each rater was asked to issue queries from the Google Zeitgeist[3] in a different month of 2003 (since our initial repository of queries to suggest was culled near the start of 2003). The Google Zeitgeist tracks popular queries on the web monthly. We chose to use such common queries for evaluation because if useful suggestions were found, they could potentially be applicable for a large number of search engine users who had the same information needs.

Each rater evaluated the suggested queries provided by the system on a 5-point Likert scale, defined as:

1: suggestion is totally off topic.
2: suggestion is not as good as original query.
3: suggestion is basically same as original query.
4: suggestion is potentially better than original query.
5: suggestion is fantastic – should suggest this query since it might help a user find what they're looking for if they issued it instead of the original query.

In our experiment we set the maximum number of suggestions for each query (MAX) to 5, although some queries yielded fewer than this number of suggestions due to having fewer suggestions pass the post-filtering process. A total of 118 user queries, which yielded 379 suggested queries (an average of 3.2 suggestions per query) were rated. Note that some raters evaluated a different number of queries than other raters.

Since each query suggestion has a kernel score associated with it, we can determine how suggestion quality is correlated with the kernel score by looking at the average rating over all suggestions that had a kernel score above a given threshold. If the kernel is effective, we would generally expect higher kernel scores to lead to more useful queries suggested to the user (as they would tend to be more on-topic even given the post-filtering mechanism that attempts to promote diversity among the query suggestions). Moreover, we would expect that overall the suggestions would often be rated close to 3 (or higher) if the kernel were effective at identifying query suggestions semantically similar to the original query.

The results of this experiment are shown in Figure 1, which shows the average user rating for query suggestions, where we use a kernel score threshold to only consider suggestions that scored at that threshold or higher with the original query. Indeed, we see that

[3] www.google.com/intl/en/press/zeitgeist.html

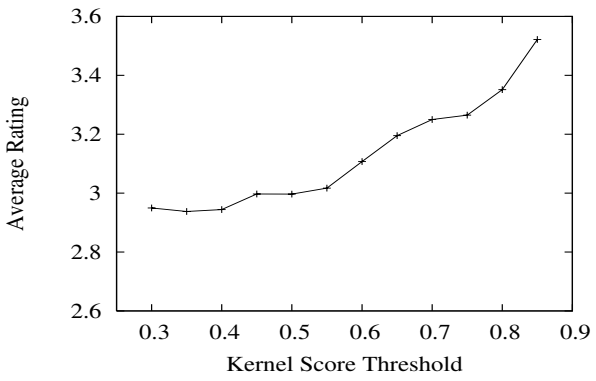

Figure 1. Average ratings at various kernel thresholds.

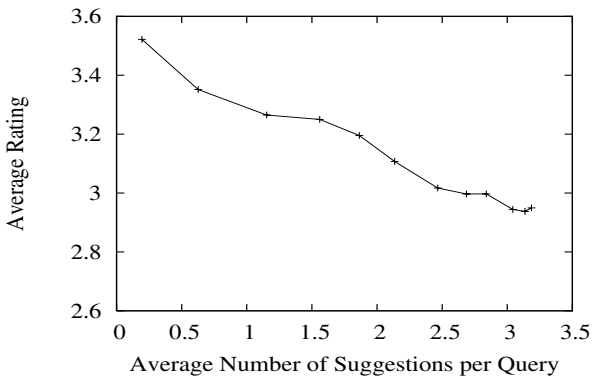

Figure 2. Average ratings versus average number of query suggestions made for each query.

the query suggestions are generally rated close to 3 (same as the original query), but that the rating tends to increase with the kernel score. This indicates that queries deemed by the kernel to be very related to the original query are quite useful to users in honing their information need, especially when we allow for some diversity in the results using the post-filtering mechanism. In fact, we found that without the use of the post-filtering mechanism, the results suggested by the system were often too similar to the original query to provide much additional utility for query suggestion (although it was indicative of the kernel being effective at finding related queries).

Figure 2 shows a graph analogous to a Precision-Recall curve, where we plot the average user rating for query suggestions versus the average number of suggestions that are given per query as we vary the kernel score threshold from 0.85 down to 0.3. We see a clear trade-off between the quality of the suggestions presented to the user and the number of suggestions given. Indeed, it is possible, on average to give two query suggestions for each query which have a quality (slightly) higher

than the original query.

7. Conclusions and Future Work

We have presented a new kernel function for measuring the semantic similarity between pairs of short text snippets. We have shown, both anecdotally and in a human-evaluated query suggestion system, that this kernel is an effective measure of similarity for short texts, and works well even when the short texts being considered have no common terms.

The are several lines of future work that this kernel lays the foundation for. The first is improvements in the generation of query expansions with the goal of improving the match score for the kernel function. The second is the incorporation of this kernel into other kernel-based learning schemes to determine its ability to provide improvement in tasks such as classification and clustering of text.

Also, there are certainly other applications, besides query suggestion, that could be considered as well. One such application is in a question answering system, where the question could be matched against a list of candidate answers to determine which is the most similar semantically. For example, using our kernel we find that: K("Who shot Abraham Lincoln", "John Wilkes Booth") = 0.730. Thus, the kernel does well in giving a high score to the correct answer to the question, even though it shares no terms in common with the question. Alternatively, K("Who shot Abraham Lincoln", "Abraham Lincoln") = 0.597, indicating that while the question is certainly semantically related to "Abraham Lincoln", the true answer to the question is in fact more semantically related to the question. Finally, we note that this kernel is not limited to being used on the web, and can also be computed using query expansions generated over domain-specific corpora in order to better capture contextual semantics in certain domains. We hope to explore such research venues in the future.

Acknowledgments

We thank Amit Singhal for many invaluable discussions related to this research. We also thank the anonymous reviewers for their thoughtful comments and pointers to related work.

References

Anick, P., & Tipirneni, S. (1999). The paraphrase search assistant: Terminological feedback for iterative information seeking. *Proceedings of the 22nd Annual SIGIR Conference* (pp. 153–159).

Buckley, C., Salton, G., Allan, J., & Singhal, A. (1994). Automatic query expansion using SMART: TREC 3. *The Third Text REtrieval Conference* (pp. 69–80).

Cristianini, N., Shawe-Taylor, J., & Lodhi, H. (2002). Latent semantic kernels. *Journal of Intelligent Information Systems*, *18*, 127–152.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*, 391–407.

Dumais, S. T., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. *CIKM-98: Proceedings of the Seventh International Conference on Information and Knowledge Management.*

Fitzpatrick, L., & Dent, M. (1997). Automatic feedback using past queries: Social searching? *Proceedings of the 20th Annual SIGIR Conference* (pp. 306–313).

Harman, D. (1992). Relevance feedback and other query modification techniques. In W. B. Frakes and R. Baeza-Yates (Eds.), *Information retrieval: Data structures and algorithms*, 241–263. Prentice Hall.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. *Proceedings of ECML-98, 10th European Conference on Machine Learning* (pp. 137–142).

Kandola, J. S., Shawe-Taylor, J., & Cristianini, N. (2002). Learning semantic similarity. *Advances in Neural Information Processing Systems (NIPS) 15* (pp. 657–664).

Mitra, M., Singhal, A., & Buckley, C. (1998). Improving automatic query expansion. *Proceedings of the 21st Annual SIGIR Conference* (pp. 206–214).

Raghavan, V. V., & Sever, H. (1995). On the reuse of past optimal queries. *Proceedings of the 18th Annual SIGIR Conference* (pp. 344–350).

Salton, G., & Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management*, *24*, 513–523.

Salton, G., & McGill, M. J. (1983). *Introduction to modern information retrieval.* McGraw-Hill Book Company.

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*, 613–620.

Vinokourov, A., Shawe-Taylor, J., & Cristianini, N. (2002). Inferring a semantic representation of text via cross-language correlation analysis. *Advances in Neural Information Processing Systems (NIPS) 15* (pp. 1473–1480).

Vlez, B., Wiess, R., Sheldon, M. A., & Gifford, D. K. (1997). Fast and effective query refinement. *Proceedings of the 20th Annual SIGIR Conference* (pp. 6–15).

Xu, J., & Croft, W. B. (1996). Query expansion using local and global document analysis. *Proceedings of the 19th Annual SIGIR Conference* (pp. 4–11).

A Semantic Kernel to classify texts with very few training examples

Roberto Basili BASILI@INFO.UNIROMA2.IT
Marco Cammisa CAMMISA@INFO.UNIROMA2.IT
Alessandro Moschitti MOSCHITTI@INFO.UNIROMA2.IT
Department of Computer Science, Systems and Production,
University of Rome "Tor Vergata",
Via del Politecnico 1, 00133 Rome, Italy

Abstract

Web-mediated access to distributed information is a complex problem. Before any learning can start, Web objects (e.g. texts) have to be detected and filtered accurately. In this perspective, text categorization is a useful device to filter out irrelevant evidence before other learning processes take place on huge sources of candidate information. The drawback is the need of a large number of training documents. One way to reduce such number relates to the use of more effective document similarities based on prior knowledge. Unfortunately, previous work has shown that such information (e.g. WordNet) causes the decrease of retrieval accuracy.

In this paper we propose kernel functions to add prior knowledge to learning algorithms for document classification. Such kernels use a term similarity measure based on the WordNet hierarchy. The kernel trick is used to implement such space in a balanced and statistically coherent way. Cross-validation results show the benefit of the approach for the Support Vector Machines when few training examples are available.

1. Introduction

Web-mediated access to distributed information is a complex problem. Before any learning can start, Web objects (e.g. texts) have to be detected and filtered accurately. In this perspective, text categorization (TC) is a useful device to filter out irrelevant evidence before other learning processes take place on huge sources of candidate information. To apply TC in Web search, methods based on small number of examples should be preferred. As such number decreases the classification accuracy decreases as well, thus, to mitigate this problem, most of the research efforts have been directed in enriching the document representation by using term clustering (*term generalization*) or adding compound terms (*term specification*). These studies are based on the assumption that the similarity between two documents can be expressed as the similarity between pairs of matching terms. Following this idea, term clustering methods based on corpus term distributions or on external (to the target corpus) prior knowledge (e.g. provided by WordNet) were used to improve the basic term matching.

An example of statistical clustering is given in (Bekkerman et al., 2001). A feature selection technique, which clusters similar features/words, called the Information Bottleneck (IB), was applied to Text Categorization (TC). Such cluster based representation outperformed the simple *bag-of-words* on only one out of the three experimented collections. The effective use of external prior knowledge is even more difficult since no attempt has ever been successful to improve document retrieval or text classification accuracy, (e.g. see (Smeaton, 1999; Sussna, 1993; Voorhees, 1993; Voorhees, 1994; Moschitti & Basili, 2004)).

The main problem of term cluster based representations seems the unclear nature of the relationship between the word and the cluster information levels. Although (semantic) clusters tend to improve the system Recall, simple terms are, on a large scale, more accurate (e.g. (Moschitti & Basili, 2004)). To overcome this problem the hybrid spaces containing terms and clusters were experimented (e.g. (Scott & Matwin, 1999)) but the results, again, showed that the mixed statistical distributions of clusters and terms impact

Appearing in *W4: Learning in Web Search, at the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

either marginally or even negatively on the overall accuracy.

In (Voorhees, 1993; Smeaton, 1999), clusters of synonymous terms as defined in WordNet (WN) (Fellbaum, 1998) were used for document retrieval. The results showed that the misleading information due to the wrong choice of the local term senses causes the overall accuracy to decrease. Word sense disambiguation (WSD) was thus applied beforehand by indexing the documents by means of disambiguated senses, i.e. synset codes (Smeaton, 1999; Sussna, 1993; Voorhees, 1993; Voorhees, 1994; Moschitti & Basili, 2004). However, even the state-of-the-art methods for WSD did not improve the accuracy because of the inherent noise introduced by the disambiguation mistakes. The above studies suggest that term clusters decrease the precision of the system as they force weakly related terms or unrelated terms (in case of disambiguation errors) to give a contribution in the similarity function. The successful introduction of prior external knowledge relies on the solution of the above problem.

In this paper, a model to introduce the semantic lexical knowledge contained in the WN hierarchy in a supervised text classification task has been proposed. Intuitively, the main idea is that the documents d are represented through the set of all pairs $< t, t' >$ originating by the terms $t \in d$ and all the words $t' \in V$, e.g. the WN's nouns. When the similarity between two documents is evaluated, their matching pairs are used to account for the final score. The weight given to each term pair is proportional to the similarity that the two terms have in WN. Thus, the term t of the first document contributes to the document similarity according to its relatedness with any of the terms of the second document and the prior external knowledge, provided by WN, quantifies the single term to term relatedness. Such approach has two advantages: (a) we obtain a well defined space which supports the similarity between terms of different surface forms based on external knowledge and (b) we avoid to explicitly define term or sense clusters which inevitably introduce noise.

The class of spaces which embeds the above pair information may be composed by $O(|V|^2)$ dimensions. If we consider only the WN nouns (about 10^5), our space contains about 10^{10} dimensions which is not manageable by most part of the learning algorithms. Kernel methods, can solve this problem as they allow us to use an implicit space representation in the learning algorithms. Among other Support Vector Machines (SVMs) (Vapnik, 1995) are kernel based learners which achieve high accuracy in presence of many irrelevant features. This is another important property for our approach as we leave the selection of the informative pairs to the SVM learning.

Moreover, as we believe that the prior knowledge in TC is not so useful when there is a sufficient amount of training documents, we experimented our model in poor training conditions (e.g. less equal than 20 documents for each category). The improvement in the accuracy, observed on the classification of the well known Reuters and 20 NewsGroups corpora, shows that our document similarity model is very promising for general IR tasks: unlike previous attempts, it makes sense of the adoption of semantic external resources (i.e. WN) in IR.

Section 2 introduces the WordNet-based term similarity. Section 3 defines the new document similarity measure, the kernel function and its use within SVMs. Section 4 presents the comparative results between the traditional linear and the WN-based kernels within SVMs. In Section 5 comparative discussion against the related IR literature is carried out. Finally Section 6 derives the conclusions.

2. Term similarity based on general knowledge

In IR, any similarity metric in the vector space models is driven by lexical matching. When small training material is available, few words can be effectively used and the resulting document similarity metrics are very weak. Semantic generalizations overcome data sparseness problems in IR as contributions from different but semantically similar words are made available.

Methods for the induction of semantically inspired word clusters have been widely used in language modeling and lexical acquisition tasks (e.g. (Clark & Weir, 2002)). The main resource employed in most works is WordNet (Fellbaum, 1998) which contains three subhierarchies: for nouns, verbs and adjectives. Each hierarchy represents lexicalized concepts (or senses) organized according to an "*is-a-kind-of*" relation. A concept s is described by a set of words $syn(s)$ called *synset*. The words $w \in syn(s)$ are synonyms according to the sense s.

For example, the words *line*, *argumentation*, *logical argument* and *line of reasoning* describe a synset which expresses the methodical process of logical reasoning (e.g. "*I can't follow your line of reasoning*"). Each word/term may be lexically related to more than one synset depending on the senses that it assumes. The word *line* is also present in the synset *line*, *dividing*

line, *demarcation* and *contrast*, to emphasize that a *line* denotes a conceptual separation or demarcation (e.g. "*there is a narrow line between sanity and insanity*").

In the next section we define a term similarity measure based on the WN noun hierarchy. Such hierarchy is a direct acyclic graph[1] in which the edges establish the *direct_isa* relations between two synsets.

2.1. The Conceptual Density

The automatic use of WordNet for NLP and IR tasks has proved to be very complex. First, how the topological distance among senses is related to their corresponding conceptual distance is unclear. The pervasive lexical ambiguity is also problematic as it impacts on the measure of conceptual distances between word pairs. Second, the approximation of a set of concepts by means of their generalization in the hierarchy implies a conceptual loss that affects the target IR (or NLP) tasks. For example, *black* and *white* are *colors* but are also *chess pieces* and this impacts on the similarity score that should be used in IR applications. Attempts to solve the above problems relates to *cuts* in the hierarchy (e.g. (Li & Abe, 1998; Resnik, 1997)) by using corpus statistics. For several tasks (e.g. in TC) this is unsatisfactory: different contexts of the same corpus (e.g. documents) may require different generalizations of the same word as they independently impact on the document similarity.

On the contrary, the *Conceptual Density* (CD) (Agirre & Rigau, 1996) is a flexible semantic similarity which depends on the generalizations of word senses not referring to any fixed level of the hierarchy. Its formal definition is given in what follows.

We denote by $\bar{s}$ the set of nodes of the hierarchy rooted in the synset s, i.e. $\{c \in S | c \; isa \; s\}$, where S is the set of WN synsets. By definition $\forall s \in S, s \in \bar{s}$. CD makes a guess about the proximity of the senses, s_1 and s_2, of two words u_1 and u_2, according to the information expressed by the minimal subhierarchy, $\bar{s}$, that includes them. Let S_i be the set of generalizations for at least one sense s_i of the word u_i, i.e. $S_i = \{s \in S | s_i \in \bar{s}, u_i \in syn(s_i)\}$. The CD of u_1 and u_2 is:

$$CD(u_1, u_2) = \begin{cases} 0 & \textbf{iff } S_1 \cap S_2 = \emptyset \\ max_{s \in S_1 \cap S_2} \frac{\sum_{i=0}^{h} (\mu(\bar{s}))^i}{|\bar{s}|} & \textbf{otherwise} \end{cases} \quad (1)$$

[1] As only the 1% of its nodes own more than one parent in the graph, most of the techniques assume the hierarchy to be a tree, and treat the few exception heuristically.

where:

- $S_1 \cap S_2$ is the set of WN shared generalizations (i.e. the common hypernyms) for u_1 and u_2
- $\mu(\bar{s})$ is the average number of children per node (i.e. the branching factor) in the sub-hierarchy $\bar{s}$. $\mu(\bar{s})$ depends on WordNet and in some cases its value can approach 1.
- h is the depth of the *ideal tree* whose leaves are only the two senses s_1 and s_2 and the average branching factor is $\mu(\bar{s})$. This value is actually estimated by:

$$h = \begin{cases} \lfloor log_{\mu(\bar{s})} 2 \rfloor & \textbf{iff } \mu(\bar{s}) \neq 1 \\ 2 & \textbf{otherwise} \end{cases} \quad (2)$$

  In cases $\mu(s)$ is exactly 1 the above equation assigns 2 to h.

- $|\bar{s}|$ is the number of nodes in the sub-hierarchy $\bar{s}$. This value is statically measured on WN and it is a negative bias for the higher level of generalizations (i.e. larger $\bar{s}$).

CD models the semantic distance as the density of the generalizations $s \in S_1 \cap S_2$. Such *density* is the ratio between the number of nodes of the *ideal tree* and $|\bar{s}|$. The ideal tree should (a) link the two senses/nodes s_1 and s_2 with the minimal number of edges (isa-relations) and (b) maintain the same branching factor (bf) observed in $\bar{s}$. In other words, this tree provides the minimal number of nodes (and isa-relations) sufficient to connect s_1 and s_2 according to the topological structure of $\bar{s}$. For example, if $\bar{s}$ has a bf of 2 the ideal tree connects the two senses with a single node (their father). If the bf is 1.5, to replicate it, the ideal tree must contain 4 nodes, i.e. the grandfather which has a bf of 1 and the father which has bf of 2 for an average of 1.5. When bf is 1 the Eq. 1 degenerates to the inverse of the number of nodes in the path between s_1 and s_2, i.e. the simple proximity measure used in (Siolas & d'Alch Buc, 2000).

It is worth noting that for each pair $CD(u_1, u_2)$ determines the similarity according to *the closest lexical senses*, s_1, $s_2 \in \bar{s}$: the remaining senses of u_1 and u_2 are irrelevant, with a resulting semantic disambiguation side effect. The CD properties seem appealing to define similarity measures between any term pairs in IR models. As the high number of such pairs increases the computational complexity of the target learning algorithm, efficient approaches are needed. The next section describes how kernel methods can make practical the use of the Conceptual Density in Text Categorization.

3. A WordNet Kernel for document similarity

Term similarities are used to design document similarities which are the core functions of most TC algorithms. The term similarity proposed in Eq. 1 is valid for all term pairs of a target vocabulary and has two main advantages: (1) the relatedness of each term occurring in the first document can be computed against *all* terms in the second document, i.e. all different pairs of similar (not just identical) tokens can contribute and (2) if we use all term pair contributions in the document similarity we obtain a measure consistent with the term probability distributions, i.e. the sum of all term contributions does not penalize or emphasize arbitrarily any subset of terms. The next section presents more formally the above idea.

3.1. A *semantic* vector space

Given two documents d_1 and $d_2 \in D$ (the document-set) we define their similarity as:

$$K(d_1, d_2) = \sum_{w_1 \in d_1, w_2 \in d_2} (\lambda_1 \lambda_2) \times \sigma(w_1, w_2) \qquad (3)$$

where λ_1 and λ_2 are the weights of the words (features) w_1 and w_2 in the documents d_1 and d_2, respectively and σ is a term similarity function, e.g. the conceptual density defined in Section 2. To prove that Eq. 3 is a valid kernel is enough to show that it is a specialization of the general definition of convolution kernels formalized in (Haussler, 1999). Hereafter, we report such definition: let $X, X_1, .., X_m$ be separable metric spaces, $x \in X$ a structure and $\vec{x} = x_1, ..., x_m$ its parts, where $x_i \in X_i \forall i = 1, .., m$. Let R be a relation on the set $X \times X_1 \times .. \times X_m$ such that $R(\vec{x}, x)$ holds if $\vec{x}$ are the parts of x. We indicate with $R^{-1}(x)$ the set $\{\vec{x} : R(\vec{x}, x)\}$. Given two objects x and $y \in X$ their similarity $K(x, y)$ is defined as:

$$K(x, y) = \sum_{\vec{x} \in R^{-1}(x)} \sum_{\vec{y} \in R^{-1}(y)} \prod_{i=1}^{m} K_i(x_i, y_i) \qquad (4)$$

If we consider X as the document set (i.e. $D = X$), $m = 1$ and $X_1 = V$ (i.e. the vocabulary of our target document corpus) we derive that: $x = d$ (i.e. a document), $\vec{x} = x_1 = w \in V$ (i.e. a word which is a part of the document d) and $R^{-1}(d)$ is the set of words in the document d. As $\prod_{i=1}^{m} K_i(x_i, y_i) = K_1(x_1, y_1)$, we can define $K_1(x_1, y_1) = K(w_1, w_2) = (\lambda_1 \lambda_2) \times \sigma(w_1, w_2)$ to obtain exactly the Eq. 3.

The above equation can be used in support vector machines as illustrated by the next section.

3.2. Support Vector Machines and Kernel methods

Given the vector space in $\mathbb{R}^\eta$ and a set of positive and negative points, SVMs classify vectors according to a separating hyperplane, $H(\vec{x}) = \vec{\omega} \cdot \vec{x} + b = 0$, where $\vec{x}$ and $\vec{\omega} \in \mathbb{R}^\eta$ and $b \in \mathbb{R}$ are learned by applying the *Structural Risk Minimization principle* (Vapnik, 1995). From the kernel theory we have that:

$$H(\vec{x}) = \Big(\sum_{h=1..l} \alpha_h \vec{x_h} \Big) \cdot \vec{x} + b = \sum_{h=1..l} \alpha_h \vec{x}_h \cdot \vec{x} + b =$$

$$\sum_{h=1..l} \alpha_h \phi(d_h) \cdot \phi(d) + b = \sum_{h=1..l} \alpha_h K(d_h, d) + b \qquad (5)$$

where, d is a classifying document and d_h are all the l training instances, projected in $\vec{x}$ and $\vec{x}_h$ respectively. The product $K(d, d_h) = <\phi(d) \cdot \phi(d_h)>$ is the *Semantic WN-based Kernel* (SK) function associated with the mapping ϕ.

Eq. 5 shows that to evaluate the separating hyperplane in $\mathbb{R}^\eta$ we do not need to evaluate the entire vector $\vec{x_h}$ or $\vec{x}$. Actually, we do not know even the mapping ϕ and the number of dimensions, η. As it is sufficient to compute $K(d, d_h)$, we can carry out the learning with Eq. 3 in the $\mathbb{R}^n$, avoiding to use the explicit representation in the $\mathbb{R}^\eta$ space. The real advantage is that we can consider only the word pairs associated with non-zero weights, i.e. we can use a sparse vector computation. Additionally, to have a uniform score across different document size, the kernel function can be normalized as follows: $\frac{SK(d_1, d_2)}{\sqrt{SK(d_1, d_1) \cdot SK(d_2, d_2)}}$

4. Experiments

The use of WordNet (WN) in the term similarity function introduces a prior knowledge whose impact on the Semantic Kernel (SK) should be experimentally assessed. The main goal is to compare the traditional Vector Space Model kernel against SK, both within the Support Vector learning algorithm.

The high complexity of the SK limits the size of the experiments that we can carry out in a feasible time. Moreover, we are not interested to large collections of training documents as in these training conditions the simple *bag-of-words* models are in general very effective, i.e. they seem to model well the document similarity needed by the learning algorithms. Thus, we carried out the experiments on small subsets of the

20NewsGroups[2] (20NG) and the *Reuters-21578*[3] corpora to simulate critical learning conditions.

4.1. Experimental set-up

For the experiments, we used the SVM-light software (Joachims, 1999) (available at `svmlight.joachims.org`) with the default linear kernel on the token space (adopted as the baseline evaluations). For the SK evaluation we implemented the Eq. 3 with $\sigma(\cdot,\cdot) = CD(\cdot,\cdot)$ (Eq. 1) inside SVM-light. As CD is sensitive only to nouns we detected them by means of a part of speech (POS) tagger. Nevertheless, given the importance of verbs, adjectives and numerical features for TC, we included them in the pair space by assigning a null value to the pairs made by different tokens. As the POS-tagger could introduce errors, we alternatively detected nouns by simply looking-up in WN, i.e. any word is considered as a noun if it is included in the noun WN hierarchy. This may be considered a rough approximation but it has the benefit to recover other useful information by including the similarity between the verb nominalizations and the other nouns, e.g. *to drive* like *drive* has a synset in common with *parkway*.

For the evaluations, we applied a careful SVM parameterization: a preliminary investigation suggested that the trade-off (between the training-set error and margin, i.e. c option in SVM-light) parameter optimizes the F_1 measure for values in the range [0.02,0.32][4]. We noted also that the cost-factor parameter (i.e. j option) is not critical, i.e. a value of 10 always optimizes the accuracy. The feature selection techniques and the weighting schemes were not applied in our experiments as they cannot be accurately estimated from the small available training data.

The classification performance was evaluated by means of the F_1 measure[5] for the single category and the MicroAverage for the final classifier pool (Yang, 1999). Given the high computational complexity of SK we selected 8 categories from the 20NG[6] and 8 from the Reuters corpus[7]

[2]Available at `www.ai.mit.edu/people/jrennie/20Newsgroups/`.

[3]The Apté split available at `kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`.

[4]We used all the values from 0.02 to 0.32 with step 0.02.

[5]F_1 assigns equal importance to Precision P and Recall R, i.e. $F_1 = \frac{2P \cdot R}{P+R}$.

[6]We selected the 8 most different categories (in terms of their content) i.e. *Atheism*, *Computer Graphics*, *Misc Forsale*, *Autos*, *Sport Baseball*, *Medicine*, *Talk Religions* and *Talk Politics*.

[7]We selected the 8 largest categories, i.e. *Acquisition*, *Crude*, *Earn*, *Grain*, *Interest*, *Money-fx*, *Trade* and *Wheat*.

To derive statistically significant results with few training documents, for each corpus, we randomly selected 10 different samples from the 8 categories. We trained the classifiers on one sample, parameterized on a second sample and derived the measures on the other 8. By rotating the training sample, we obtained 80 different measures for each model. The size of the samples ranges from 24 to 160 documents depending on the target experiment.

4.2. Cross validation results

The SK (Eq. 3) was compared with the linear kernel which obtained the best F_1 measure in (Joachims, 1999). Table 1 reports the first comparative results for 8 categories of 20NG on 40 training documents. The results are expressed as the *Mean* and the *Std. Dev.* over 80 runs. The F_1 are reported in Column 2 for the linear kernel, i.e. *bow*, in Column 3 for SK without applying POS information and in Column 4 for SK with the use of POS information (SK-POS). The last row shows the MicroAverage performance for the above three models on all 8 categories. We note that SK improves *bow* of 3%, i.e. 34.3% vs. 31.5% and that the POS information reduces the improvement of SK, i.e. 33.5% vs. 34.3%.

Category	*bow*	*SK*	*SK*-POS
Atheism	29.5±19.8	32.0±16.3	25.2±17.2
Comp.Graph	39.2±20.7	39.3±20.8	29.3±21.8
Misc.Forsale	61.3±17.7	51.3±18.7	49.5±20.4
Autos	26.2±22.7	26.0±20.6	33.5±26.8
Sport.Baseb.	32.7±20.1	36.9±22.5	41.8±19.2
Sci.Med	26.1±17.2	18.5±17.4	16.6±17.2
Talk.Relig.	23.5±11.6	28.4±19.0	27.6±17.0
Talk.Polit.	28.3±17.5	30.7±15.5	30.3±14.3
MicroAvg. F_1	31.5±4.8	34.3±5.8	33.5±6.4

Table 1. Performance of the linear and Semantic Kernel with 40 training documents over 8 categories of 20NewsGroups collection.

Category	24 docs		160 docs	
	bow	*SK*	*bow*	*SK*
Acq.	55.3±18.1	50.8±18.1	86.7±4.6	84.2±4.3
Crude	3.4±5.6	3.5±5.7	64.0±20.6	62.0±16.
Earn	64.0±10.0	64.7±10.3	91.3±5.5	90.4±5.1
Grain	45.0±33.4	44.4±29.6	69.9±16.3	73.7±14.
Interest	23.9±29.9	24.9±28.6	67.2±12.9	59.8±12.
Money-fx	36.1±34.3	39.2±29.5	69.1±11.9	67.4±13.
Trade	9.8±21.2	10.3±17.9	57.1±23.8	60.1±15.
Wheat	8.6±19.7	13.3±26.3	23.9±24.8	31.2±23.
Mic.Avg.	37.2±5.9	41.7±6.0	75.9±11.0	77.9±5.7

Table 2. Performance of the linear and Semantic Kernel with 24 and 160 training documents over 8 categories of the Reuters corpus.

To verify the hypothesis that WN information is useful

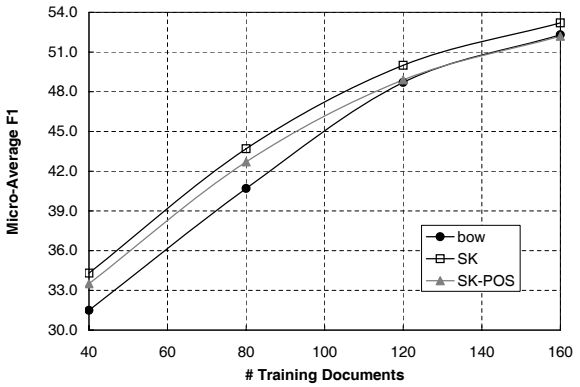

Figure 1. MicroAverage F_1 of SVMs using *bow*, SK and SK-POS kernels over the 8 categories of 20NewsGroups.

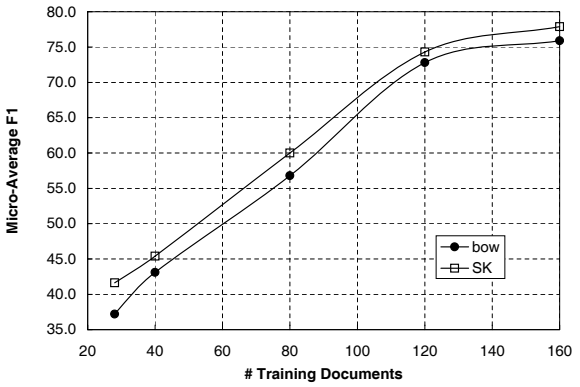

Figure 2. MicroAverage F_1 of SVMs using *bow* and SK over the 8 categories of the Reuters corpus.

in low training data conditions we repeated the evaluation over the 8 categories of Reuters with samples of 24 and 160 documents, respectively. The results reported in Table 2 shows that (1) again SK improves *bow* (41.7% - 37.2% = 4.5%) and (2) as the number of documents increases the improvement decreases (77.9% - 75.9% = 2%). It is worth noting that the standard deviations tend to assume high values. However, such variability does not affect the confidence test on the SK superiority. To verify that SK improves *bow*, we evaluated the Std. Dev. of the difference, d, between the MicroAverage F_1 of SK and the MicroAverage F_1 of *bow* over the samples. In relation to the Table 2 experiment, we obtained that the mean and the Std. Dev. of d on the 80 test samples of 24 documents are 4.53 and 6.57, respectively. We tested the hypothesis that *bow* has a higher or equal MicroAverage F_1 than SK, i.e. $d \leq 0$. Accordigly, the maximum value of the population average μ cannot be higher than 0, thus we tried the hypothesis $\mu = 0$. By using a Normal Distribution, d is in the range [-∞,μ+2.13] at a confidence level of 99.5%. Since the mean of the MicroAverage trough the samples (4.53) is not in such interval, we should reject such hypothesis.

The above findings confirm that SK outperforms the *bag-of-words* kernel in critical learning conditions as the semantic contribution of the SK recovers useful information. To complete this study we carried out experiments with samples of different size, i.e. 3, 5, 10, 15 and 20 documents for each category. Figures 1 and 2 show the learning curves for 20NG and Reuters corpora. Each point refers to the average on 80 samples.

As expected the improvement provided by SK decreases when more training data is available. However, the improvement is not negligible yet. The SK model (without POS information) preserves about 2-3% of improvement with 160 training documents. The matching allowed between noun-verb pairs still captures semantic information which is useful for topic detection. In particular, during the similarity estimation, each word activates 60.05 pairs on average. This is particularly useful to increase the amount of information available to the SVMs.

Finally, we carried out some experiments with 160 Reuters documents by discarding the string matching from SK. Only words having different surface forms were allowed to give contributions to the Eq. 3.

The interesting outcome is that SK converges to a MicroAverage F_1 measure of 56.4% (compare with Table 2). This shows that the word similarity provided by WN is consistent and effective for TC.

5. Related Work

The IR studies in this area focus on the term similarity models to embed statistical and external knowledge in document similarity.

In (Kontostathis & Pottenger, 2002) a *Latent Semantic Indexing* analysis was used for term clustering. Such approach assumes that values x_{ij} in the transformed term-term matrix represents the similarity (> 0) and anti-similarity between terms i and j. By extension, a negative value represents an anti-similarity between i and j enabling both positive and negative clusters of terms. Evaluation of query expansion techniques showed that positive clusters can improve Recall of about 18% for the *CISI* collection, 2.9% for *MED* and 3.4% for *CRAN*. Furthermore, the negative clusters, when used to prune the result set, improve the precision.

The use of external semantic knowledge seems to be

more problematic in IR. In (Smeaton, 1999), the impact of semantic ambiguity on IR is studied. A WN-based semantic similarity function between noun pairs is used to improve indexing and document-query matching. However, the WSD algorithm had a performance ranging between 60-70%, and this made the overall semantic similarity not effective.

Other studies using semantic information for improving IR were carried out in (Sussna, 1993) and (Voorhees, 1993; Voorhees, 1994). Word semantic information was here used for text indexing and query expansion, respectively. In (Voorhees, 1994) it is shown that semantic information derived directly from WN without a priori WSD produces poor results.

The latter methods are even more problematic in TC (Moschitti & Basili, 2004). Word senses tend to systematically correlate with the positive examples of a category. Different categories are better characterized by different words rather than different senses. Patterns of lexical co-occurrences in the training data seem to suffice for automatic disambiguation. (Scott & Matwin, 1999) use WN senses to replace simple words without word sense disambiguation and small improvements are derived only for a small corpus. The scale and assessment provided in (Moschitti & Basili, 2004) (3 corpora using cross-validation techniques) showed that even the accurate disambiguation of WN senses (about 80% accuracy on nouns) did not improve TC.

In (Siolas & d'Alch Buc, 2000) was proposed an approach similar to the one presented in this article. A term proximity function is used to design a kernel able to semantically smooth the similarity between two document terms. Such semantic kernel was designed as a combination of the Radial Basis Function (RBF) kernel with the term proximity matrix. Entries in this matrix are inversely proportional to the length of the WN hierarchy path linking the two terms. The performance, measured over the 20NewsGroups corpus, showed an improvement of 2% over the *bag-of-words*. The main differences with our approach are: first, the term proximity is not fully sensitive to the information of the WN hierarchy. For example, if we consider pairs of equidistant terms, the nearer to the WN top level a pair is the lower similarity it should receive, e.g. *Sky* and *Location* (hyponyms of *Entity*) should not accumulate similarity like *knife* and *gun* (hyponyms of *weapon*). Measures, like CD, that deal with this problem have been widely proposed in literature (e.g. (Resnik, 1997)) and should be always applied. Second, as our main goal was the study of the CD information in document retrieval/categorization scenario, our kernel function was based on the simple CD similarity. In (Siolas & d'Alch Buc, 2000) weighting schemes and the RBF kernel were used along with the proximitry matrix. Probably, this combination has downgraded the role of WN semantics. Finally, the experiments were carried out by using only 200 features (selected via Mutual Information statistics). In this way the contribution of rare or non statistically significant terms is neglected. In our view, the latter features may give, instead, a relevant contribution once we move in the SK space generated by the WN similarities.

Other important work on semantic kernel for retrieval has been developed in (Cristianini et al., 2002; Kandola et al., 2002). Two methods for inferring semantic similarity from a corpus were proposed. In the first a system of equations were derived from the dual relation between word-similarity based on document-similarity and viceversa. The equilibrium point was used to derive the semantic similarity measure. The second method models semantic relations by means of a diffusion process on a graph defined by lexicon and co-occurrence information. The major difference with our approach is the use of a different source of prior knowledge, i.e. WN. Similar techniques were also applied in (Hofmann, 2000) to derive a Fisher kernel based on a latent class decomposition of the term-document matrix.

6. Conclusions

The introduction of semantic prior knowledge in IR and TC is important as a way to lower the training set size and thus increase the applicability of Web learning from suitably selected examples. In this paper, we used the conceptual density function on the WordNet (WN) hierarchy to define a document similarity metric and derive a semantic kernel to train Support Vector Machine classifiers. Cross-validation experiments over 8 categories of 20NewsGroups and Reuters over multiple samples have shown that in poor training data conditions, the WN prior knowledge can be effectively used to improve (up to 4.5 absolute percent points, i.e. 10%) the TC accuracy.

These promising results enable a number of future researches: (1) larger scale experiments with different measures and semantic similarity models (e.g. (Resnik, 1997)); (2) domain-driven specialization of the term similarity by selectively tuning WordNet to the target categories, (3) the impact of feature selection on SK, and (4) the extension of the semantic similarity by a general (i.e. non binary) application of the conceptual density model, e.g. the most important category terms as a prior bias for the similarity score.

Acknowledgments

This research is partially supported by the European project, PrestoSpace (FP6-IST-507336).

References

Agirre, E., & Rigau, G. (1996). Word sense disambiguation using conceptual density. *Proceedings of COLING'96, pages 16–22, Copenhagen, Danmark.*.

Bekkerman, R., El-Yaniv, R., Tishby, N., & Winter, Y. (2001). On feature distributional clustering for text categorization. *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 146–153). New Orleans, Louisiana, United States: ACM Press.

Clark, S., & Weir, D. (2002). Class-based probability estimation using a semantic hierarchy. *Comput. Linguist.*, *28*, 187–206.

Cristianini, N., Shawe-Taylor, J., & Lodhi, H. (2002). Latent semantic kernels. *J. Intell. Inf. Syst.*, *18*, 127–152.

Fellbaum, C. (1998). *Wordnet: An electronic lexical database*. MIT Press.

Haussler, D. (1999). *Convolution kernels on discrete structures*Technical Report UCS-CRL-99-10). University of California Santa Cruz.

Hofmann, T. (2000). Learning probabilistic models of the web. *Research and Development in Information Retrieval* (pp. 369–371).

Joachims, T. (1999). Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*.

Kandola, J., Shawe-Taylor, J., & Cristianini, N. (2002). Learning semantic similarity. *in Neural Information Processing Systems (NIPS 15) - MIT Press.*.

Kontostathis, A., & Pottenger, W. (2002). Improving retrieval performance with positive and negative equivalence classes of terms.

Li, H., & Abe, N. (1998). Generalizing case frames using a thesaurus and the mdl principle. *Computational Linguistics*, *23*.

Moschitti, A., & Basili, R. (2004). Complex linguistic features for text classification: a comprehensive study. *Proceedings of ECIR-04, 26th European Conference on Information Retrieval*. Sunderland, UK: Springer Verlag.

Resnik, P. (1997). Selectional preference and sense disambiguation. *Proceedings of ACL Siglex Workshop on Tagging Text with Lexical Semantics, Why, What and How?, Washington, April 4-5, 1997.*.

Scott, S., & Matwin, S. (1999). Feature engineering for text classification. *Proceedings of ICML-99, 16th International Conference on Machine Learning* (pp. 379–388). Bled, SL: Morgan Kaufmann Publishers, San Francisco, US.

Siolas, G., & d'Alch Buc, F. (2000). Support vector machines based on a semantic kernel for text categorization. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 5* (p. 5205). IEEE Computer Society.

Smeaton, A. F. (1999). Using NLP or NLP resources for information retrieval tasks. In T. Strzalkowski (Ed.), *Natural language information retrieval*, 99–111. Dordrecht, NL: Kluwer Academic Publishers.

Sussua, M. (1993). Word sense disambiguation for free-text indexing using a massive semantic network. *The Second International Conference on Information and Knowledge Management (CKIM 93)* (pp. 67–74).

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer.

Voorhees, E. M. (1993). Using wordnet to disambiguate word senses for text retrieval. *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, June 27 - July 1, 1993* (pp. 171–180). ACM.

Voorhees, E. M. (1994). Query expansion using lexical-semantic relations. *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)* (pp. 61–69). ACM/Springer.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*.

Learning Word-to-Concept Mappings for Automatic Text Classification

Georgiana Ifrim IFRIM@MPI-INF.MPG.DE
Martin Theobald MARTIN.THEOBALD@MPI-INF.MPG.DE
Gerhard Weikum WEIKUM@MPI-INF.MPG.DE
Max-Planck Institute for Informatics, D-66041 Saarbruecken, Germany

Abstract

For both classification and retrieval of natural language text documents, the standard document representation is a term vector where a term is simply a morphological normal form of the corresponding word. A potentially better approach would be to map every word onto a concept, the proper word sense and use this additional information in the learning process. In this paper we address the problem of automatically classifying natural language text documents. We investigate the effect of word to concept mappings and word sense disambiguation techniques on improving classification accuracy. We use the WordNet thesaurus as a background knowledge base and propose a generative language model approach to document classification. We show experimental results comparing the performance of our model with Naive Bayes and SVM classifiers.

1. Introduction

1.1. Motivation

Text classification, e.g., for categorizing Web documents into topics like sports, science, math, etc., is usually based on supervised learning techniques such as support vector machines (SVM) with feature vectors as representatives of both the training and test documents. The features are usually derived from the bag-of-words model, where individual words or word stems constitute features and various frequency measures are used to compute weights, e.g., using the $tf \cdot idf$ approach or statistical language models (Manning & Schütze, 2000; Croft & Lafferty, 2003). Classification accuracy is limited by three potential bottlenecks: 1) the quality of the training data, 2) the discriminative power of the classifier and 3) the richness of the features to represent documents. The first point is usually an application issue and beyond control of the classifier, and with the great advances in statistical learning, the second point is widely perceived as the least limiting factor. In this paper, we address the third point.

Despite the sophisticated statistical models for computing feature weights, using words or word stems as features is a semantically poor representation of the text content. Richer features could be derived from syntactic analysis of the text (using part-of-speech tagging, chunk parsing, etc. (Manning & Schütze, 2000)), and most importantly, using concepts rather than words, thus capturing the intended word sense instead of the literal expressions in the document. As an example, consider the word "Java", a classical polysem (i.e., a word with multiple word senses). For classifying a document into topic "travel" or "computer science", the word itself is not helpful. But if we could map it to its proper meaning, within the context of the document, then we would be able to boost the classifier: if "Java" is used as the concept "island (part of Indonesia)" it should raise the probability of category "travel", whereas the use as the concept "object-oriented programming language" would give higher evidence to the category "computer science".

For mapping words onto concepts, we build on the availability of rich knowledge sources like lexicons, thesauri, and ontologies. For the scope of this paper, we specifically use the WordNet thesaurus (Fellbaum, 1999), which contains around 150,000 concepts (word senses in WordNet's terminology), each with a short textual description, and semantic relationships between concepts - hypernym/hyponym (IS A), holonym/meronym (PART OF). We use a machine learning approach, based on latent variables and EM

Appearing in *W4: Learning in Web Search, at the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

iteration for parameter estimation, to compute the actual word-to-concept mappings. It is important to note that WordNet, albeit probably the most prominent source of this kind, is just an example of the explicit concept collections that could be leveraged for better text representation and classification accuracy. Ontologies are being built up (Staab & Studer, 2004), and it is conceivable that concepts can be mined from encyclopedia like Wikipedia.

1.2. Contribution

Our approach is based on a generative model for text documents, where words are generated by concepts which in turn are generated by topics. We postulate conditional independence between words and topics given the concepts. Once the corresponding probabilities for word-concept and concept-topic pairs are estimated, we can use Bayesian inference to compute the probability that a previously unseen test document with known words but unobservable concepts belongs to a certain topic. The concepts are used as latent variables here, but note that unlike earlier work on spectral decomposition (Deerwester & Dumais & Harshman, 1990; Hofmann, 2001) for text retrieval our concepts are named and can be explicitly identified in the underlying thesaurus or ontology.

The learning procedure for estimating the probabilities that involve latent variables is a maximum-likelihood estimator based on the observed word-topic pairs in the training data. We use an EM (expectation-maximization) procedure for iteratively solving the analytically intractable estimation problem. The number of concepts that we consider in this approach is naturally limited and determined by an initialization step that uses a text-context similarity comparison for an initial, heuristic mapping of words onto concepts. Note, however, that the final result of the word-to-concept mapping is usually much better than the outcome of the initial heuristics. Our overall approach can also be seen as a learning-based method for word sense disambiguation coupled with a classifier for topic labeling.

Different flavors of latent variable models for text data exist in the literature (Cai & Hofmann, 2003; Bhattacharya & Getoor & Bengio, 2004). Previous work employed Wordnet for feature engineering (Scott & Matwin, 1999; Bloehdorn & Hotho, 2004), but our model has the following major advantages, which we claim as our main contributions:

1. By using explicit concepts from a thesaurus or ontology and by initially using a heuristic technique for bootstrapping the word-to-concept mapping, we avoid the model selection problem faced inevitably by all techniques based on latent dimensions and spectral analysis (i.e., choosing an appropriate number of latent dimensions).

2. By the same token, we avoid the potential combinatorial explosion in the space of parameters to be estimated, and we can do away with the need for parameter smoothing (often a very tricky and treacherous issue).

3. The initial mapping provides us with a good initialization of the EM iteration, positively affecting its convergence and reducing the (empirical) risk that it gets stuck in a local maximum of the likelihood function.

In our experiments, with real-life datasets from the Reuters newswire corpus and editorial reviews of books from the Amazon web site, we compare our approach with a Naive Bayes classifier and an SVM classifier (Hastie & Tibshirani & Friedman, 2003; McCallum & Nigam, 1998; Joachims, 1998). The results show that our method can provide substantial gains in classification accuracy for rich text data where the expressiveness and potential ambiguity of natural language becomes a bottleneck for traditional bag-of-words classifiers.

The rest of the paper is organized as follows. Section 2 describes our probabilistic generative model. Section 3 presents our techniques for efficiently estimating the model parameters. Section 4 discusses experimental results.

2. Probabilistic Model

2.1. Generative Model

In this section we introduce our framework and the theoretical model proposed. The general setup is the following:

- A *document collection*, $D = \{d_1, \ldots, d_r\}$, with known *topic labels*, $T = \{t_1, \ldots, t_m\}$, which is split into training and test data. In this work we assume a one-to-one mapping between documents and topic labels.

- A set of *lexical features*, $F = \{f_1, \ldots, f_n\}$, that can be observed in documents (*individual or composite words*).

- An *ontology DAG of concepts*, $C = \{c_1, \ldots, c_k\}$, where each concept has a set of synonyms and a

short textual description, and is related to other concepts by semantic edges.

The goal is solving a document classification problem: for a given document d with observed features, we would like to predict $P[t|d]$ for every topic t or find $argmax_t P[t|d]$. To get an intuition behind our model, consider analyzing documents labeled with a certain topic label, e.g. *physics*. Conceptually, this broad concept (the topic label) can be described at semantic level by a subset of more fine grained concepts that describe for example phenomena or structures related to physics, e.g *atom, molecule, particle, corpuscle, physical science*, etc. In turn, these concepts are expressed at the lexical level, by means of simple terms or compounds: *physical science*, *material*. Thus, we want to explain feature-topic associations by means of latent concepts. Figure 1 shows a graphical representation of our generative model. The model proposed by us is similar to the *aspect model* developed in (Hofmann, 2001). It is a latent variable model for co-occurrence data which associates an unobserved variable $c \in \{c_1 \dots c_k\}$ with each observation.

Our model differs from the aspect model in the following respects. In the aspect model, the number of concepts is fixed beforehand, but the concepts themselves are derived in an unsupervised way from the data collection, *without recourse to a lexicon or thesaurus*; an observation is the occurrence of a word in a particular document; parameters are randomly initialized. Our model uses the existing knowledge resources to identify and select the latent concepts at runtime; an observation is a pair (f, t), where $f \in F$ is a feature observed in some document and $t \in T$ stands for a topic label; parameters are pre-initialized to help model robustness. Our generative model for feature-topic co-occurrence can be described as:

1. Select a topic t with probability $P[t]$;
2. Pick a latent variable c with probability $P[c|t]$, the probability that concept c describes topic t;
3. Generate a feature f with probability $P[f|c]$, the probability that feature f means concept c.

The pairs (f, t) can be directly observed, while the existence of concepts implies some process of word sense disambiguation and they are treated as latent variables. The model is based on two independence assumptions: observation pairs (f, t) are assumed to be generated independently and it is assumed that features f are conditionally independent of the topics t, given the latent variable c: $P[(f, t)|c] = P[f|c] \cdot P[t|c]$. To describe the generative process of an observation (f, t) we sum up over all the possible values that the latent variables might take

$$P[f, t] = \sum_c P[c] \cdot P[(f, t)|c]. \quad (1)$$

The likelihood of the observed pairs (f, t) can be expressed as:

$$L = \Pi_{f,t} P[f, t]^{n(f,t)} \quad (2)$$

where $n(f, t)$ is the number of occurrences of feature f in the training set of topic t.
The learning problem can be expressed now as a maximization of the observed data log-likelihood:

$$\begin{aligned} l &= \sum_{(f,t)} n(f, t) \cdot log(P[f, t]) \quad (3) \\ &= \sum_{(f,t)} n(f, t) \cdot log(\sum_c P[c] \cdot P[(f, t)|c]) \end{aligned}$$

Due to the existence of the sum inside the logarithm direct maximization of the log-likelihood by partial derivatives is difficult. A solution in setups in which maximization of the likelihood is difficult, but made easier by enlarging the sample with latent data, is to apply an Expectation-Maximization (EM) algorithm. The EM algorithm works by 2 iterative steps:

- **E-Step:** Expectation step, in which posterior probabilities are estimated for the latent variables, taking as evidence the observed data (current estimates of the model parameters). For calculating the probabilities of the E-step, we use Bayes' formula:

$$P[c|(f, t)] = \frac{P[f|c] \cdot P[c|t]}{\sum_c P[f|c] \cdot P[c|t]} \quad (4)$$

- **M-Step:** Maximization step, in which the current parameters are updated based on the expected complete data log-likelihood which depends on the posterior probabilities estimated in the E-Step.

$$P[f|c] = \frac{\sum_t n(f, t) P[c|(f, t)]}{\sum_f \sum_t n(f, t) P[c|(f, t)]} \quad (5)$$

$$P[c|t] = \frac{\sum_f n(f, t) P[c|(f, t)]}{\sum_c \sum_f n(f, t) P[c|(f, t)]} \quad (6)$$

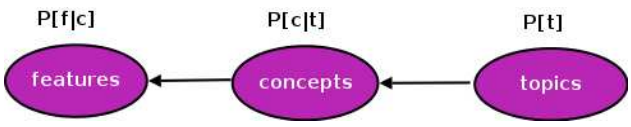

Figure 1. Graphical model representation of the generative model

$$P[t] = \frac{\sum_{f,c} n(f,t)P[c|(f,t)]}{\sum_t \sum_{f,c} n(f,t)P[c|(f,t)]} \quad (7)$$

In our implementation the E-step and the M-step are iterated until convergence of the likelihood. Alternatively, one can also use the technique of *early stopping* - stop the algorithm when the performance on some held-out data starts decreasing, in order to avoid overfitting the model.

Now, we estimate the distribution of a document d, given a topic label t, by making use of the learned features' marginal distributions during the training process:

$$\begin{aligned} P[d|t] &= \Pi_{f\in d} P[f|t] = \Pi_{f\in d} \frac{P[f,t]}{P[t]} \quad (8) \\ &= \Pi_{f\in d} \sum_{c\in C} P[f|c] \cdot P[c|t] \end{aligned}$$

where $P[f|c]$ and $P[c|t]$ are estimated by the EM procedure so as to maximize $P[f,t]$ and implicitly $P[d|t]$.

2.2. Naive Bayes Classifier

Once we have estimates for the marginal distribution describing the generative model, we can use Bayes rule to reverse the model and predict which topic generated a certain document:

$$P[t|d] = \frac{P[d|t] \cdot P[t]}{P[d]} = \frac{P[d|t] \cdot P[t]}{\sum_t P[d|t] \cdot P[t]} \quad (9)$$

We can then substitute (8) into (9) and have a decision procedure for the classifier. The hope is that by the means of the latent variable model the distribution that generated the given document will be estimated in a more accurate way.

3. Model Parameter Estimation

EM can face two major problems:

- The combinatorial explosion of the variable space in the model, since the number of parameters is directly proportional to the cross-product of the number of features, concepts and topics. These parameters are sparsely represented in the observed training data.
- The possibility of converging to a local maximum of the likelihood function (i.e. not finding the global maximum).

For the first problem, it is desirable to prune the parameter space to reflect only the meaningful latent variables. For the second problem, it is desirable to pre-initialize the model parameters to values that are close to the global maximum of the likelihood function.

3.1. Pruning the Parameter Space

3.1.1. Feature Selection

The feature selection process is done by retaining the features that have the highest average Mutual Information with the topic variable (McCallum & Nigam, 1998). For *multinomial models* the quantity is computed by calculating the mutual information between the topic of the document from which a word occurrence is drawn, and a random variable over all word occurrences.

$$f_k = \begin{cases} 1 & \text{if } w_k \text{ is present,} \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

$$\begin{aligned} MI(T;W_k) &= H(T) - H(T|W_k) \quad (11) \\ &= \sum_{t\in T} \sum_{f_k\in\{0,1\}} P(t,f_k) \cdot log\left(\frac{P(t,f_k)}{P(t)\cdot P(f_k)}\right) \end{aligned}$$

As a preprocessing step before applying feature selection, we extract semantically significant compounds using a background dictionary (WordNet) e.g. *exchange market*, *linear algebra*, etc. This is a further step in capturing the semantics of interesting and common language constructions; it also reduces some of the computational overhead, while also achieving an increase in accuracy: many compound terms have only one meaning, e.g. *exchange market*, as a compound has fewer meanings than if analyzed separately *exchange* and *market*. After this stage, we can apply the MI selection criterion. Sorting the features in descending order of this measure gives us a ranking in terms of discriminative power of the features.

3.1.2. Concept Set Selection

WordNet contains around 150,000 concepts linked by hierarchical relations. Using the full set of concepts provided by the ontology results in a high computational overhead combined with a high amount of noise. A better approach is to select from the ontology only a subset of concepts that reflects well the semantics of the training collection. In our work, we call this the *candidate set of concepts*. The set is selected in a preprocessing step, before running the EM algorithm. One way of capturing the candidate set well is to gather for each feature all the corresponding concepts (senses) from the ontology. The size order of this subset is only of a few thousands concepts, as opposed to some hundred-thousands available in the

ontology. Another way of even further improving the performance of this approach, is using PoS annotated data. We considered both approaches in our implementation.

3.2. Pre-initializing the Model Parameters

The standard way of using the EM algorithm is to randomly initialize the model parameters and iterate the algorithm until convergence. Since EM tends to stop in a local maximum of the likelihood function, the algorithm is restarted several times, and the values of the parameters that give the highest value of the likelihood are retained. However, this solution still does not guarantee that EM will stop at a global maximum. Our pre-initialization proposal combines the learning approach with a simpler approach of mapping features to concepts and concepts to topics, based on *similarity measures*.

For the initial mapping of words onto concepts in a thesaurus (ontology) we follow the approach in (Theobald & Schenkel & Weikum, 2003). The WordNet thesaurus can be seen as a DAG where the nodes are the different meanings and the edges are semantic relationships (Fellbaum, 1999). The vertices can be nouns, adverbs, verbs or adjectives.

Let w be a word that we want to map to the ontological senses. First, we query WordNet for the possible meanings of word w; for improving precision we can use PoS annotations (i.e., noun vs. verb vs. adjective). Let $\{c_1, \ldots, c_m\}$ be the set of meanings associated with w. For example, if we query WordNet for the word *mouse* we get:

- *The* **noun** *mouse has 2 senses in WordNet.*
  1. *mouse – (any of numerous small rodents...)*
  2. *mouse, computer mouse – (a hand-operated electronic device...)*
- *The* **verb** *mouse has 2 senses in WordNet.*
  1. *sneak, mouse, creep, steal, pussyfoot – (to go stealthily or furtively)*
  2. *mouse – (manipulate the mouse of a computer)*

By taking also the synonyms of these word senses, we can form *synsets* for each of the word meanings. Next, we apply a word sense disambiguation step.

The disambiguation technique proposed uses word statistics for a local context around both the word observed in a document and each of the possible meanings it may take. The context for the word is a window around its offset in the text document; the context for the concept is taken from the ontology: for each sense c_i we take its synonyms, hypernyms, hyponyms, holonyms, and siblings and their short textual descriptions. The context of a concept in the ontology graph can be taken until a certain depth, depending on the amount of noise one is willing to introduce in the disambiguation process. In this work we use depth 2. For each of the candidate senses c_i, we compare the context around the word $context(w)$ with $context(c_i)$ in terms of bag-of-words similarity measures. We use the cosine similarity measure between the $tf \cdot idf$ vectors of $context(w)$ and $context(c_i)$, $\quad i \in \{1, \ldots, m\}$. This process can either be seen as a proper word sense disambiguation step, if we take as corresponding word sense the one with the highest context similarity to the word's context, or as a step of establishing how words and concepts are related together and in what degree.

In a similar fashion, we relate concepts to topics based on similarity of bags-of-words. The context for a topic t is defined to be the bag-of-features selected from the training collection by decreasing Mutual Information value. For our implementation, we used the top 50 (compound) terms with regards to MI rank. Once we have computed all the similarities for (feature, concept) and (concept,topic) pairs, we normalize them, and interpret them as estimates of the probabilities $P[f|c]$ and $P[c|t]$. In the $sim(f, c)$ and $sim(c, t)$ computations, we only consider the (f, c) and (c, t) pairs in the pruned parameter space. The computed values are then used for initializing EM, as a preprocessing stage, in the model fitting process.

4. Preliminary Experiments

4.1. Setup

We present some preliminary experiments on two data collections. We analyze and compare four classification methods: *LatentM* - a first version of the latent generative model proposed by us that does not exploit PoS tags; *LatentMPoS* - our generative model, enhanced with methods for exploiting PoS information; *NBayes* - a terms-only Naive Bayes classifier; *SVM* - a multiclass SVM classifier. For the SVM classifier, we have used the SVM Light and SVM Struct tools, developed for multiclass classification (Tsochantaridis & Hoffman & Joachims & Altun, 2004). To measure classification quality, we use microaveraged F_1-measure (Manning & Schütze, 2000). Training and test were performed on disjoint document sets.

4.2. Reuters-21578

The Reuters-21578 dataset is a news collection compiled from the Reuters newswire in 1987. We used the

"ModApte" split, that led to a corpus of 9,603 training documents and 3,299 test documents. We parsed the document collection and retained only documents belonging to one topic. Out of these, we selected the top five categories in terms of number of training documents available: *earn, acq, crude, trade, money-fx*. This split the collection into approximately 5,000 files for training and 2,000 files for testing. The classification task is to assign articles to their corresponding topics. For many categories, there is a direct correspondence between words and category labels e.g., the appearance of the term *acquisition* is a very good predictor of the *acq* category. The vocabulary is fairly small and uniform, each topic is described with standard terms, e.g. *crude oil, opec, barrel* are very frequent terms in the topic *crude*, so by using frequency of terms only we can get a high classification accuracy. We tested the sensitivity to the training set size for all the four methods. We averaged the performance over 3 randomly selected training sets of sizes: 10 to 200 documents per topic. The number of features is set to 300 based on studies concerning the appropriate vocabulary size for Reuters (McCallum & Nigam, 1998), which indicate this number of features is enough for obtaining a high classification accuracy. Particularly for topic *trade*, a high amount of noise is introduced by enlarging the feature space. Table 1 shows statistics regarding the number of concepts in our model, for different training set sizes. For the method using part of speech annotations, we use nouns and verbs. Table 2 shows microaveraged F1 results for the 5 chosen topics. We can observe that on the Reuters collection, exploiting the semantics of natural language does not outperform the methods that use simple term frequencies. We explain this effect by the nature of the vocabulary used in this collection in which term frequencies capture the nature of the training data in each topic well enough. Further studies are necessary in order to fully understand the behavior of the techniques proposed on this data collection.

Table 1. Number of concepts extracted for various training set sizes on Reuters-21578.

Training per topic	Concepts LatentM	Concepts LatentMPoS
10	2669	1560
20	2426	1395
30	2412	1321
40	2364	1447
50	2411	1317
100	2475	1372
150	2477	1385
200	2480	1387

Table 2. Microaveraged F1 results on Reuters-21578.

Training per topic	NBayes	LatentM	LatentM PoS	SVM
10	88.9%	88.7%	87.8%	90.0%
20	89.6%	92.2%	90.7%	92.1%
30	92.7%	94.0%	92.2%	93.6%
40	92.1%	93.0%	91.2%	94.5%
50	93.8%	95.0%	93.8%	93.8%
100	95.3%	95.0%	93.8%	95.5%
150	96.0%	95.0%	94.4%	95.4%
200	95.9%	95.8%	94.5%	95.9%

4.3. Amazon

In order to further test our methods, we extracted a real-world collection of natural language text from `http://www.amazon.com`. This site promotes books, which are grouped according to a representative category. From the available taxonomy, we selected all the editorial reviews for books in: *Biological Sciences, Mathematics, Physics.* Total number of documents extracted was 6,000 (48MB). We split this set into training (largest 500 documents per topic) and test (remaining documents after training selection). After this process we obtained 1,500 training documents and 4,500 test documents. The dataset is available at `http://www.mpi-sb.mpg.de/~ifrim/data/Amazon.zip`. Table 3 shows the distribution of documents over topics. For the method using PoS annotations, we use nouns, adjectives and verbs. For each of the methods analyzed, we tested the sensitivity to vocabulary size. Table 4 presents statistics regarding the concepts involved in the latent models for different dimensions of the feature space. Figure 2 shows microaveraged F1 results. We can observe a significant improvement in terms of performance achieved at different dimensionalities of the feature space. The *PoS* label attached to each method's name stands for the

Table 3. Training/test documents on Amazon.

Category Name	Train size	Test size
Mathematics	500	2,237
Biological Sciences	500	1,476
Physics	500	787

Table 4. Number of concepts extracted for various feature set sizes on Amazon.

Number of features	Concepts LatentM	Concepts LatentMPoS
100	1099	509
200	1957	936
300	2886	1390
400	3677	1922
500	4623	2232
600	5354	2547
700	5973	2867
800	6551	3231
900	7230	3677
1,000	7877	3959

usage of PoS annotated features for the respective method. The only difference between $LatentMPos$ and $NBayesPoS$ or $SVMPoS$ is the mapping of features onto the concept space.

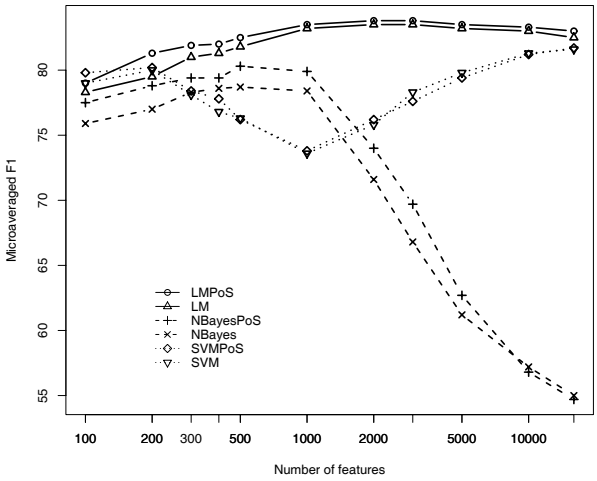

Figure 2. Microaveraged F_1 at different number of features.

Since this collection has a richer vocabulary, synonymy and polysemy effects can have more impact. We observe that exploiting semantics can have the potential of boosting classification performance. In Table 5 and 6, we show the exact values for microaveraged F_1 at higher dimensionalities of the feature space. We observe that SVM performance using all the distinct terms in the collection (16,000) is inferior to our model with 1,000 features. Feature selection by MI does not eliminate correlations among features. This can have an effect on SVM performance for small dimensionalities of the feature space. We trained SVM using the default settings of SVM Struct: linear kernel and $C = 0.01$. In the future we plan a systematic study regarding SVM parameters tuning.

In Figure 3 and Table 7 we show the sensitivity of microaveraged F_1 to the training set size for all the methods under discussion. The number of features was set to 500 for Naive Bayes methods. For SVM we used all the available terms. Also, we compared our initialization heuristic to the random one. Table 8 shows the EM behavior, using the LatentMPOS model with 500 features on the entire training collection. As compared to the random initialization, our similarity based heuristic does not gain much in terms of accuracy. However, it converges faster. Table 9 shows experimental results targeted at assessing the strength of the heuristic itself, without any EM iteration. The column *Heuristic* shows classification results using only the similarity-based initialization heuristic, compared to the performance achieved after one EM iteration (column *Heuristic-EM1*). Column *Random-EM1* shows the performance after one EM iteration with *random* initialization of parameters.

Table 5. Microavg F_1 for different number of features.

Number of features	Microavg F_1 NBayes	Microavg F_1 LatentM	Microavg F_1 SVM
100	75.9%	**78.3%**	79.0%
200	77.0%	**79.5%**	80.0%
300	78.3%	**81.0%**	78.1%
400	78.6%	**81.3%**	76.8%
500	78.7%	**81.8%**	76.3%
1,000	78.4%	**83.2%**	73.6%
2,000	71.6%	**83.5%**	75.8%
3,000	66.8%	**83.5%**	78.3%
5,000	61.2%	**83.1%**	79.8%
10,000	57.2%	**82.7%**	81.3%
16,000	55.0%	**82.4%**	81.6%

Table 6. Microavg F_1 for different number of PoS features.

Number of features	Microavg F_1 NBayesPoS	Microavg F_1 LatentMPoS	Microavg F_1 SVMPoS
100	77.5%	**79.0%**	79.8%
200	78.8%	**81.3%**	80.2%
300	79.4%	**81.9%**	78.4%
400	79.9%	**82.0%**	77.8 %
500	80.3%	**82.5%**	76.2%
1,000	79.9%	**83.5%**	73.8%
2,000	74.0%	**83.8%**	76.2%
3,000	69.7%	**83.8%**	77.6%
5,000	62.7%	**83.4%**	79.4%
10,000	56.8%	**83.1%**	81.2%
16,000	54.7%	**82.5%**	81.7%

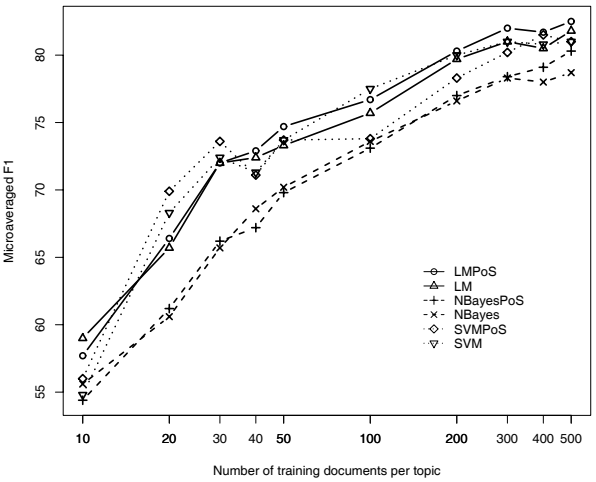

Figure 3. Microaveraged F_1 for different training set size.

Table 7. Microaveraged F_1 for different training set size.

Training	Microavg F_1 NBayesPoS	Microavg F_1 LatentMPoS	Microavg F_1 SVMPoS
10	54.4%	**57.7%**	56.0%
20	61.2%	**66.4%**	69.9%
30	66.2%	**71.9%**	73.6%
40	67.2%	**72.9%**	71.1%
50	69.8%	**74.7%**	73.8%
100	73.1%	**76.7%**	78.3%
200	77.0%	**80.3%**	80.2%
300	78.4%	**82.0%**	81.5%
400	79.1%	**81.7%**	81.0%
500	80.3%	**82.5%**	81.7%

Table 8. Sim-based vs random initialization.

EM Iteration	Sim-based Init	Random Init
1	80.5%	59.0%
2	81.5%	70.6%
3	81.9%	76.5%
4	82.2%	79.8%
5	82.3%	80.9%
10	82.5%	82.3%
15	82.5%	82.4%

Table 9. Heuristic, Heuristic & EM1, Random & EM1.

Training	Heuristic	Heuristic-EM1	Random-EM1
10	38.1%	56.8%	49.8%
20	66.6%	60.9%	49.6%
30	68.2%	67.7%	49.6%
40	40.3%	70.5%	49.8%
50	43.4%	71.7%	49.8%
100	27.3%	74.8%	49.8 %
200	29.9%	79.3%	49.8%
300	27.6%	80.8%	51.0%
400	30.4%	80.3%	51.0%
500	32.3%	80.5%	52.0%

4.4. Discussion

The results above clearly demonstrate the benefits of combining the initialization heuristic with EM; neither technique alone can achieve good performance. Further experiments are needed for a better understanding of the behavior of the proposed techniques.

5. Conclusions

In this paper, we proposed a generative language model approach to automatic document classification. Many similar models exist in the literature, but our approach is a step towards increasing the model robustness by introducing explicit information on the model and pruning the parameter space to only necessary data, encoded in the training collection. The approach proposed seems to be beneficial for collections with a rich natural language vocabulary, setups in which classical terms-only methods risk to be trapped in the semantic variations. Our future work includes more comprehensive experimental studies on various data collections and also studying the usage of different knowledge resources, such as customized ontologies extracted from large corpora.

References

Baker, L. D., & McCallum, A. (1998). *Distributional Clustering of Words for Text Classification. Proceedings of the 21st ACM-SIGIR International Conference on Research and Development in Information Retrieval* (pp. 96–103).

Bhattacharya, I., & Getoor, L. & Bengio, Y. (2004). *Unsupervised Sense Disambiguation Using Bilingual Probabilistic Models. Meeting of the Association for Computational Linguistics.*

Bloehdorn, S., & Hotho, A. (2004). *Text Classification by Boosting Weak Learners based on Terms and Concepts. International Conference on Data Mining* (pp. 331–334).

Cai, L., & Hofmann, T. (2003). *Text Categorization by Boosting Automatically Extracted Concepts. 26th Annual International ACM-SIGIR Conference.*

Chakrabarti, S. (2003). *Mining the Web: Discovering Knowledge from Hypertext Data.* San Francisco: Morgan Kaufman.

Croft, W. B., & Lafferty, J. (2003). *Language Modeling for Information Retrieval.* Kluwer Academic Publishers.

Deerwester, S., & Dumais, S. T., & Harshman, R. (1990) *Indexing by Latent Semantic Analysis. Journal of the American Society of Information Science* 41(6) (pp. 391–407).

Fellbaum, C. (1999). *WordNet: An Electronic Lexical Database.* Cambridge: MIT Press.

Hastie, T., & Tibshirani, R., & Friedman, J. H. (2003). *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* New York: Springer Verlag.

Hofmann, T. (2001). *Unsupervised Learning by Probabilistic Latent Semantic Analysis.* Kluwer Academic Publishers.

Joachims, T. (1998). *Text categorization with support vector machines: learning with many relevant features* (pp. 137–142). *Proceedings 10th European Conference on Machine Learning.*

Manning, C. D., & Schütze, H. (2000). *Foundations of Statistical Natural Language Processing.* Cambridge: MIT Press.

McCallum, A., & Nigam, K. (1998). *A Comparison of Event Models for Naive Bayes Text Classification. AAAI-98 Workshop on "Learning for Text Categorization".*

Scott, S., & Matwin, S. (1999). *Feature Engineering for Text Classification Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 379–388)

Staab, S., & Studer, R. (2004). *Handbook on Ontologies* Berlin: Springer.

Theobald, M., & Schenkel, R., & Weikum, G. (2003). *Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data. Sixth International Workshop on the Web and Databases.*

Tsochantaridis, I., & Hoffman, T., & Joachims, T., & Altun Y. (2004). *Support Vector machine Learning for Interdependent and Structured Output Spaces. Proceedings of the 21st International Conference on Machine Learning.*

Unsupervised Ontology-based Semantic Tagging for Knowledge Markup

Paul Buitelaar PAULB@DFKI.DE
Srikanth Ramaka SRIKANTH.RAMAKA@DFKI.DE
DFKI GmbH, Language Technology, Stuhlsatzenhausweg 3, 66123 Saarbruecken, Germany

Abstract

A promising approach to automating knowledge markup for the Semantic Web is the application of information extraction technology, which may be used to instantiate classes and their attributes directly from textual data. An important prerequisite for information extraction is the identification and classification of linguistic entities (single words, complex terms, names, etc.) according to concepts in a given ontology. Classification can be handled by standard machine learning approaches, in which concept classifiers are generated by the collection of context models from a training set. Here we describe an unsupervised approach to concept tagging for ontology-based knowledge markup. We discuss the architecture of this system, and our strategy for and results of performance evaluation.

1. Introduction

A central aspect of Semantic Web development is knowledge markup: annotation of data with formalized semantic metadata in order to allow for automatic processing of such data by autonomous systems such as intelligent agents or semantic web services (see e.g. McIlraith et al., 2001). As much of today's information is available as text only, knowledge markup often involves the annotation of textual data to explicitly structure the knowledge that is available in text only implicitly. Automating this process involves the use of information extraction technology that allows for the mapping of linguistic entities (single words, complex terms, names, etc.) to shallow semantic representations, mostly referred to as 'templates' (see e.g. Ciravegna, 2003). Consider for instance the following example from the football domain, which expresses a typical event with a number of roles to be filled by information extraction from relevant textual data, e.g.: *In the last minute Johnson saved with his legs from Huckerby*

```
RESCUE-EVENT [
 goalkeeper   : GOALKEEPER    > Johnson
 player       : PLAYER        > Huckerby
 manner       : BODYPART      > legs
 atMinute     : INT         ] > 90
```

Obviously, if such templates are expressed in a formally defined knowledge markup language such as RDFS or OWL, they roughly correspond to an ontologically defined class with its attributes (properties). In the context of this paper we therefore assume an interpretation of information extraction for knowledge markup as *concept instantiation*[1] that includes:

- concept tagging – mapping of linguistic entities to concepts/classes as defined by an ontology
- attribute filling – mapping of linguistic structure over linguistic entities that are tagged with a class to attributes of that class as defined by an ontology

Here we focus primarily on concept tagging, which is a prerequisite for attribute filling. We treat concept tagging as a classification task that can be handled by standard machine learning approaches, in which concept classifiers are generated by the collection of context models from a training set. Context models may be generated from manually annotated, i.e. *supervised* training sets, but this is very costly and non-robust as for each new ontology a supervised training set needs to be constructed. Instead, we present development of an *unsupervised* approach that can be trained on any relevant training data, without previous manual annotation.

Appearing in *W4: Learning in Web Search at 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

[1] Concept instantiation has also been referred to as 'ontology population' (e.g. in the context of the AKT project - http://www.aktors.org/akt/), which emphasizes the database aspect of an ontology and its corresponding knowledge base.

This is similar to the SemTag approach to large-scale semantic tagging for the Semantic Web (Dill et al., 2003), but the emphasis of our approach is somewhat different. We focus here on an unsupervised approach to concept tagging as a necessary prerequisite for further information extraction and more complex knowledge markup, whereas the SemTag approach emphasizes the large-scale aspects of concept tagging without a clear vision on the eventual use of the added semantic tags.

The remainder of the paper gives an overview of the system architecture of our approach in section 2, followed in section 3 by a discussion of our evaluation strategy and results of this. In section 4 we give an outline of the application of the system in two Semantic Web projects. Related work is presented in section 5.

2. System Architecture

The unsupervised concept tagging system we are developing consists of the following components:

- a set of hierarchically organized classes from a domain ontology
- a domain-relevant document collection for training and classification
- a shallow linguistic module for preprocessing class labels and documents
- a machine learning environment for generating context models and classifiers
- a knowledge base to store marked up concept instantiations

In the training phase, a context model and classifier is generated from a domain-specific document collection for a set of classes from a corresponding domain ontology, over which various parameters are evaluated to select the best classifier. In the application phase, the classifier is used in tagging linguistic entities with the appropriate class and to store corresponding class instances in the knowledge base. In information extraction, these instances (with linguistic contexts) are submitted to a further process that maps them to relevant class attributes. We will not address this any further here, but applications of the information extraction process are discussed in section 4.

2.1 Ontology and Document Collection

The system assumes as primary input an ontology in RDFS or OWL with a hierarchy of classes as specified for a particular domain. The following two example classes from the "Soccer V2.0" ontology[2] on football express two events ('to clear' and 'counter attack') that are defined as sub-classes of a class that expresses the more general event 'other player action'[3]:

```
<rdfs:Class rdf:ID="Clear">
 <rdfs:subClassOf
  rdf:resource="#Other_player_action"/>
  <rdfs:label
   xml:lang="en">Clear
  </rdfs:label>
  <rdfs:label
   xml:lang="de">Klären
  </rdfs:label>
</rdfs:Class>

<rdfs:Class rdf:ID="Counter_attack">
 <rdfs:subClassOf
  rdf:resource="#Other_player_action"/>
  <rdfs:label
   xml:lang="en">Counter_attack
  </rdfs:label>
  <rdfs:label
   xml:lang="de">Konterangriff
  </rdfs:label>
</rdfs:Class>
```

Next to a domain ontology, the system assumes a document collection on the same domain. For instance, for the SmartWeb project[4] that will be discussed in Section 4 below, we are working with a football ontology and a document collection on UK football matches[5].

2.2 Linguistic Preprocessing

In order to map linguistic entities in the document collection on classes in the ontology, we normalize them into a common linguistic representation. For this purpose we linguistically preprocess the class names in the ontology as well as all text segments in the document collection.

Linguistic preprocessing[6] includes part-of-speech (PoS) tagging with the TnT tagger (Brants, 2000) and lemmatization based on Mmorph (Petitpierre and Russell, 1995). Part-of-speech tagging assigns the correct syntactic class (e.g. noun, verb) to a particular word given its context. For instance, the word *works* will be either a verb *(working the whole day)* or a noun (*all his works have been sold).*

---

2 Available from http://www.lgi2p.ema.fr/~ranwezs/ontologies/soccerV2.0.daml, which we adapted to OWL and added German labels.

3 We use the OWL API (Bechhofer et al., 2003) in parsing the ontology.

4 More information on the SmartWeb project can be obtained from http://www.smartweb-projekt.de

5 The football document collection used here is obtained by crawling a web portal on premiere league football in the UK: http://4thegame.com

6 Linguistic preprocessing is accessed via an XML-based format based on proposals in (Buitelaar and Declerck, 2003).

Lemmatization involves normalization over inflectional, derivational and compound information of a word. Inflectional information reduces the plural noun *works* to the lemma *work*, whereas derivational information reduces the verb forms *working* and *works* to the lemma *work*. Compound information determines the internal structure of a word. In many languages other than English the morphological system is very rich and enables the construction of semantically complex compound words. For instance the German word "*Schiedsrichterfahne*" corresponds in English with two words "*referee flag*".

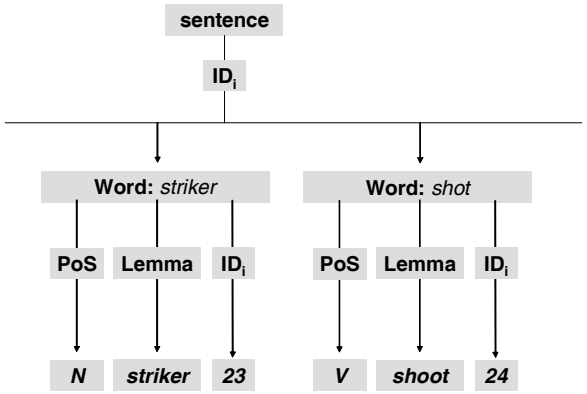

Figure 1: Linguistic Annotation Example

2.3 Generating Context Models and Classifiers

The concept tagging system is based on an instance-based learning approach to classification as implemented for instance in the WEKA machine learning environment. Instance-based learning involves a nearest neighbor classification method, in which the instance to be classified *i* is compared with all training instances, using a distance metric, and the closest training instance is then used to assign its class to *i.* The generalization of this method that we use here is the *k*-nearest neighbor method, where the class of the instance *i* is computed using the closest *k* training instances.

An instance-based learning algorithm consists of a training step and an application step. We first discuss the training step, in which context models and corresponding classifiers are generated. In the next sub-section we discuss the application of such classifiers in concept tagging.

Training involves the construction of classified instances from a training set. As the methods discussed here are unsupervised, this training set has not been previously annotated. An instance is a set of attribute-value pairs, one of which identifies the class that needs to be determined.

Constructing an instance involves the following. Let *w* be a word in the training set, for which we can build instances with the attribute-value pairs of each instance filled by its left and right neighbor words in a context of size *N*. The attribute-value pair that represents the class of this instance is filled by matching the word *w* with the preprocessed class name and the class names of all of its sub-classes. To illustrate the construction of particular instances, consider the following sentences from the document collection on football:

Even during those early minutes Palace's former Carlisle ***attacker*** *Matt Jansen looked up for a big game, and no wonder as he was facing his boyhood idols!*

Arsenal's new French ***midfielder*** *Patrick Vieira started the rot for Leeds this time after only 44 seconds.*

That they went home empty-handed was largely down to another of Gullit's instant imported hits, former Strasbourg ***sweeper*** *Frank Leboeuf.*

The words *attacker, midfielder, sweeper* match with the classes **`attacker`**, **`midfielder`**, **`sweeper`** in the football ontology, which are sub-classes of the class **`player`**. From the sentences we may now derive the following instances for this class with context size 5 (2 words on the left, 2 words on the right):

N-2	N-1	N+1	N+2
former	Carlisle	Matt	Jansen
new	French	Patrick	Vieira
former	Strassbourg	Frank	Leboeuf

In this way, we can build up a context model and corresponding classifier for each class. In the application phase these classifiers will be used to classify unseen terms. Consider for instance the word *striker* in the following sentence:

The big French ***striker*** *stepped up to drill home the penalty himself.*

The word *striker* (in this context) expresses the sub-class **`striker`** of the class **`player`**, which has not been modeled as such in the football ontology. We therefore can use classification to extend the coverage of the concept tagging system and at the same time to acquire additional sub-classes for each of the classes modeled in the training step. In this way, knowledge markup can be connected to ontology learning, which aims at automatic or semi-automatic extension and/or adaptation of ontologies[7].

[7] See the collection of papers from the ECAI04 workshop on Ontology Learning and Population for an overview of recent work http://olp.dfki.de/ecai04/cfp.htm.

2.4 Classification: Concept Tagging

In the application step, we use the generated classifiers to classify an occurrence of word *w* by finding the *k* most similar training instances. For instance, for the sentence with *striker* above, we extract the corresponding instance to be classified (with the class missing):

```
[big, French, stepped, up, -]
```

Now we classify the instance using the generated classifiers to obtain:

```
[big, French, stepped, up, player]
```

The output of this process is a classified instance that will be represented in two ways:

- Concept Tagging – mark up of corresponding tokens in the document with the assigned class in XML[8]
- Knowledge Base Instantiation – generation of an RDF instance for the assigned class in the ontology (with a pointer to corresponding tokens in the document)

To illustrate this, consider the example in Figure 2 below. Here, the word *striker* is marked as **player** with an indication of the origin of this class through the information stored in the `ontology` attribute. An instance in RDF can be created accordingly and stored in the knowledge base.

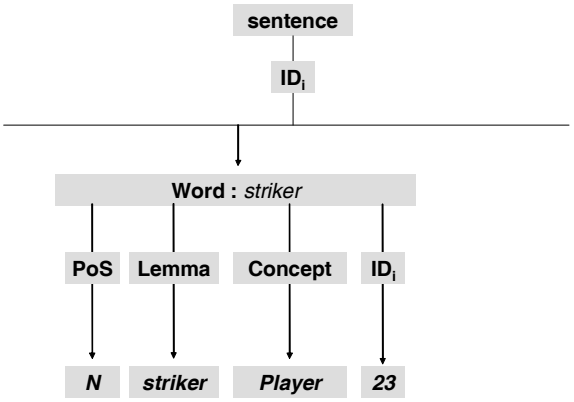

Figure 2: Concept Tagging Example

3. Evaluation

An important step in system development is performance evaluation, in order to determine the appropriate research direction for the task at hand. In the context of this paper we were interested to determine an answer to the following research questions:

1. How well does the system perform on correctly classifying new terms (i.e. terms that are not yet modeled in the ontology)?
2. What is the influence of linguistic preprocessing (PoS tagging, lemmatization) on classification results?

In this section we discuss our strategy in evaluating these questions, the evaluation set we constructed and the results obtained with this evaluation set.

3.1 Evaluation Strategy

To evaluate our approach we developed a performance evaluation strategy that assumes a gold standard with which different data sets can be automatically compared and on the basis of which recall and precision numbers can be computed in a straightforward way. A major criticism of performance evaluation is that it evaluates only the clearly measurable aspects of the technology used, without taking the wider user-oriented context into account. Although this is surely correct from a wider user-oriented perspective, for comparing results on many different parameters there seems to be no alternative to the use of a gold standard. We therefore developed a gold standard classification set for the football domain, derived from the document collection and football ontology mentioned earlier.

3.2 Evaluation Sets

The gold standard was constructed by pooling: running the system with the same data set over a number of different parameters (context size, value of *k*). We then merged the resulting classified data sets by taking the intersection of all classified instances. This resulted in an evaluation set of 869 classified instances that we gave to three evaluators to judge on correctness[9]. The task of the evaluators was to judge if a word *w* was correctly classified with class *c*, given its context (sentence) *s*. The classified instances were presented to the evaluator as follows:

c: **other_player_action**

w: *volleying*

s: *Wiltord fed the ball through to Dennis Bergkamp and his chip into Henry's path led to the French striker volleying over from six yards when it appeared easier to score.*

The evaluators were then asked to judge this classification by assigning it a 3 (very good), 2 (good), or 1 (incorrect). We were able to assemble a gold standard from these

8 Concept tagging extends the XML output of linguistic preprocessing as discussed in section 2.2 (see also Buitelaar and Declerck, 2003)

9 The evaluators qualified as `domain experts` as they were all football aficionados.

judgments by taking a voting account of the three assignments for each classified instance. For 863 instances a majority could be established in this way, for the remaining 6 instances each evaluator assigned a different score. These instances were therefore left out of the resulting gold standard.

The 863 instances in the gold standard are distributed over 4 classes in the football ontology that we selected for evaluation:

other_player_action with sub-classes: **beat, charge, clear, ...**

person with sub-classes: **official, player, ...**

place with sub-classes: **area, field, line, ...**

stoppage with sub-classes: **corner, fault, goal, ...**

The distribution of judgments over these classes is as follows:

Table 1: Distribution of judgments over the 4 selected classes

	very good	*good*	*incorrect*
other_player_action	47	32	104
person	50	4	57
place	24	14	118
stoppage	4	2	407
Total	125	52	686

From the set of evaluated instances we then created two gold standard evaluation sets, a "strict" one (including only the instances judged to be classified "very good") and a "relaxed" one (including the "very good" as well as the "good" instances). The "strict" set has 125 and the "relaxed" set 177 instances.

3.3 Evaluation Results

We used the two gold standard sets to evaluate different settings for *N* (context size) and the number of closest *k* training instances. To evaluate the influence of context size we varied *N* between 1, 2 and 5, each time with *k* between 1, 2 and 10. The results are presented in the following tables.

The results in table 2 show that a larger context size degrades recall significantly as we consider only contexts within sentence boundaries. Obviously, there are more n-grams of length 3 (*N=1*) than of length 11 (*N=5*) within a sentence. The influence of *k* seems not significant, although *k=1* gives the best results at *N=1*.

Table 2: Evaluation results

N	*k*	*Strict Set*		*Relaxed Set*	
		Rec.	*Prec.*	*Rec.*	*Prec.*
1	**1**	89% (111)	89% (99)	92% (162)	89% (144)
	2	89% (111)	87% (97)	92% (162)	87% (141)
	10	87% (109)	83% (90)	89% (158)	84% (132)
2	**1**	69% (86)	83% (71)	66% (117)	82% (96)
	2	66% (82)	85% (70)	64% (114)	82% (94)
	10	66% (83)	84% (70)	65% (115)	82% (94)
5	**1**	17% (21)	81% (17)	18% (31)	81% (25)
	2	15% (19)	84% (16)	16% (29)	76% (22)
	10	14% (18)	83% (15)	15% (27)	81% (22)

The results in table 2 provide an answer to our first research question (how well do we classify?). The answer to the second question (does linguistic preprocessing matter?) is given by the results in the following table. In this case we did not use any linguistic preprocessing in training and application. As the table shows, the results are worse than with linguistic preprocessing (only results for *N=1* are shown).

Table 3: Evaluation results – no linguistic preprocessing

N	*k*	*Strict Set*		*Relaxed Set*	
		Rec.	*Prec.*	*Rec.*	*Prec.*
1	**1**	74% (92)	85% (78)	76% (135)	84% (113)
	2	74% (92)	83% (76)	76% (135)	81% (110)
	10	74% (92)	79% (73)	75% (132)	79% (104)

4. Application

The concept tagging system described in this paper is being developed in the context of two projects (SmartWeb, VieW*s*) that we are currently working on. The projects have different scenarios and application domains, but share a need for tagging of text documents with classes from a given ontology for information extraction purposes.

4.1 SmartWeb

SmartWeb is a large German funded project that aims at intelligent, broadband mobile access to the Semantic Web. For this purpose it combines such diverse technologies as speech recognition, dialogue processing, question answering, information extraction, knowledge management and semantic web services into an ambitious common framework to realize an intelligent mobile information system.

A first demonstrator is targeted to the football world cup 2006, which will be held in Germany. The SmartWeb system will be able to assist the football fan over speech input in booking his tickets for the games he wants to see, as well as hotels, restaurants, etc. Additionally, the system will be able to answer questions on any football related issue (e.g. game history, end scores, names and achievements of players) or otherwise (e.g. the weather, local events, news).

In order to be able to answer such questions, the system will need to have knowledge of many topics which will be handled by a combination of several technologies: open-domain question answering on the web (based on an information retrieval approach), semantic web service access to web-based databases and ontology-based information extraction from football related web documents for knowledge base generation. Concept tagging with the SmartWeb football ontology is a prerequisite for the ontology-based information extraction task.

4.2 VieW*s*

The VIeW*s*[10] project has as its central aim to demonstrate how web portals can be dynamically tailored to special interest groups. The VIeW*s* system combines ontologies, information extraction, and automatic hyperlinking to enrich web documents with additional relevant background information, relative to particular ontologies that are selected by individual users. A tourist for instance will be shown additional information on hotels, restaurants or cultural events by selecting the tourist ontology.

On entering a VIeW*s* enhanced web portal the system analyses the web document provided by the server and identifies anchors for the hyperlinks, e.g. city names. A Google-based web search is then started for the recognized city names in combination with keywords ("hotel", "restaurant", etc.) derived from the ontology.

The results of the web search and information already existing in the knowledge base will be shown in the form of generated hyperlink menus on each of the identified city names. Additionally, an information extraction process is started in the background over the retrieved documents and relevant extracted information is stored in the knowledge base for future access. Obviously also here ontology-based concept tagging is a prerequisite for the information extraction process.

5. Related Work

As mentioned before, the work discussed here is related to the SemTag work on large-scale semantic tagging for the Semantic Web (Dill et al., 2003). Also much of the work on semantic annotation (for a recent overview see: Handschuh and Staab, 2003) and ontology learning (for a recent overview see: Buitelaar et al., 2005) for the Semantic Web is directly related. However, next to this also various other tasks in natural language processing and information retrieval are concerned with similar issues.

First of all, the large body of work on semantic tagging and word sense disambiguation is of direct interest as this is also concerned with the assignment of semantic classes to words (for an overview see Ide and Veronis, 1998; Kilgarriff and Palmer, 1999; Edmonds and Kilgarriff, 2003). However, there is also an important difference as this work has been almost exclusively concerned with the use of lexical resources such as dictionaries or wordnets for the assignment of semantics to words in text. The use of ontologies brings in a rather different perspective, e.g. on lexical ambiguity, on lexical inference and on the mapping of linguistic structure to semantic structure.

A second important area of related work is named-entity recognition (for a recent overview see e.g. Tjong Kim Sang and De Meulder, 2003). Named-entity recognition (NER) is also concerned with the assignment of semantic classes to words or rather names in text. However, the typical number of semantic classes used in NER is mostly small, not extending beyond distinctions such as person, location, organization, and time. Nevertheless, there is an important overlap in the methods and goals of NER and the work discussed here, that is if we imagine NER with a larger and hierarchically ordered set of semantic classes as specified by an ontology. Such a direction in NER has been given much consideration lately, as witnessed for instance by the SEER[11] (Stanford Edinburgh Entity Recognition) project.

[10] http://views.dfki.de

11 http://www.ltg.ed.ac.uk/seer/

6. Conclusions

We presented ongoing work on developing an ontology-based concept tagging system as an important prerequisite in information extraction for knowledge markup. The system we discussed implements an unsupervised approach, in which no prior manual tagging is needed. Such an approach allows for a robust application of the system in different domains. Evaluation indicates that good results can be obtained with such an approach and that linguistic preprocessing helps to increase recall and precision.

Acknowledgements

This research has been supported by grants for the projects VIeWs (by the Saarland Ministry of Economic Affairs) and SmartWeb (by the German Ministry of Education and Research: 01 IMD01 A).

References

Bechhofer Sean, Phillip Lord, Raphael Volz. *Cooking the Semantic Web with the OWL API.* 2nd International Semantic Web Conference, ISWC, Sanibel Island, Florida, October 2003.

Brants, Thorsten. *TnT - A Statistical Part-of-Speech Tagger*. In: Proceedings of 6th ANLP Conference, Seattle, 2000.

Buitelaar, Paul and Thierry Declerck. *Linguistic Annotation for the Semantic Web.* In: Handschuh S., Staab S. (eds.) Annotation for the Semantic Web, IOS Press, 2003.

Buitelaar, Paul, Philipp Cimiano and Bernardo Magnini (eds.) Ontology Learning from Text: Methods, Evaluation and Applications. IOS Press, 2005.

Ciravegna, Fabio. *Designing adaptive information extraction for the semantic web in amilcare*. In Siegfried Handschuh and Steffen Staab, editors, Annotation for the Semantic Web, Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, 2003.

Dill S., N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien, *SemTag and Seeker: Bootstrapping the semantic Web via automated semantic annotation*, 12th International World Wide Web Conference Budapest, Hungary, 2003.

Edmonds, Phil and Adam Kilgarriff (eds.). Journal of Natural Language Engineering (*special issue based on Senseval-2*), vol.9 no. 1, Jan. 2003.

Handschuh, Siegfried and Steffen Staab (eds.) Annotation for the Semantic Web. IOS Press, 2003.

Ide, N. and Veronis J. *Introduction to the special issue on word sense disambiguation: The state of the art.* Computational Linguistics, 24(1):1--40. 1998.

Kilgarriff, Adam and Martha Palmer (eds.). Computers and the Humanities (*special issue based on Senseval-1*), vol.34 no. 1-2, 1999.

McIlraith, Sheila A., Tran Cao Son, and Honglei Zeng *Semantic Web Services* IEEE Intelligent Systems, March/April 2001, Vol 16, No 2, pp. 46-53.

Petitpierre, D. and Russell, G. *MMORPH - The Multext Morphology Program.* Multext deliverable report for the task 2.3.1, ISSCO, University of Geneva. 1995.

Tjong Kim Sang, Erik F. and Fien De Meulder. 2003. *Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition.* In: Walter Daelemans and Miles Osborne (eds.), Proceedings of CoNLL-2003, Edmonton, Canada.

Generating Accurate Training Data from Implicit Feedback (Invited Talk)

Thorsten Joachims

Cornell University, Department of Computer Science, USA

Abstract: Machine learning methods have shown much promise in designing adaptive and personalized information system, ranging from email readers to search engines. But how can we generate the data to train these systems? The availability of training data is the crucial bottleneck in many of these applications, since generating training data manually is time consuming and often goes beyond the user's willingness to participate. To overcome this bottleneck, researchers have tried to infer training data from observable user behavior. Such implicit feedback can be collected at low cost and in huge quantities, but does it provide valid training data?

In this talk, we propose and analyze strategies for generating training data from observable user behavior. Focusing on clickthrough data in web search, we conducted an eye-tracking study to analyze the relationship between user behavior and the relevance of a page. The study shows that a particular interpretation of clickthrough data provides reliable training data. While clicks do not indicate the relevance of a page on an absolute scale, clicks accurately indicate relative training data of the kind "for query Q, document A should be ranked higher than document B".

Topic-Specific Scoring of Documents for Relevant Retrieval

Wray Buntine, Jaakko Löfström, Sami Perttu and Kimmo Valtonen FIRST.LAST@HIIT.FI
Helsinki Inst. of Information Technology
P.O. Box 9800, FIN-02015 HUT, Finland

Abstract

There has been mixed success in applying semantic component analysis (LSA, PLSA, discrete PCA, etc.) to information retrieval. Here we combine topic-specific link analysis with discrete PCA (a semantic component method) to develop a topic relevancy score for information retrieval that is used in post-filtering documents retrieved via regular Tf.Idf methods. When combined with a novel and intuitive "topic by example" interface, this allows a user-friendly manner to include topic relevance into search. To evaluate the resultant topic and link based scoring, a demonstration has been built using the Wikipedia, the public domain encyclopedia on the web.

1. Introduction

More sophisticated language models are starting to be used in information retrieval (Ponte & Croft, 1998; Nallapati, 2004) and some real successes are being achieved with their use (Craswell & Hawking, 2003). A document modelling approach based on discrete versions of principal components analysis (PCA) (Hofmann, 1999; Blei et al., 2003; Buntine & Jakulin, 2004) has been applied to the language modelling task in information retrieval (Buntine & Jakulin, 2004; Canny, 2004). However, it has been shown experimentally that this is not necessarily the right approach to use (Azzopardi et al., 2003). The problem can be explained as follows: when answering a query about "computing entropy," a general statistical model built on the full Wikipedia, for instance, often lacks the fidelty on these two key words combined. In the language of minimum description length, it is wasting its bits across the full spectrum of words, instead of conserving bits for the only two words of real interest. Ideally, one would like a statistical model more specifically about "computing entropy," if it were feasible. Thus the statistically based language modelling approach to information retrieval is stll needing of development.

Thus, arguably, supervised models are needed for information retrieval. Here we take an alternative path for using statistical models in information retrieval. Our approach is motived by the widespread observation that people would like to be able to bias their searches towards specific areas, but they find it difficult to do so in general. Web critics have reported that Google, for instance, suffers perceived bias in some searches because of the overriding statistics of word usage in its corpus ("the web") in contrast with their dictionary word senses (Johnson, 2003): on the internet an "apple" is a computer, not something you eat, "Madonna" is an often-times risque pop icon, not a religious icon, and moreover "latex" is not a typesetting system, but apparantly something the certain people where in certain situations. Thus one might want to use a keyword "Madonna" but bias the topic somehow towards Christianity in order to get the religious word sense.

A major user interface problem here is that people have trouble navigating concept hierarchies or ontologies (Suomela & Kekäläinen, 2005), especially when they are unfamiliar with them. Even when they are familiar with them, a point and click menu on a 200-topic hierarchy is unwieldy. This is further confounded because good topic hierarchies and ontologies are usually multifaceted, and search might require specifying multiple nodes in the hierarchy.

To address this problem, we apply machine learning and statistical inference technology in a novel combination.

Topic by example: Users do not have to know the hierarchy, or browse it, or navigate multiple paths to get multiple facets for their search. They just enter a few words describing their general topic

Appearing in *W4: Learning in Web Search, at the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

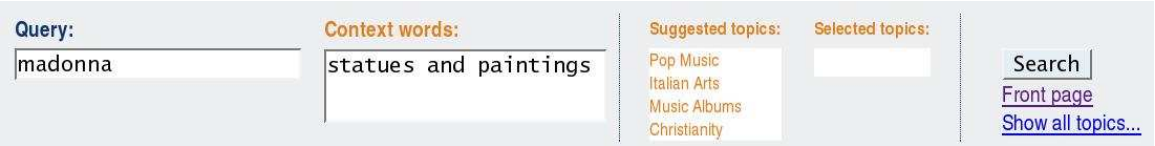

Figure 1. The search options on the results page

area in a "context words" box and let the system work out the topics "by example". An example of the input screen is shown in Figure 1. Topics can then be used on masse or selected individually.

Topic specific page-rank: Many pages can be topically relevant, but when dealing with a specific topic area or combination of topic areas, which pages are considered the most important in terms of topically relevant citations? Topic specific versions of page rank (Haveliwala, 2002; Richardson & Domingos, 2002) address this.

Result filtering: The top results from a regular Tf.Idf query are reranked using a weighted combination of topic-specific page rank. In this way, the choice of topic "by example" affects the results but in a computationally feasible and scalable manner.

Here we first apply the discrete PCA method to develop topics automatically. This gives topics suitable for the corpus, and a multi-faceted classification of all pages in it. We then apply these using a topic-specific version of page rank (Richardson & Domingos, 2002) that is based on the notion of a random surfer willing to hit the back button when a non-topical page is encountered. This gives topic specific rankings for pages that can be used in the topic-augmented search interface.

Our intent is that these techniques yield a secondary topical score for retrieval in conjunction with a primary key-word based score such as Tf.Idf. Thus relevance of a document is a combination of both keyword relevance and topical relevance. Because search users are usually daunted by anything more than just a keyword box, and because keyword search currently works quite well, our default is to make the keyword entry and the topical entry equivalent initially in a search, and only give the option to change the topic, as shown in Figure 1, after a first batch of results have been returned. Thus the initial search screen contains no "context words" box.

Our platform for experiments with these methods is the English language part of the Wikipedia[1], an open source Encyclopedia. This has a good internal link structure and about 500,000 pages, so it is a reasonable sized test. The system is demonstrated at our test website `http://kearsage.hiit.fi/wikisearch.html`[2].

The combination of topic-specific and link-based scoring is fundamental, we believe, to the success of this method. Topic-based scoring alone can return documents with high topical scores, but they are not "characteristic" documents for the topic and keyword combination, rather they are "typical". A document with high topical content is not necessarily characteristic. For instance, entering the query "madonna" gives the following pages titles as top results under a standard OKAPI BM25 version of Tf.Idf, under Google, and under our system ("Topical filtering"). These are listed in rank order:

Tf.Idf: Madonna (entertainer), Unreleased Madonna songs, List of music videos by year work in progress, Bedtime Stories (Madonna), American Life

Google: Madonna (entertainer), Madonna (singer), Madonna, Unreleased Madonna Songs, Black Madonna

Topical filtering: Madonna, Madonna (entertainer), Unreleased Madonna songs, The Madonna, American Life

Tf.Idf essentially returns documents with many instances of the word Madonna. Google essentially returns documents voted by web-links as being most important, mostly Madonna the entertainer. Our approach sees Madonna is a word with both entertainment and religious connotations, and returns important documents with a better topical mix. "Madonna" in this case is the main disambiguating page that points to the different versions of Madonna. It becomes the highest ranked using our topical filtering

[1] `http://en.wikipedia.org`

[2] The website is being used to test interface concepts as well as perform user studies, thus its performance is not robust.

due to it being a better topical match to the query. Another example is the query "stars".

Tf.Idf: List of The Simpsons episodes, List of stars on the Hollywood Walk of Fame, Star Wars, Star Trek, List of stars by constellation, Star, Star Trek Other Storylines

Google: Star, Neutron star, Flag of the United States, Movie star, List of nearest stars, Stars and Stripes, List of brightest stars

Topical filtering: Star system, Star (glyph), Star Trek Further Reading, Star (disambiguation), Star Wreck, Star, List of LucasArts Star Wars games

In this case, "Star (glyph)" is the mathematical concept of a star. In this case, the disambiguation page is only seen in the results from topical filtering, as well as a broader range of topical versions of star.

This paper first presents the background on discrete PCA (DPCA), and topic specific ranking using a topically motivated random surfer. Then the combination of these methods is described. The paper described the results of the topic specific ranking, a very appealing and rational set of document rankings for different topics. Finally the application of these techniques to information retrieval are discussed and presented.

2. Background

2.1. Topic Specific Ranking

We use the term "random surfer model" in a broad sense: to encompass general Monte Carlo Markov chain methods, modelling eye-balls on pages, used to determine scores for documents. Examples are (Haveliwala, 2002; Richardson & Domingos, 2002). A general method for topic-specific ranking roughly following ((Richardson & Domingos, 2002) goes as follows:

Our surfer restarts with probability α at a page i with probability r_i. From that page, they uniformly select a link to document i', and jump to this next page. They then consider the topic of the new page, whose strength of relevance is determined by another probability $t_{i'}$. With this probability $t_{i'}$ they accept the new page, and with probability $1 - t_{i'}$ they go back to the page i to try a new link. The stationary distribution of the Markov Chain for the probability of being on page p_i is then given by the update equations:

$$p_i \longleftarrow \alpha r_i + (1-\alpha) \sum_{i' \,:\, i' \to i} p_{i'} \frac{t_i}{\sum_{j \,:\, i' \to j} t_j}$$

where we perform the calculation only for those pages i with $r_i > 0$, and $i' \to i$ denotes page i' links to page i. The vectors $\vec{r}$ and $\vec{t}$ allow specialization to a topic,so a set of such rankings $\vec{p}$ can be developed for every topic: $\vec{r}$ represents the starting documents for a topic and $\vec{t}$ represents the probability that someone interested in the topic will stay at a page.

Note that previous applications of this technique have been hampered because full multifaceted topical assignments for documents have not been available. Hence we apply discrete PCA to obtain a rich set of multifaceted topics.

2.2. Discrete PCA

Principal component analysis (PCA), latent semantic indexing (LSI), and independent component analysis (ICA) are key methods in the statistical engineering toolbox. They have a long history and are used in many different ways. A fairly recent innovation here is discrete versions: genotype inference using admixtures (Pritchard et al., 2000), probabilistic latent semantic indexing (Hofmann, 1999) latent Dirichlet allocation (Blei et al., 2003), discrete PCA (Buntine & Jakulin, 2004) and Gamma-Poisson (GaP) models (Canny, 2004) are just a few of the known versions. These methods are variations of one another, ignoring statistical methodology and notation, and form a discrete version of ICA (Buntine, 2005; Buntine & Jakulin, 2004; Canny, 2004).

Each document is represented as an integer vector, $\vec{w}$, usually sparse. The vector may be as simple as bag of words, or it may be more complex, separate bags for title, abstract and content, separate bags for nouns and verbs, etc. The model also assigns a set of independent components to a document somehow representing the topical content. In the general Gamma-Poisson (GaP) model (Canny, 2004) the k-th component is a Gamma(α_k, β_k) variable. In multinomial PCA or LDA it is a Gamma$(\alpha_k, 1)$ variable, but then the set of variables is also normalized to yield a Dirichlet (Buntine & Jakulin, 2004). Finally, component distributions complete the model: each component k has proportion vector $\vec{\Omega}_k$ giving the proportion of each word/lexeme in the vector $\vec{w}$, where $\sum_j \Omega_{j,k} = 1$. The distribution for document $\vec{w}$, is then given using hidden components $\vec{m}$ and model parameters $\vec{\Omega}$:

$$\begin{aligned} m_k &\sim \text{Gamma}(\alpha_k, \beta_k) && \text{for } k = 1, ..., K \\ w_j &\sim \text{Poisson}\left(\sum_k \Omega_{j,k} m_k\right) && \text{for } j = 1, ..., J \end{aligned}$$

Alternatively, the distribution on $\vec{w}$ can be represented

using the total count of $\vec{w}$, $w_0 = \sum_k w_k$, as:

$$w_0 \sim \text{Poisson}\left(\sum_k m_k\right)$$
$$\vec{w} \sim \text{multinomial}\left(\sum_k \frac{\vec{\Omega}_k m_k}{\sum_k m_k}, w_0\right)$$

If $\beta_k = \beta$ is constant as in LDA then this normalized $\vec{m}$ is a Dirichlet and the totals safely ignored.

The family of models can be fit using mean field, maximum likelihood, Gibbs estimation, or Gibbs estimation using Rao-Blackwellization (Buntine, 2005). Experiments reported here use the MPCA suite of software which integrates topic specific ranking and topic estimation into a server[3].

2.3. Setting up Topic Specific Ranking

Topic specific page rank can work off the normalized component values $m^*_k = m_k / \sum_k m_k$ for each document. For documents $i = 1, ..., I$, let these be $m^*_{i,k}$. The restart vector $\vec{r}$ for topic k can be given by $r_i = m^*_{i,k} / \sum_i m^*_{i,k}$. The topic relevance is more complicated. In general in discrete PCA, most pages may have a mix of topics with perhaps 5-10 different topics or components occurring for one document. Thus a document with $m^*_k = 0.2$ in these cases can be said to have the relevant topical content, since we rarely expect much more than 0.2. Thus, to derive the page relevance vector $\vec{t}$ from discrete PCA, we put the $m^*_{i,k}$ through a scaled tanh function so that when $m^*_{i,k} = 0.2$, t_i will already be near 1.

3. Experiments: Sample Rankings

We downloaded the Wikipedia in April 2005. It has approximately 513,000 documents with over 2.5Gb of text, and a rich link structure. The lexicon of the top 310,000 nouns, 13,000 verbs, 31,000 adjectives and 3,500 adverbs are used in training. Words with less than 4 occurrences in the corpus are ignored. Words are stemmed and sorted this way because it greatly improves interpretability of the model.

We ran discrete PCA using Pritchard *et al.*'s Gibbs algorithm (Buntine, 2005). with $K = 100$ components with Gamma$(1/50, 1)$ priors, and using Jeffreys' prior for the component proportions Ω_k (Dirichlet with a constant vector of $1/2$ for the parameters). This uses the MPCA software using a 800 cycle burn-in and 200 recording cycles, about 34 hours on a dual 3GHz CPU under Linux. Note that this sized corpus could easily support upto $K = 1000$ component model, but in this experiment we have chosen to limit the complexity of the search engine. Computing the set of 100 topic specific ranks for the documents takes 20 minutes using a naive algorithm with no handling of sparsity.

We compared some typical URLs (those with a high topic proportion) with those having a high rank for the topic in Table 1. A complete set of results for all components on this experiment can be viewed at our website[4]. Each topic has its own web page, accessed by clicking on the numbers, and document topic-specific rankings are given at the bottom of these pages. The difference between the typical titles (those for documents with a high topic proportion) and high-ranked titles is stark. High-ranked titles clearly describe the topic. Typical titles just give examples. For this reason, we believed that these topic-specific rankings could be used effectively in a search engine.

4. Using Discrete PCA in Information Retrieval

PLSI introduced by (Hofmann, 1999) was first suggested as an approach for information retrieval, and the GaP model has also been applied here by (Canny, 2004). The general method for applying it is the so-called language modelling approach to information retrieval of (Ponte & Croft, 1998). This goes as follows: one develops a statistical model for each document, denote the model for the i-th document by $\mathcal{D}_i$. Under this model, one can pose questions such as, what is the probability that query words $\vec{q}$ would also be added to the document? This is $p(\vec{q} \,|\, \mathcal{D}_i, \mathcal{M})$. where the model construct $\mathcal{M}$ specifies the form used. This approach then looks to retrieve the document i maximising this probability.

The effort then is placed in the development of the so-called language models which are depend on individual documents $\mathcal{D}_i$. This needs to be a very flexible model because it needs to work for any smaller query set $\vec{q}$. (Azzopardi et al., 2003) have shown that high perplexity general models, ones with high values for $p(\mathcal{D}_i \,|\, \mathcal{M})$, are not always useful for information retrieval. We conjecture that a significant part of this may be that high perplexity models are not necessarily good at predicting individual words. That is, while the quality of $p(\mathcal{D}_i \,|\, \mathcal{M})$ can be good, and experiments show this is the case for discrete PCA (Hofmann, 1999), it does not imply that the quality of $p(\vec{q} \,|\, \mathcal{D}_i, \mathcal{M})$ will follow.

[3] Available at the code website `http://cosco.hiit.fi/search/MPCA`.

[4] See the *topic browser* at the demonstration Wikipedia search engine.

Common nouns	Typical titles	High-ranked titles
Star, Earth, Moon, Sun, planet, objects, astronomer, Galaxy, asteroids	204 Kallisto, 217 Eudora, 228 Agathe, 266 Aline, 245 Vera, 258 Tyche, 219 Thusnelda	Astronomy, Earth, Sun, Moon, Star, Asteroid, Astronomer
language, word, English, IPA, name, Unicode, dialect, letter, span	List of consonants, List of phonetics topics, Digraph (orthography), Table of consonants, Ubykh phonology, Code page 855,	IPA chart for English English language, Language, Latin, Linguistics, Greek language, French language, International Phonetic Alphabet
theory, term, example, people, philosophy, time, idea, work, World	Incommensurability, Qualitative psychological research, Social constructionism, Culture theory, Internalism and Externalism, Ethical egoism, Positive (social sciences)	Philosophy, Psychology, Mathematics, Economics, Science, Biology, Physics
music, composer, instruments, opera, song, piano, Orchestra, work, Symphony	Piano quintet, String quintet, List of atonal pieces, List of pieces which use the whole tone scale, String trio, Piano sextet, Trio sonata	Music, Composer, Opera, Musical instrument, Classical music, Jazz, Piano
mythology, God, goddess, son, deities, Greek mythology, Norse, name, myth	Tethys (mythology), Uranus (mythology), Oceanid, Psamathe, Phorcys, List of Greek mythological characters, Galatea (mythology)	Greek mythology, Mythology, Norse mythology, Polynesian mythology, Roman mythology, Zeus, Homer

Table 1. A sample of components

Information retrieval applied to a large news corpus should really build a model relevant to the two words "computing entropy", or another two words "molecular biology", not to the whole corpus in one go. The minimum description length intuition is that bits used to describe the general model are wasted for the specific task.

Traditionally, language modeling has achieved reasonable performance by a compromise. The probability of a word q_j in a query is usually obtained by smoothing the model probability $p(q_j \,|\, \mathcal{D}_i, \mathcal{M})$ with the observed frequency of the word in the document itself. Suppose the frequency of the word q_j in the i-th document is $\widehat{p}(q_j \,|\, \vec{w}_i)$, then use the probability

$$\alpha\, p(q_j \,|\, \mathcal{D}_i, \mathcal{M}) + (1-\alpha)\, \widehat{p}(q_j \,|\, \vec{w}_i) \ .$$

This trick has allowed the method to achieve impressive results in some applications such as web search where separate models for title words, link text, etc. were combined by (Craswell & Hawking, 2003). It is not clear at this stage, however, whether this trick represents some some fundamental theoretical property or correction term of language modelling for information retrieval.

When a high perplexity discrete PCA model is applied without this smoothing, performance is not always good, but if the query is rather general, it can be surprisingly good. Some examples are presented by (Buntine et al., 2004; Buntine & Jakulin, 2004). Intuitively, for general queries where $p(\vec{q} \,|\, \mathcal{D}_i, \mathcal{M})$ has significant statistical support from the model $p(\mathcal{D}_i \,|\, \mathcal{M})$, better performance in information retrieval might be expected. Thus one approach to using discrete PCA in information retrieval is to use query probabilities as a way of scoring broad topical relevance of a document, and thus combining it with other retrieval scores. That is, apply discrete PCA in situations where we expect the high perplexity model to translate to a high fidelity query probability $p(\vec{q} \,|\, \mathcal{D}_i, \mathcal{M})$, where the query is instead words for a general topical area.

5. Information Retrieval with Topic Specific Ranking

Taking the previously discussed experience and views into consideration, we developed a search engine that uses standard Tf.Idf as its retrieval engine, and then does post-filtering (i.e., re-ordering) or retrieved documents using topic specific page rank. We use the Okapi BM25 version of Tf.Idf described in (Zhai, 2001), recoded within our indexing system. The top 500 documents with no less that 25% of the Tf.Idf score of the best document are retained from a query q and put through the reranking phase.

For the query q, we also have topic words t that may be the same as q (if obtained from our initial search screen) or may be different (if obtained from subsequent search screens). For the query words t, the normalized component proportions (see section on discrete PCA) are estimated using Gibbs importance sampling with 2000 cycles (Buntine & Jakulin, 2004), to yield the 100-dimensional normalised vector $\vec{m}_t^*$. A topic-specific ranking probability is then obtained for

each page i by making then linear product of $\vec{m}_t^*$ with the $K = 100$ topic specific page ranks for the page represented as a 100-dimensional vector $\vec{r}_i$. This is then combined with the Tf.Idf score to produce a final ranking for the i-th document:

$$C * Tf.Idf(q, i) + \log\left(\sum_k r_{i,k} m_{t,k}^*\right) \qquad (1)$$

This heuristic formula is justified as follows:

- while Tf.Idf is not properly calibrated to any probability, we guess it is best viewed as a log probability, but of unknown scale[5],
- the constant C with we currently set to 0.05 is then intended to convert it to units of log probability,
- the sum inside the log is our approximation to what the topic specific page rank for topic words t would be for each page.

This formula is only evaluated on at most 500 documents, so is relatively cheap to do. Our system operates in real-time.

This formula has two significant advantages when the topic words t and the query words q are identical.

- If the top results are topically coherent, then it is no different to standard tf.Idf,
- If the top results vary dramatically in topic, then a difference in response is seen. Normally a broader topical range is returned, with a focus on the most central topic.

The performance of this technique can be evaluated by using the search engine demonstrated at our test website. The commentary pages at the site also give details of the results of the topic-specific link analysis performed here. To view results with Tf.Idf alone, after the first query is done, blank the content of the "context words" box and resubmit a query.

6. Examples of Queries

We briefly present here a number of examples. For the query "jazz musician playing clarinet," topical filtering yields (taking context words from the query)

> Ted Lewis (musician), Pee Wee Russell, Benny Goodman, Dixieland, Han Bennink, Louis Armstrong and his Hot Five, Leon Roppolo

and Tf.Idf yields

> Dixieland, Music of the United States before 1900, Benny Goodman, Music of Brittany, Pee Wee Russell, Klezmer, List of jazz musicians.

The latter has more irrelevant entries. This next example illustrates biasing the search with different context words. For the query "madonna" with context words "paintings and statues", topical filtering yields

> The Madonna of Port Lligat, Black Madonna, Madonna and Child (Duccio), Pier Antonio Mezzastris, Madonna (art), The Madonna, Madonna Inn

and Tf.Idf with the query 'madonna paintings and statues" yields

> Leonardo da Vinci, List of artwork, Michelangelo Buonarroti, Quito, Vizzini, Icon, List of statues on Charles Bridge

One sees a better emphasis in topical filtering on Madonna, whereas in Tf.Idf the topic words swamp the query. This ability to topically bias the queries works well. The suggested topics are also applicable over 85% of the time, and thus usually very useful. For instance, for the query "stars", the suggested topics are "Space Opera", "Astronomy". "Movies" and "Music Albums". The suggested topics for "Madonna" are shownon Figure 1.

We evaluated the system using the following set of queries (queries are semi-colon delimited):

> system; power; reputation; tiger; nomenclature; caravan; spring; rendition; political history; drug addiction; forensic science; railway; evolution; probability computing; minimum description length.

Each query was run through Tf.Idf, topical filtering, and Tf.Idf with standard pagerank (computed on the same link structure as topical filtering). The third method we denote here as ranked Tf.Idf. the top 10 results of each query where then blindly evaluated on the three methods and these evaluations collated. The

[5]Clearly questionable since it can also be viewed as a utility.

relative scores, averaged between 1-5 are Tf.Idf: 3.5, topical filtering: 4.2, ranked Tf.Idf: 3.0.

The new method was consistently good, but not always better. These queries have some ambiguity, and Tf.Idf alone does poorly in some of these cases, as does ranked Tf.Idf. Topic-specific page rank tends to make the ranking score more relevant to the query, whereas in general page rank, the ranking score is oblivious to the query.

7. Conclusion

The novel combination of topic specific ranking and semantic component analysis presented here has a number of advantages.

Topic specific scoring provided by the adapted random surfer model, as shown by the Wikipedia examples, provides a far more characteristic score for documents than the proportion of component. The titles of high-ranking documents are indicative of the component, and in many cases can serve as reasonable component titles or descriptions. In contrast, documents containing a large proportion of the component are best described as "typical". They are neither indicative or characteristic. Topic-specific link analysis is therefore a valuable tool for the interpretation of topics developed by discrete PCA.

The ranking works well as a *topically biased post-ranking filter* for standard information retrieval. Experience on the Wikipedia search engine so-developed shows the resultant retrieval to be effective in many cases, though it has a small negative effect in a few cases. In more than half the cases, where there is no topical ambiguity, it appears no different to regular Tf.Idf. In some typically ambiguous queries, it shows a dramatic improvement.

Perhaps the best potential for the topically biased post-ranking filter, however, is that it provides an effective means for users to bias their search in topical directions using our novel "topic by example" interface. This ability is suggested by web commentary on search engines, and serves as a simple and immediately available counterpart to full semantic web capability, which itself is not currently available. While "topic by example" has no counterpart in existing information retrieval, it is also something that needs to gain acceptance from the fickle users of search engines.

Acknowledgments.

The work was supported by the ALVIS project, funded by the IST Priority of the EU's 6th framework programme, and the Search-Ina-Box project, funded by the Finnish TEKES programme. It benefits greatly from discussions with Natalie Jhaveri and Tomi Heimonen and of the Tampere Unit for Computer-Human Interaction at University of Tampere.

References

Azzopardi, L., Girolami, M., & van Risjbergen, K. (2003). Investigating the relationship between language model perplexity and IR precision-recall measures. *SIGIR '03* (pp. 369–370). Toronto, Canada.

Blei, D., Ng, A., & Jordan, M. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Buntine, W. (2005). Discrete principal component analysis. submitted.

Buntine, W., & Jakulin, A. (2004). Applying discrete PCA in data analysis. *UAI-2004*. Banff, Canada.

Buntine, W., Perttu, S., & Tuulos, V. (2004). Using discrete PCA on web pages. *Workshop on Statistical Approaches to Web Mining, SAWM'04*. At ECML 2004.

Canny, J. (2004). GaP: a factor model for discrete data. *SIGIR 2004* (pp. 122–129).

Craswell, N., & Hawking, D. (2003). Overview of the TREC 2003 web track. *Proc. TREC 2003*.

Haveliwala, T. (2002). Topic-specific pagerank. *11th World Wide Web*.

Hofmann, T. (1999). Probabilistic latent semantic indexing. *Research and Development in Information Retrieval* (pp. 50–57).

Johnson, S. (2003). Digging for googleholes. *Slate*. `http://slate.msn.com/id/2085668/index.html`.

Nallapati, R. (2004). Disciminative models for information retrieval. *ACM SIGIR Conference*.

Ponte, J., & Croft, W. (1998). A language modeling approach to information retrieval. *Research and Development in Information Retrieval* (pp. 275–281).

Pritchard, J., Stephens, M., & Donnelly, P. (2000). Inference of population structure using multilocus genotype data. *Genetics*, *155*, 945–959.

Richardson, M., & Domingos, P. (2002). The intelligent surfer: Probabilistic combination of link and content information in pagerank. *NIPS*14*.

Suomela, S., & Kekäläinen, J. (2005). Ontology as a search-tool: A study of real users' query formulation with and without conceptual support. *ECIR 2005* (pp. 315–329).

Zhai, C. (2001). *Notes on the Lemur TFIDF model* (note with Lemur 1.9 documentation). School of CS, CMU.

Evaluating the Robustness of Learning from Implicit Feedback

Filip Radlinski FILIP@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Thorsten Joachims TJ@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

This paper evaluates the robustness of learning from implicit feedback in web search. In particular, we create a model of user behavior by drawing upon user studies in laboratory and real-world settings. The model is used to understand the effect of user behavior on the performance of a learning algorithm for ranked retrieval. We explore a wide range of possible user behaviors and find that learning from implicit feedback can be surprisingly robust. This complements previous results that demonstrated our algorithm's effectiveness in a real-world search engine application.

1. Introduction

The task of learning ranked retrieval functions has recently received significant interest in the machine learning community (Bartell & Cottrell, 1995; Freund et al., 1998; Joachims, 2002; Kemp & Ramamohanarao, 2003). This is largely motivated by a goal of learning improved retrieval functions for web search.

The two standard approaches for collecting training data in this setting use explicit and implicit feedback. Explicit feedback involves actively soliciting relevance feedback by recording user queries and then explicitly judging the relevance of the results (Crammer & Singer, 2001; Herbrich et al., 2000; Rajaram et al., 2003). Acquiring explicit relevance judgments is time consuming and tedious, making large amounts of such data impractical to obtain. The alternative is to extract implicit relevance feedback from search engine log files (Kelly & Teevan, 2003; Cohen et al., 1999; Joachims, 2002; Kemp & Ramamohanarao, 2003). This allows virtually unlimited data to be collected at very low cost, but this data tends to be noisy and biased (Joachims et al., 2005; Radlinski & Joachims, 2005). In this paper, we consider a method for learning from implicit feedback and use modeling to understand when it is effective.

In contrast to typical learning problems where we have a fixed dataset, the task of learning to rank from implicit feedback is an interactive process between the user and learning algorithm. The training data is collected by observing user behavior given a particular ranking. If an algorithm presents users with a different ranking, different training data will be collected.

This type of interactive learning requires that we either run systems with real users, or build simulations to evaluate algorithm performance. The first involves building a search system to collect training data and evaluate real user behavior. While providing the most compelling results, this approach has a number of drawbacks. First, evaluating with real users is slow and requires a significant number of different users. Moreover, if a particular learning method proves ineffective, users quickly switch to other search engines. Finally, when we only collect the behavior of real users, the behavior is determined by the user base. Such results do not allow us to study the robustness of learning algorithms and feedback mechanisms. It is this issue that is our primary concern in this paper.

The alternative, often used in reinforcement learning, is to build a simulation environment. Obviously this has the drawback that it is merely a simulation, but it also has significant advantages. It allows more rapid testing of algorithms than by relying on user participation. It also allows exploration of the parameters of user behavior. In particular, we can use a model to explore the robustness of a learning algorithm to noise in the training data. We cannot have such control when real users are involved, and unlike the usual learning problem setting we are unaware of any way to inject realistic implicit feedback noise into real-world

Appearing in *W4: Learning in Web Search, at the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

training data and evaluate its effect.

In this paper, we present a user model to analyze the robustness of the Osmot search engine (Radlinski & Joachims, 2005). Osmot learns ranked retrieval functions by observing how users reformulate queries and how they click on results. We first present the learning algorithm, then the user model where we draw on the results of an eye-tracking study (Granka et al., 2004). We next demonstrate our algorithm's tolerance to noise in user behavior, having previously shown it to be effective in a real-world search engine (Radlinski & Joachims, 2005). We find Osmot to tolerate a strong user preference to click on higher ranked documents, and that it is able to learn despite most users only looking at the top few results. Our approach is generally interesting because it provides a practical method to evaluate the robustness of learning from implicit feedback. We plan to publicly release Osmot, including our model implementation.

2. Learning to Rank

Before we present our simulation model, we describe how Osmot learns from implicit feedback. For this, we assume a standard web search setting.

Our method relies on implicit feedback collected from log files. We record the queries users run as well as the documents they click on in the results. In these log files, we assume that documents clicked on are likely more relevant than documents seen earlier by the user, but not clicked on. This allows us to extract implicit relevance judgments according to a given set of feedback strategies. Within each search session, we assume each user runs a sequence, or chain, of queries while looking for information on some topic. We segment the log data into query chains using a simple heuristic (Radlinski & Joachims, 2005).

2.1. Implicit Feedback Strategies

We generate preference feedback using the six strategies illustrated in Figure 1. They are validated and discussed more in (Radlinski & Joachims, 2005). The first two strategies show single query preferences. "Click $>_q$ Skip Above" proposes that given a clicked-on document, any higher ranked document that was not clicked on is less relevant. The preference is indicated by an arrow labeled with the query, to show that the preference is with respect to that query. Note that these preferences are not stating that the clicked-on document *is* relevant, rather that it is *more likely* to be relevant than the ones not clicked on. The second strategy, "Click 1^{st} $>_q$ No-Click 2^{nd}" assumes that users typically view both of the top two results before clicking, as suggested by an eye-tracking study described below (Joachims et al., 2005). It states that if the first document is clicked on, but the second is not, the first is likely more relevant than the second.

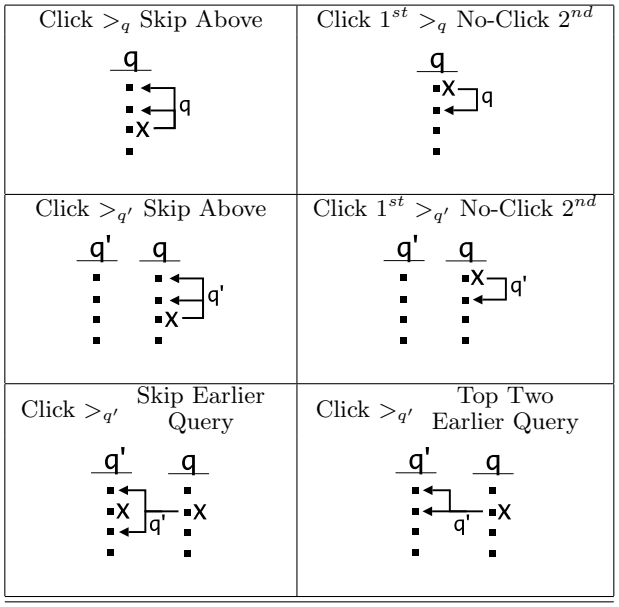

Figure 1. Feedback strategies. The user has run query q' followed by q. Each dot represents a result and an x indicates it was clicked on. We generate a constraint for each arrow shown, with respect to the query marked.

The next two strategies are identical to the first two except that they generate feedback with respect to the *earlier* query. The intuition is that since the two queries belong to the same query chain, the user is looking for the same information with both. Had the user been presented with the later results for the earlier query, she would have preferred the clicked-on document over those skipped over.

The last two strategies make the most use of query chains. They state that a clicked-on result is preferred over any result not clicked on in an earlier query (within the same query chain). This judgment is made with respect to the earlier query. We assume the user looked at all the documents in the earlier query up to one past the last one clicked on. In the event that no documents were clicked on in the earlier query, we assume the user looked at the top two results.

Ultimately, given some query chain, we make use of all six strategies as illustrated in the example in Figure 2.

2.2. Learning ranking functions

We define the relevance of d_i to q as a linear function,

$$rel(d_i, q) := w \cdot \Phi(d_i, q) \qquad (1)$$

q1	q2
d1	d4 x
d2 x	d5
d3	d6

$d_2 >_{q1} d_1$	$d_4 >_{q2} d_5$	$d_4 >_{q1} d_5$
$d_4 >_{q1} d_1$	$d_4 >_{q1} d_3$	

Figure 2. Sample query chain and the feedback that would be generated. Two queries were run, each returning three results of which one was clicked on. $d_i >_q d_j$ means that d_i is preferred over d_j with respect to the query q.

where $\Phi(d_i, q)$ maps documents and queries to a feature vector. Intuitively, Φ can be though of as describing the quality of the match between d_i and the query q. w is a weight vector that assigns weights to each of the features in Φ, giving a real valued retrieval function where a higher score indicates d_i is estimated to be more relevant to q. The task of learning a ranking function becomes one of learning w.

The definition of $\Phi(d_i, q)$ is key in determining the class of ranking functions we can learn. We define two types of features: rank features, $\phi_{rank}(d_i, q)$, and term/document features, $\phi_{terms}(d_i, q)$. Rank features serve to exploit an existing static retrieval function rel_0, while term/document features allow us to learn fine-grained relationships between particular query terms and specific documents. Note that rel_0 is the only ranking function we have before any learning has occurred and is thus used to generate the original ranking of documents. In our case, we use a simple TFIDF weighted cosine similarity metric as rel_0.

Let $W := \{t_1, \ \ldots, \ t_N\}$ be all the terms in our dictionary. A query q is a set of terms $q := \{t'_1, \ \ldots, \ t'_n\}$ where $t'_i \in W$. Let $D := \{d_1, \ \ldots, \ d_M\}$ be the set of all documents. We also define $r_0(q)$ as the ordered set of results as ranked by rel_0 given query q. Now,

$$\Phi(d, q) = \begin{bmatrix} \phi_{rank}(d, \ q) \\ \phi_{terms}(d, \ q) \end{bmatrix}$$

$$\phi_{rank}(d, \ q) = \begin{bmatrix} \mathbf{1}(Rank(d \ in \ r_0(q)) \leq 1) \\ \vdots \\ \mathbf{1}(Rank(d \ in \ r_0(q)) \leq 100) \end{bmatrix}$$

$$\phi_{terms}(d, \ q) = \begin{bmatrix} \mathbf{1}(d = d_1 \wedge t_1 \in q) \\ \vdots \\ \mathbf{1}(d = d_M \wedge t_N \in q) \end{bmatrix}$$

where $\mathbf{1}$ is the indicator function.

Before looking at the term features $\phi_{terms}(d, q)$, consider the rank features $\phi_{rank}(d, q)$. We have 28 rank features (for ranks 1,2,..,10,15,20,..,100), with each set to 1 if document d in $r_0(q)$ is at or above the specified rank. The rank features allow us to make use of the original ranking function.

The term features, $\phi_{terms}(d, q)$, are each of the form $\phi_{term}^{t_i,d_j}(d, q)$, set to either 0 or 1. There is one for every (term, document) pair in $W \times D$. These features allow the ranking function to learn associations between specific query words and documents. This is usually a very large number of features, although most never appear in the training data. Furthermore, the feature vector $\phi_{terms}(d, q)$ is very sparse. For any particular document d, given a query with $|q|$ terms, only $|q|$ of the $\phi_{term}^{t_i,d_j}(d, q)$ features are set to 1.

We use a modified ranking SVM (Joachims, 2002) to learn w from Equation 1. Let d_i be more relevant than d_j to query q: $rel(d_i, q) > rel(d_j, q)$. We can rewrite this, adding margin and non-negative slack variables:

$$w \cdot \Phi(d_i, \ q) \geq w \cdot \Phi(d_j, \ q) + 1 - \xi_{ij} \qquad (2)$$

We also have additional prior knowledge that absent any other information, documents with a higher rank in $r_0(q)$ should be ranked higher in the learned ranking system. There are both intuitive and practical reasons for these constraints (Radlinski & Joachims, 2005).

This gives the following optimization problem that we solve using SVM^{light} (Joachims, 1999) with $C = 0.1$:

$$\begin{array}{l} \min_{w,\xi_{ij}} \frac{1}{2} w \cdot w + C \sum_{ij} \xi_{ij} \ \text{ subject to} \\ \quad \forall (q, i, j) : \ w \cdot (\Phi(d_i, q) - \Phi(d_j, q)) \geq 1 - \xi_{ij} \\ \quad \forall i \in [1, \ 28] : \ \ w^i \geq 0.01 \\ \quad \forall i, j : \ \ \xi_{ij} \geq 0 \end{array} \qquad (3)$$

We have shown that this algorithm works in a real-world setting in the Cornell University library web search engine (Radlinski & Joachims, 2005). Due to space constraints we do not repeat those results here.

3. Model Description

We now present a model of user behavior when searching. This model will allow us to measure the robustness of Osmot to changes in user behavior. One part generates documents, and another simulates users searching the collection. After presenting the model, we support it by drawing on user behavior studies. Although it is clearly a simplification of reality, we show that this model is nonetheless useful.

3.1. Document Generation

Documents are generated as described in Table 1. The set of words is W, with word frequencies obeying a Zipf law. We define a set of topics T by uniformly picking N words from W for each topic. Some topics thus include

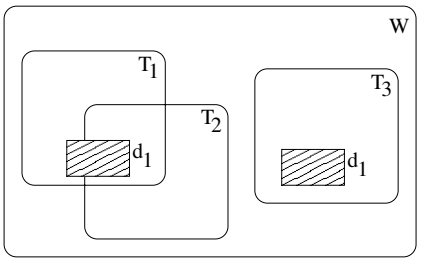

Figure 3. Document generation illustration. T_1, T_2 and T_3 are topics. Document d_1 is picked as relevant to two topics ($k_d = 2$), T_1 and T_3, although in selecting words from T_1, we also happened to select some words in T_2.

Table 1. Document Generation Model.

1. Let W be the set of all words. Let T be the set of topics, with each topic described by $T_i \subset W$.
2. Let each document d be generated as follows:
   - 2.1. $\forall T_i \in T: \ rel(d, T_i) = 0$
   - 2.2. Pick k_d binomially from $[0, \ MAX_T]$.
   - 2.3. If $k_d = 0$ Then
     Pick L words from W.
   - 2.4. Otherwise, do the following k_d times
     - a. Pick t from $[1, |T|]$.
     - b. Pick L/k_d words from T_t.
     - c. $rel(d, T_t) = rel(d, T_t) + 1/k_d$.

more common words than others (for example consider two topics, basketball and machine learning). This construct is illustrated in Figure 3. In our experiments, each word is on average in two topics.

Next, we generate each document d with L words one at a time. We pick k_d, which specifies how many different topics d is relevant to, as described in Table 1. Topics are picked according to a Zipf law to account for some topics being much more popular than others (again consider basketball versus machine learning). We set the relevance of the document to each topic to be proportional to the number of times the topic was picked with the sum of the relevances normalized to 1.

3.2. User Model

The process each user goes through as they search the web is specified in Table 2. This is a simple model, but as we will show it is reasonable and useful. Assume the user has a question q and wants to find the most relevant documents to the related topic $T_q \in T$. Users differ in their patience p and relevance threshold r. The patience determines how many results the user is likely to look at, while the relevance threshold specifies how relevant a document must appear to be (according to the abstract shown by the search engine) before the user clicks on it.

Table 2. User Behavior Model

1. Let q be the user's question, and p and r the user's patience and relevance thresholds respectively. They are sampled uniformly from (0,5] and [0.375,0.875] respectively.
2. While question q is unanswered
   - 2.1. Generate a query for question q. Let $d_1 \ldots d_n$ be the results for this query.
   - 2.2. Let $i = 1$, $p_q = p$.
   - 2.3. While $p_q > 0$
     - a. If $obsRel(d_i, q) > r$ Then
       If $obsRel(d_{i+1}, q) > obsRel(d_i, q) + c$
       Go to step (c)
       Otherwise
       Click on d_i.
       $p_q = p_q - 0.5 - (1 - rel(d_i, q))$.
       If $rel(d_i, q) = 1$ the user is done.
     - b. Otherwise
       $p_q = p_q - (r - obsRel(d_i, q))$
     - c. $i = i + 1$.
   - 2.4. With 50% probability, the user gives up.

Given a question, the user generates a query. We implement this by sampling words from the question topic with a Zipf law. This query returns a set of results and the user considers each in order. When the user observes a result, she estimates it's relevance to her question given a short abstract, observing $obsRel(d_i, q)$. The real relevance of d_i to query q is $rel(d_i, q)$. $obsRel(d_i, q)$ is drawn from an incomplete Beta distribution with α dependent on the level

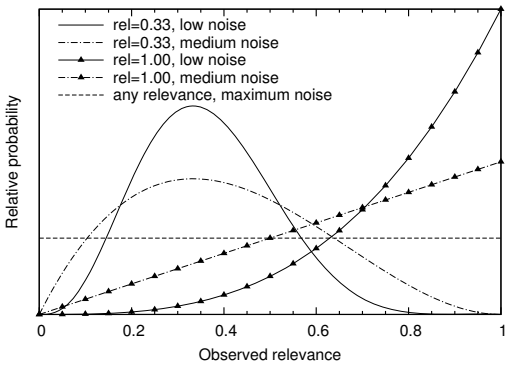

Figure 4. Probability of observing different perceived relevance as a function of the actual relevance.

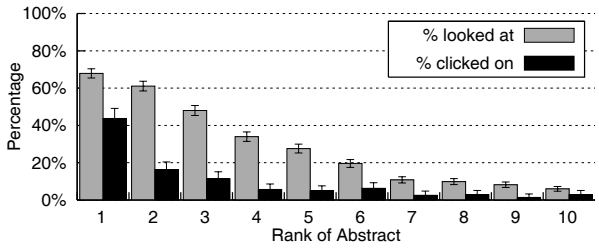

Figure 5. Percentage of time an abstract was viewed and clicked on depending on the rank of the result.

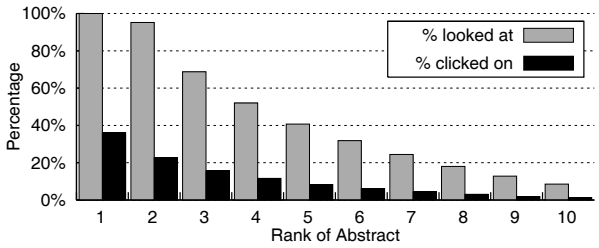

Figure 6. Percentage of time an abstract was viewed and clicked on in model depending on the rank of the result.

of noise and β selected so that the mode is at $rel(d_i, q)$ (unless $rel(d_i, q) = 0$, when the mode is at 0.05) as shown in Figure 4. This ensures the observed relevance is in the range [0,1] and has a level of noise that can be controlled.

If $obsRel(d_i, q) > r$, the user's relevance threshold, the user intends to click on d_i. However, the eye tracking study described below showed that users typically look at the next document below any they click on. Hence before clicking, the user looks at the next document, and moves on to it if it appears substantially more relevant. Otherwise, if $obsRel(d_i, q) \leq r$, the user moves on and her patience is reduced. The patience is reduced more for documents that appear less relevant because if she sees a document that appears to be completely irrelevant, she is more discouraged than if she sees a document that appears somewhat relevant.

When the user clicks on a document, she sees $rel(d_i, q)$. If she finds a document with maximum relevance, she stops searching. Otherwise, she returns to the search results and continues looking until her patience runs out, and then runs a new query with 50% probability.

3.3. Model Justification

We base our usage model on results obtained in an eye tracking study (Granka, 2004; Granka et al., 2004; Joachims et al., 2005). The study aimed to observe how users formulate queries, assess the results returned by the search engine and select the links they click on. Thirty six student volunteers were asked to search for the answers to ten queries. The subjects were asked to start from the Google search page and find the answers. There were no restrictions on what queries they may choose, how and when to reformulate queries, or which links to follow. All clicks and the results returned by Google were recorded by an HTTP proxy. Movement of the eyes was recorded using a commercial eye tracker. Details of the study are provided in (Granka et al., 2004).

Figure 5 shows the fraction of time users looked at, and clicked on, each of the top 10 search results after running a query. It tells us that users usually look at the top two result abstracts, and are much more likely to click on the first result than any other. Additionally, (Joachims et al., 2005) show that users usually look sequentially at the results from the top to the one below the last one clicked on.

Table 3. Behavioral dimensions explored

Short Name	Description
noise	Accuracy of relevance estimates.
ambiguity	Topic and word ambiguity.
trust	User's trust in presented ranking.
threshold	User selectivity over results.
patience	Number of results looked at.
reformulation	How often users reformulate.
improvement	Query improvement over time.

We observe in Figure 6 that the looks and clicks generated by this model resemble those seen in the user study. The most significant difference is in where users looked. Some of the time in the eye tracking study, the results show that users did not look at any results. We believe that this is partly due to errors in the eye tracker, and partly due to queries that did not return any results (such as spelling errors). For simplicity, we ignore these cases here.

We also measured the fraction of users who click on each of the top ten results in the Cornell University library search engine. The results confirmed that the distribution of clicks seen in Figures 5 and 6 is typical.

4. Learning Experiments

In this section, we explore the effect of different aspects of user behavior on the performance of Osmot. There are a number dimensions along which we assume user behavior may vary. These are listed in Table 3. For each, we present the effect of a change on our learning results and draw conclusions. Where possible, we relate the modeled results to real-world results to verify that the modeled results are realistic.

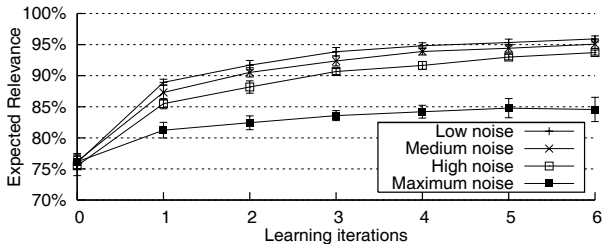

Figure 7. Ranking function performance for various noise levels.

4.1. High Level Parameters

We first consider the effect of two high level parameters: the level of difficulty users have in obtaining accurate relevance estimates from result abstracts, and the ambiguity in words appearing in documents and queries.

4.1.1. Accuracy of relevance estimates

After running a query, users select where to click by estimating the relevance of results from the abstracts presented. We now vary the noise in the user relevance estimate and examine the effect.

Figure 7 shows the mean relevance of the most relevant document in the top five results for various noise levels for the first query in each query chain. This relevance is known because the evaluation is on a synthetic dataset. Consider the first set of points, at iteration 0. We used rel_0 as a ranking function and modeled 4,000 users running queries and clicking on results. This gave about 75% mean highest top-5 relevance. Each curve shows the performance of the learning algorithm for different levels of noise in users' estimates of document relevance. For each noise level, using the data generated we learned a new ranking function. These results are shown at iteration 1. We see that in each case performance improves and this improvement is smaller with more noise.

Using the learned ranking function, we collect more training data. We then use the training data to learn a second ranking function, re-evaluate (the results are shown at iteration 2) and so forth. The noise levels correspond to setting α to 4, 2, 1.4 and 1 in the incomplete Beta distribution.

We see that most of the improvement occurs in the first two learning iterations, although it keeps accruing. We also see that the decay in improvement as more noise is introduced is gradual, which tells us that the Osmot algorithm can be decays gracefully with more noise.

Given that the preferences are generated over a known document collection, we can measure the error in the constraints generated according to the real document relevance. In this analysis, we ignore all preferences that indicate a preference over documents that have the same true relevance to a query. The fraction of constraints indicating that two documents should be ranked one way while the reverse is true for the four noise levels considered are 5%, 11%, 18% and 48%. These percentages show the mismatch between the generated preferences and the known ground truth on the 0th iteration. They measure how often a preference indicates that $d_i >_q d_j$ when $d_i <^*_q d_j$ in reality.

In order to measure the level of noise in real data, we collected explicit relevance judgments for the data recorded during the eye tracking study. Five judges were asked to (weakly) order all result documents encountered during each query chain according to their relevance to the question (Radlinski & Joachims, 2005). From this data, we found that the inter-judge disagreement in real preference constraints generated according to Figure 1 is about 14%. Note that this is a different measure than above because we are comparing the preferences of two judges rather than preferences of one judge to a ground truth. This means that the error rate between the ground truth and a human judge is in the range 7-14%, depending on the level of independence between the judgments of the two judges. These results tell us that the error rate in the medium noise setting is likely to be realistic.

The maximum noise case is special because in this case the users effectively ignore the document abstracts when deciding whether to click. Despite this, we still observe improved performance as we run the learning algorithm. How can this be explained? As mentioned above, the error rate in these constraints is 48%, meaning that 52% of the constraints correctly state a valid preference over documents. This comes about because users still start from the top result and stop searching after finding (clicking on) a completely relevant document, producing some bias. Also note that we generate the most preferences for the last (and often completely relevant) document clicked on within a query chain. While some of this effect may be an artifact of our setup, we still find it interesting that this learning approach appears to be effective with such a small signal to noise ratio.

4.1.2. Topic and word ambiguity

In the dataset used above, each word is on average in two topics. We also created collections where words were never in more than one topic, and where each word is on average in three topics. Figure 8 shows the results for the three collections. We see that with unambiguous words the ranking algorithm learns faster

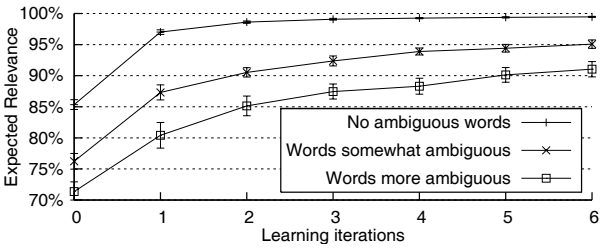

Figure 8. Ranking function performance for document collections with different levels of word ambiguity.

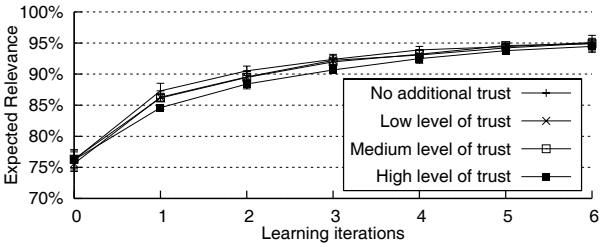

Figure 9. Ranking function performance versus the additional trust users place in the search engine.

and that even with more word ambiguity, our learning algorithm performs well.

4.2. Lower Level Parameters

The remainder of the behavioral dimensions are at a lower level, determining individual user behavior. We next explore the effect of these parameters.

4.2.1. User trust in ranking presented

We saw earlier that users click surprisingly often on the top link. In fact, users appear to have inherent trust in Google that is not correlated to the relevance of the result abstracts (Joachims et al., 2005). We tested if such trust affects Osmot. Figure 9 shows that additional trust (implemented by increasing *obsRel* proportionally to the inverse of the rank of each result) has no lasting effect. This is interesting because it demonstrates that even when click-through feedback is strongly biased, it still provides useful training data.

An alternative explanation for users clicking predominantly on the top few results is that some users are more selective than others. Many may click on the first partially relevant result, i.e. the top one while others may only click on results that appear highly relevant. To test this, we added a constant to the threshold value picked in the user model. We found that performance was very similar over a reasonable range of values.

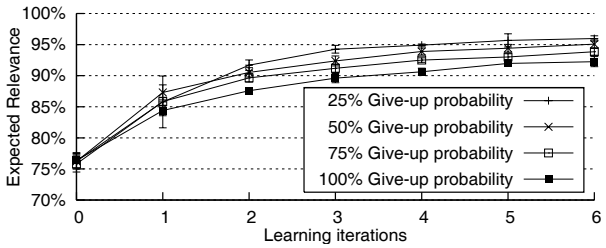

Figure 10. Ranking function performance for various probabilities that unsuccessful users will reformulate their query.

4.2.2. Number of results looked at

Figure 5 also showed us that users look at surprisingly few of the search results. In order to explore the effect of this on the effectiveness of our learning approach, we changed the range of patience levels that users have. In the four settings tested, about 3%, 7%, 15% and 23% of users looked past the top 5 abstracts. The results showed that this has no significant effect on the performance for the first few iterations of learning, although the improvement in expected relevance tapers out faster in the case where users view fewer results. We omit the full results due to space constraints.

4.2.3. How, and how often users reformulate

Previous work studying web search behavior (Lau & Horvitz, 1999; Silverstein et al., 1998) observed that users rarely run only one query and immediately find suitable results. Rather, users tend to perform a sequence of queries. Such query chains are also observed in the eye tracking study and our real-world search engine. Given Osmot's dependence on query chains, we wished to measure the effect of the probability of reformulation on the ranking function performance. The results are shown in Figure 10.

We see that the reformulation probability has a small but visible effect on ranking performance. While these results agree with our real-world experience that the presence of query chains makes a difference in algorithm performance (Radlinski & Joachims, 2005), we conjecture that in practice the difference is larger than seen here. In particular, unlike the model of user behavior presented in this paper, we suspect that later queries are not identically distributed to earlier queries. Rather we hypothesize that later queries are better and that this accounts for an additional improvement in performance when users chain multiple queries.

Using the relevance judgments of the five judges on the data gathered in the eye tracking study, we tested this hypothesis. Indeed, when a strict preference judgment

is made by a human judge comparing the top result of two queries in a query chain, 70% of the time the top result of the later query is judged more relevant. We see a similar result when comparing the second ranked documents. We attempted to add such an effect to our model by making later queries progressively longer, but this did not end up having any discernible effect. We intend to explore this question more in the future.

5. Conclusions and Future Work

In this paper we have presented a simple model for simulating user behavior in a web search setting. We used this model to study the robustness of an algorithm for learning to rank that we previously found to be effective in a real-world search engine. We demonstrated that the learning method is robust to noise in user behavior for a number of document collections with different levels of word ambiguity. Our results are important because they show that modeling allows fast explorations of the properties of algorithms for learning to rank. Although a more realistic model of user search behavior can be constructed, we have presented a reasonable starting model.

The model currently has a number of limitations that we intend to improve upon in the future. However, we believe that even in its present state it provides a valuable tool for understanding the performance of algorithms for learning to rank. We plan to make our implementation available to the research community.

6. Acknowledgments

We would like to thank Laura Granka, Bing Pang, Helene Hembrooke and Geri Gay for their collaboration in the eye tracking study. We also thank the subjects of the eye tracking study and the relevance judges. This work was funded under NSF CAREER Award IIS-0237381 and the KD-D grant.

References

Bartell, B., & Cottrell, G. W. (1995). Learning to retrieve information. *Proceedings of the Swedish Conference on Connectionism*.

Cohen, W. W., Shapire, R. E., & Singer, Y. (1999). Learning to order things. *Journal of Artificial Intelligence Research*, *10*, 243–270.

Crammer, K., & Singer, Y. (2001). Pranking with ranking. *Proceedings of the conference on Neural Information Processing Systems (NIPS)*.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (1998). An efficient boosting algorithm for combining preferences. *International Conference on Machine Learning (ICML)*.

Granka, L. (2004). Eye tracking analysis of user behaviors in online search. Master's thesis, Cornell University.

Granka, L., Joachims, T., & Gay, G. (2004). Eye-tracking analysis of user behavior in www search. *Poster Abstract, Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*.

Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers* (pp. 115–132).

Joachims, T. (1999). Making large-scale SVM learning practical. In B. Schlkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods – support vector machines*. MIT Press.

Joachims, T. (2002). Optimizing search engines using clickthrough data. *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Joachims, T., Granka, L., Pang, B., Hembrooke, H., & Gay, G. (2005). Accurately interpreting clickthrough data as implicit feedback. *Annual ACM Conference on Research and Development in Information Retrieval (SIGIR)*.

Kelly, D., & Teevan, J. (2003). Implicit feedback for inferring user preference: A bibliography. *SIGIR Forum*, *32*.

Kemp, C., & Ramamohanarao, K. (2003). Long-term learning for web search engines. *PKDD* (pp. 263–274).

Lau, T., & Horvitz, E. (1999). Patterns of search: Analyzing and modelling web query refinement. *Proceedings of the 7th International Conference on User Modeling*.

Radlinski, F., & Joachims, T. (2005). Query chains: Learning to rank from implicit feedback. *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Rajaram, S., Garg, A., Zhou, Z. S., & Huang, T. S. (2003). Classification approach towards ranking and sorting problems. *Lecture Notes in Artificial Intelligence* (pp. 301–312).

Silverstein, C., Henzinger, M., Marais, H., & Moricz, M. (1998). *Analysis of a very large AltaVista query log* (Technical Report 1998-014). Digital SRC.

Type-enabled Keyword Searches with Uncertain Schema (Invited Talk)

Soumen Chakrabarti

Department of Computer Science, Indian Institute of Technology, Bombay, India

Abstract: Web search is beginning to exploit powerful machine learning tools that annotate the corpus with entities and relationships. Such annotations, together with techniques for disambiguation and linkage resolution, will lead to graphical models that capture flexible type information, as well as represent the inherent uncertainty in the extracted structure. The next piece in the puzzle is a schema-agnostic query language that enables embedding type constraints in a user-friendly way; alternatively, machine learning techniques can extract type specs from unstructured queries. The final challenge is to devise a model for matching, scoring, and top-k search that naturally handles the uncertainty in the graph structure, and leads to manageable indices, scalable query execution algorithms, and user satisfaction.

Pipelets: A Framework for Distributed Computation

John Carnahan CARNAHAJ@YAHOO-INC.COM
Dennis DeCoste DECOSTED@YAHOO-INC.COM
Yahoo! Research Labs, 210 S. De Lacey Ave. Suite 105, CA 91105 USA

Abstract

The Pipelet framework provides a standards-based approach to the parallel processing of large data sets on clusters of machines. Pipelets are small interdependent computational units that can be assembled using a simple declarative language. In this framework both data and computational units can be dynamically distributed across a large group of machines in order to optimize the flow of data between components. The Pipelet framework presents a programming model for both simple and complex tasks in machine learning and information retrieval.

1. Introduction

Many problems in machine learning and information retrieval can be easily parallelized in order to distribute computation across a cluster of machines. For example pattern matching can be performed on a large set of text documents by dividing the documents into subsets, mapping a grep function to each subset on a separate machine and then combining the results from each machine (the reduction step) (1). More complex operations can performed by combining such map and reduce pairs serially to form a pipeline of serial operations. There are limits to this model for distributed computing. More complex problems require other constructs applied to the entire data set. For example performing web page classifications may include crawling a large set of documents, extracting text features from each document and identifying named entities. Such problems may require higher order language constructs to be performed on the entire data set. In order to exploit a cluster of machines such problems must be expressed as a single task made up of different components that can and cannot be parallelized. In this way parallelization and data flow can also be optimized for the entire problem.

The Pipelet framework provides a programming model to define a computational problem as a set of processing components in a pipeline. Our framework uses a standard declarative language for modeling the relationships between components. It also includes an implementation for executing such a pipeline on a single machine or a group of machines. In this way tasks can be optimized to increase the flow of data between components. Our framework includes an API for creating components within the pipeline that resembles popular related standards. The intent of this framework is to provide a standard means of describing both experiments and solutions in machine learning and information retrieval.

2. Pipeline Language

Pipelines in this framework are defined using the XML Pipeline Definition Language (XPDL) from W3C (2). The XPDL provides a simple vocabulary for describing processing relationships between a set of components. Using this language inputs and outputs for each component can be declared that can be linked to other components in the same pipeline. Pipeline components are defined within a single dependency graph with inputs of one component linked to one or more outputs from other components. Indirectly this language provides higher order language constructs such as loops and conditionals using recursive dependencies and arbitrary numbers of inputs. All pipeline processes or components are managed by a single pipeline controller. In this way the controller can manage how and when data is transferred between components as well as how the computational components are distributed in a cluster. Controllers can optimize the execution of all components in order to maximize the flow of data among components on a single or multiple machines.

3. Implementation

Our framework includes several pipeline controllers that comply with the XPDL specification version 1.0 (2002). Pipelines defined in our framework are intended to be used a variety of computing environments. We have created controllers that are optimized for executing pipelines on a single machine and others suited to cluster environments. Our implementation includes aggressive memoization and persistence for complex and large-scale tasks. We have also built software components that allow

Appearing in W4: Learning in Web Search, at the 22nd International Conference on Machine Learning, Bonn, Germany, 2005.

targets within individual pipelines to be invoked as individual web services.

4. Pipelet API

Our Pipelet framework includes a Java API for authoring individual components within a pipeline. This is a novel software API but is similar in many ways to Sun Microsystem's Servlet API (3). This API provides a stream interface for reading and writing to other components within the pipeline. The Pipelet API also includes a Service-Provider Interface (SPI) for extending the capabilities of pipelets and the creation of new controllers.

References

Dean, J & Ghemawat S (2004). *MapReduce: Simplified Data Processing on Large Clusters.* OSDI 2004

Walsh, N & Maler E (2002). *XML Pipeline Definition Language V 1.0.* http://www.w3.org/TR/2002/NOTE-xml-pipeline-20020228/

Java Servlet API v.2.4 (JSR-000154). Sun Microsystems http://java.sun.com/products/servlet/

Sailing the Web with *Captain Nemo*: a Personalized Metasearch Engine

Stefanos Souldatos STEF@DBLAB.ECE.NTUA.GR
Theodore Dalamagas DALAMAG@DBLAB.ECE.NTUA.GR
Timos Sellis TIMOS@DBLAB.ECE.NTUA.GR
School of Electrical and Computer Engineering, National Technical University of Athens, Athens, GR, 157 73

Abstract

Personalization on the Web is an issue that has gained a lot of interest lately. Web sites have already started providing services such as preferences for the interface, the layout and the functionality of the applications. Personalization services have also been introduced in Web search and metasearch engines, i.e. tools that retrieve Web pages relevant to keywords given by the users. However, those services deal mostly with the presentation style and ignore issues like the retrieval model, the ranking algorithm and topic preferences. In this paper, we present *Captain Nemo*, a fully-functionable metasearch engine that exploits personal user search spaces. Users can define their personal retrieval model and presentation style. They can also define topics of interest. *Captain Nemo* exploits several popular Web search engines to retrieve Web pages relevant to keywords given by the users. The resulting pages are presented according to the defined presentation style and retrieval model. For every page, *Captain Nemo* can recommend a relevant topic of interest to classify the page, exploiting nearest-neighbour classification techniques.

1. Introduction

Nowadays, huge volumes of data are available on the Web. Searching for information is extremely difficult, due to the large number of information sources and their diversity in organizing data. Users should not only identify these sources, but also determine those containing the most relevant information to satisfy their information need.

Search and metasearch engines are tools that help the user identify such relevant information. Search engines retrieve Web pages that contain information relevant to a specific subject described with a set of keywords given by the user. Metasearch engines work at a higher level. They retrieve Web pages relevant to a set of keywords, exploiting other already existing search engines.

Personalization on the Web is an issue that has gained a lot of interest lately. Web sites have already started providing services such as preferences for the interface, the layout and the functionality of the applications. Personalization services have also been introduced in Web search and metasearch engines. However, those services deal mostly with the presentation style and ignore issues like the retrieval model, the ranking algorithm and topic preferences.

In this paper, we present *Captain Nemo*, a fully-functionable metasearch engine that creates personal user search spaces. Users can define their personal retrieval model. For example, they can select the search engines to be used and their weight for the ranking of the retrieved pages, the number of pages retrieved by each engine, etc. Users can also define topics of interest. For every retrieved Web page, *Captain Nemo* can recommend a relevant topic of interest to classify the page, exploiting nearest-neighbour classification techniques. The presentation style is also customizable, as far as the grouping and the appearance of the retrieved pages is concerned.

A typical application scenario for *Captain Nemo* starts with a set of keywords given by the user. *Captain Nemo* exploits several popular Web search engines to retrieve Web pages relevant to those keywords. The resulting pages are presented according to the user-defined presentation style and retrieval model. We

Appearing in *W4: Learning in Web Search, at the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

note that users can maintain more than one different *profiles* which result to different presentation styles and retrieval models. For every retrieved Web page, *Captain Nemo* can recommend relevant topics of interest to classify the retrieved pages, exploiting nearest-neighbour classification techniques. User can optionally save the retrieved pages to certain folders that correspond to topics of interest for future use.

Contribution. The main contributions of our work are:

- We present personalization techniques for metasearch engines. These techniques do not only deal with the presentation style but also with the retrieval model and the ranking of the retrieved pages.
- We suggest semi-automatic classification techniques in order to recommend relevant topics of interest to classify the retrieved Web pages.
- We present a fully-functionable metasearch engine, called *Captain Nemo*[1], that implements the above framework.

Related Work. The need for Web information personalization has been discussed in (Shahabi & Chen, 2003; Sahami et al., 2004). Following this, several Web search and metasearch engines[2] offer personalization services. For example, Alltheweb offers the option to use personal stylesheets to customize the look and feel of its search page. Altavista provides styles to present the retrieved Web pages with high or low detail. The metasearch engines WebCrawler, MetaCrawler, Dogpile can group the Web pages according to the search engine that actually retrieves them. Regarding the retrieval model, several metasearch engines let the user define the search engines to be used (e.g. Query Server, Profusion, Infogrid, Mamma, Search, Ixquick). Some of them (e.g. Query Server, Profusion, Infogrid, Mamma) have a timeout option (i.e. time to wait for Web pages to be retrieved). Also, Query Server and Profusion offer the option of setting the number of Web pages retrieved by each engine. To the best of our knowledge, there is not any metasearch engine that offers the option of setting the weights of the search engines for the ranking of the retrieved pages.

Concerning the topics of interest, Buntine et al. (2004) claim that topic-based search will be necessary for the next generation of information retrieval tools. The search engine Northern Light[3] has an approach called *custom folders* that organizes search results into categories. Inquirus2 (Glover et al., 2001) uses a classifier to recognize web pages of a specific category and learn modifications to queries that bias results toward documents in that category. Chakrabarti et al. (1998) proposes statistical models for hypertext categorization by exploiting link information in a small neibourhood around documents.

Outline. The rest of this paper is organized as follows. The personalization features of *Captain Nemo* are discussed in Section 2. Section 3 presents the classification algorithm that recommends relevant topics of interest to classify retrieved Web pages. The architecture of *Captain Nemo* and several implementation issues are discussed in Section 4. Finally, Section 5 concludes this paper.

2. Maintenance of User Profiles

Captain Nemo maintains user profiles for different presentation styles and retrieval models. A user can have more than one different *profiles* which result to different presentation styles and retrieval models. Figure 1 illustrates the personal search space offered to users by *Captain Nemo*. We next discuss the available personalization options for the retrieval model, the presentation style and the topics of interest.

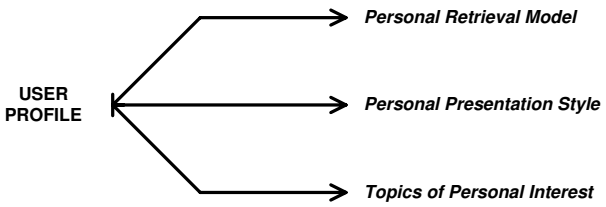

Figure 1. Personal search space offered by *Captain Nemo*.

2.1. Retrieval Model

As seen before, most of the existing metasearch engines employ a standard retrieval model. In *Captain Nemo*, this restriction is eliminated and users can create their own retrieval model, by setting certain parameters in the system. These parameters are described below:

Participating Search Engines. Users can declare the search engines they trust, so that only these en-

[1] http://www.dbnet.ece.ntua.gr/~stef/nemo/

[2] Google, Alltheweb, Yahoo, AltaVista, WebCrawler, MetaCrawler, Dogpile, etc.

[3] http://www.northernlight.com/index.html

gines are used by the metasearch engine.

Search Engine Weights. In a metasearch engine, retrieved Web pages may be ranked according to their ranking in every individual search engine that is exploited. In *Captain Nemo* (as shown in Section 4), the search engines can participate in the ranking algorithm with different weights. For example, a lower weight for a search engine indicates low reliability and importance for that particular engine. Users have the option to set their own weights for every search engine exploited by *Captain Nemo*.

Number of Results. A recent research (iProspect, 2004) has shown that the majority of search engine users (81.7%) rarely read beyond the third page of search results. Users can define the number of retrieved Web pages per search engine.

Search Engine Timeout. Delays in the retrieval task of a search engine can dramatically deteriorate the response time of any metasearch engine that exploits the particular search engine. In *Captain Nemo*, users can set a timeout option, i.e. time to wait for Web pages to be retrieved for each search engine. Results from delaying search engines are ingored.

2.2. Presentation Style.

Users of *Captain Nemo* can customize the look and feel for the presentation of the retrieved Web pages, having the following options:

Grouping. In a typical metasearch engine, the results returned by search engines are merged, ranked and presented as a single list. Beside this typical presentation style, *Captain Nemo* can group the retrieved Web pages (a) by search engine or (b) topic of interest pre-defined by the user. The latter is based on a semi-automatic classification technique which will be described in Sections 4. Figure 2 illustrates an example where retrieved Web pages are grouped by topic of interest.

Content. The results retrieved by *Captain Nemo* include the page title, page description and page URL. The user can declare which of these parts should be displayed.

Look and Feel. Users can customize the general look and feel of the applications. They can select among color themes and page layouts to define different ways of presenting results. Figure 3 shows the available options for customizing the look and feel of the application.

Figure 2. Grouping of retrieved Web pages by topic of interest.

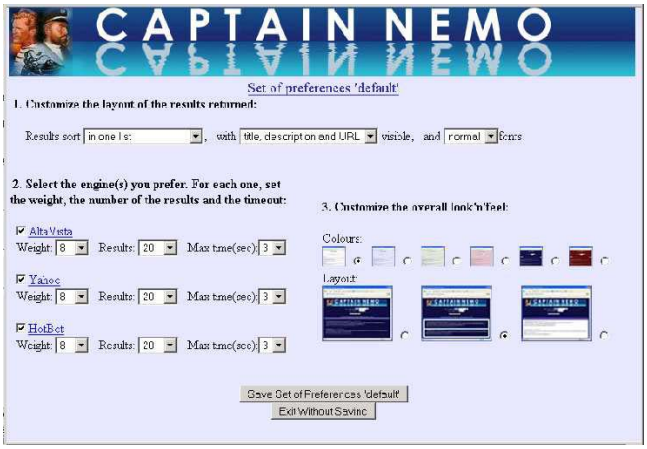

Figure 3. Editing set of preferences.

2.3. Topics of Interest

In *Captain Nemo*, the retrieved Web pages are presented according to the user-defined presentation style and retrieval model. For every retrieved Web page, *Captain Nemo* can recommend relevant topics of interest to classify the retrieved pages. Users can optionally save the retrieved pages to certain folders that correspond to topics of interest for future use.

Users can define and edit topics of interests (i.e. thematic categories). For each topic of interest, a set of keywords that describe its content should be provided. Topics and keyword descriptions can be altered anytime. The retrieved Web pages can be saved for future reference in folders that correspond to the defined topics of interest. Those folders have a role similar to *Favorites* or *Bookmarks* in Web browsers.

Figure 4 shows the administration options for manag-

Figure 4. Administrating topics of interest.

ing topics of interest.

3. Automatic Classification of Retrieved Web pages

Captain Nemo recommends relevant topics of interest to classify the retrieved pages, exploiting nearest-neighbour classification techniques. The description of a retrieved Web page includes its title and a part of its content (which is usually its first few lines). The description of a topic of interest includes a set of keywords given by the user. The classification algorithm identifies the most relevant topic of interest for all retrieved pages, considering the description of retrieved Web pages and pre-defined topics of interest.

Classification Algorithm. *Captain Nemo* exploits Nearest Neighbor (Witten et al., 1999) as its main classification algorithm. The algorithm needs to calculate similarity measures between the description of each retrieved Web page and the description of every topic of interest. The similarity measure employed is a $tf-idf$ one (Witten et al., 1999). Let D be the description of a topic of interest and R the description of a retrieved Web page. The similarity between the topic of interest and the retrieved Web page, $Sim(R, D)$, is defined as follows:

$$Sim(R,D) = \frac{\sum_{t \epsilon R \cap D} w_{R,t} \times w_{D,t}}{\sqrt{\sum_{t \epsilon R \cap D} w_{R,t}^2} \times \sqrt{\sum_{t \epsilon R \cap D} w_{D,t}^2}} \quad (1)$$

where t is a term, $w_{R,t}$ and $w_{D,t}$ are the weights of term t in R and D respectively. These weights are:

$$w_{R,t} = \log\left(1 + \frac{C}{C_t}\right) \quad (2)$$

$$w_{D,t} = 1 + \log f_{D,t} \quad (3)$$

where C is the total number of topics of interest, C_t is the number of topics of interest including term t in their description and $f_{D,t}$ is the frequency of occurence of t in description D.

Having a new, retrieved Web page, we rank the topics of interest according to their similarity with the page (the topic of interest with the highest similarity will be on the top). Then the top-ranked topic of interest is selected as the most appropriate for the retrieved page.

Example. Let us assume that a user has the following three topics of interest: (t1) Sports: sports football basketball baseball swimming tennis soccer game, (t2) Science: science scientific mathematics physics computer technology and (t3) Arts: arts art painting sculpture poetry music decorating.

The result "Alen Computer Co. can teach you the art of programming...Technology is just a game now...computer science for beginners" receives the following similarity scores for each topic of interest:

$$Sim(x, t_1) = 0.287$$

$$Sim(x, t_2) = 0.892$$

$$Sim(x, t_3) = 0.368$$

The highest score corresponds to $t2$. Consequently, the most relevant topic of interest is "Science".

4. System Implementation

This section presents the architecture of our application and discusses various interesting implementation issues. Figure 5 describes the main modules of *Captain Nemo*.

Search Module. It implements the main functionality of the metasearch engine, providing connections to the search engines specified by the users. It retrieves the relevant Web pages according to the retrieval model defined by the user. The results are sent to the ranking module for further processing.

Ranking Module. The retrieved Web pages are ranked and grouped according to the retrieval model defined by the user. The ranking algorithm is presented in the next section. For every retrieved Web page, a matching topic of interest is determined.

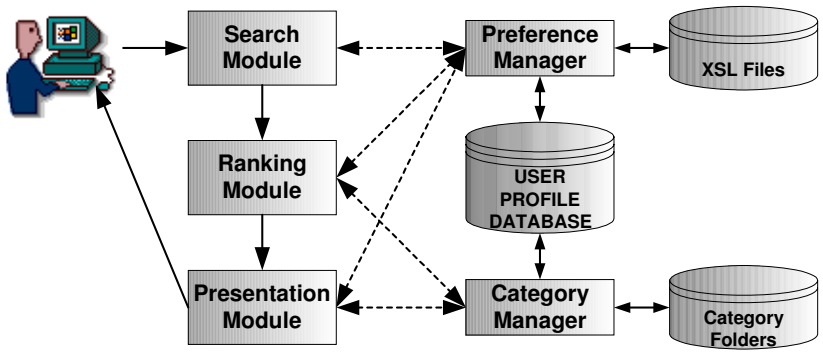

Figure 5. System architecture.

Presentation Module. It maintains several XSL filters that construct the resulting pages (wrapped as XML documents) according to the look and feel settings determined by the user.

Preference Manager. It provides the connection between the three aforementioned modules (i.e. search module, ranking module, presentation module) and the information stored in user profiles. It is also responsible for updating user profiles and the used XSL files.

Category Manager. It manages the topics of interests, keeps the appropriate folders on disk in accordance with the user profiles and provides all the necessary information for the automatic classification of results to those folders.

Our application is implemented on top of the PostgreSQL database system[4], exploiting Perl CGI scripts to wrap the results of search engines[5].

The next subsection discusses in detail the ranking mechanisms used in our application.

4.1. Ranking

Given a query, a typical metasearch engine sends it to several search engines, ranks the retrieved Web pages and merges them in a single list. After the merge, the most relevant retrieved pages should be on top. There are two approaches used to implement such a ranking task. The first one assumes that the initial scores assigned to the retrieved pages by each one of the search engines are known. The other one does not have any information about those scores.

In Rasolofo et al. (2001), it is pointed out that the scale used in the similarity measure in several search engines may be different. Therefore, normalization is required in order to achieve a common measure of comparison. Moreover, the reliability of each search engine must be incorporated in the ranking algorithm through a weight factor. This factor is calculated separately during each search. Search engines that return more Web pages should receive higher weight. This is due of the perception that the number of relevant Web pages retrieved is proportional to the total number of Web pages retrieved as relevant for all search engines exploited by the metasearch engine.

On the other hand, Dumais (1994), Gravano and Papakonstantinou (1998) and Towell et al. (1995) stress that the scores of various search engines are not compatible and comparable even when normalized. For example, Towell et al. (1995) notes that the same document receives different scores in various search engines and Dumais (1994) concludes that the score depends on the document collection used by a search engine. In addition, Gravano and Papakonstantinou (1998) points out that the comparison is not feasible not even among engines using the same ranking algorithm and claims that search engines should provide statistical elements together with the results.

In Aslam and Montague (2001), ranking algorithms are proposed which completely ignore the scores assigned by the search engines to the retrieved Web pages: bayes-fuse uses probabilistic theory to calculate the probability of a result to be relevant to the query, while borda-fuse is based on democratic voting. The latter considers that each search engine gives votes in the results it returns, giving N votes in the first result, $N-1$ in the second, etc. The metasearch engine gathers the votes for the retrieved Web pages from all search engines and the ranking is determined

[4] http://www.postgresql.org/

[5] http://search.cpan.org/dist/WWW-Search/lib/WWW/Search.pm

democratically by summing up the votes.

The algorithm adopted by *Captain Nemo* is the weighted alternative of Borda-fuse. In this algorithm, search engines are not treated equally, but their votes are considered with weights depending on the reliability of each search engine. These weights are set by the users in their profiles. Thus, the votes that the i result of the j search engine receives are:

$$V(r_{i,j}) = w_j * (max_k(r_k) - i + 1) \qquad (4)$$

where w_j is the weight of the j search engine and r_k is the number of results rendered by search engine k. Retrieved pages that appear in more than one search engines receive the sum of their votes.

4.2. Application Examples

The main page of *Captain Nemo* is illustrated in Figure 6. It includes the results of query 'perl', presenting only titles in compact format according to the user profile defined.

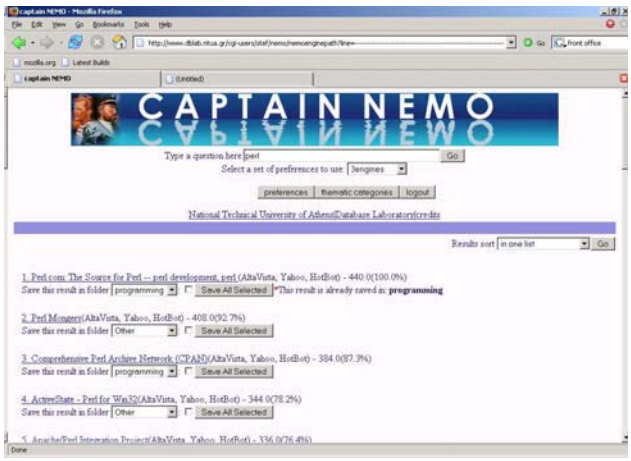

Figure 6. Captain Nemo.

Figure 7 shows the same results formatted by another presentation style. According to the preferences set, the results are merged in one list. For each retrieved Web page, we can see (a) the title, the description and the URL, (b) the names of search engines that have retrieved this particular page and (c) the absolute and relative similarity score calculated by the ranking module. A topic of interest is suggested for each retreived Web page.

Figure 8 shows the results for keywords 'java sql', grouped by topic of interest.

Figure 7. Retrieved Web pages for the keyword 'perl'.

Figure 8. Retrieved Web pages grouped by topic of interest.

5. Conclusion

In this paper we presented *Captain Nemo*, a fully-functionable metasearch engine that exploits personal user search spaces. Users can define their personal retrieval model and presentation style. They can also define topics of interest. *Captain Nemo* exploits several popular Web search engines to retrieve Web pages relevant to keywords given by the users. The resulting pages are presented according to the defined presentation style and retrieval model. For every page, *Captain Nemo* can recommend a relevant topic of interest to classify the page, exploiting nearest-neighbour classification techniques.

For future work, we plan to replace the flat model of topics of interest by a hierarchy of topics in the spirit of Kunz and Botsch (2002). Also, we will improve the classification process, exploiting background knowl-

edge in the form of ontologies (Bloehdorn & Hotho, 2004).

References

Aslam, J. A., & Montague, M. (2001). Models for metasearch. *Proceedings of the 24th ACM SIGIR Conference*.

Baker, L. D., & McCallum, A. K. (1998). Distributional clustering of words for text classification. *Proceedings of the 21st ACM SIGIR Conference* (pp. 96–103). Melbourne, Australia: ACM Press.

Bloehdorn, S., & Hotho, A. (2004). Text classification by boosting weak learners based on terms and concepts. *Proceedings of the 4th ICDM Conference* (pp. 331–334).

Buntine, W. L., Löfström, J., Perkiö, J., Perttu, S., Poroshin, V., Silander, T., Tirri, H., Tuominen, A. J., & Tuulos, V. H. (2004). A scalable topic-based open source search engine. *Proceedings of the ACM WI Conference* (pp. 228–234).

Chakrabarti, S., Dom, B. E., & Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. *Proceedings of the ACM SIGMOD Conference* (pp. 307–318). Seattle, US: ACM Press, New York, US.

Cohn, D., & Hofmann, T. (2001). The missing link - a probabilistic model of document content and hypertext connectivity. *Proceedings of the 15th NIPS Conference*.

Dumais, S. T. (1994). Latent semantic indexing (lsi) and trec-2. *Proceedings of the 2nd TREC Conference*.

Glover, E., Flake, G., Lawrence, S., Birmingham, W. P., Kruger, A., Giles, C. L., & Pennock, D. (2001). Improving category specific web search by learning query modifications. *Proceedings of the SAINT Symposium* (pp. 23–31). San Diego, CA: IEEE Computer Society, Los Alamitos, CA.

Gravano, L., & Papakonstantinou, Y. (1998). Mediating and metasearching on the internet. *IEEE Data Engineering Bulletin, 21*.

iProspect (2004). iProspect search engine user attitudes. http://www.iprospect.com/premiumPDFs/iProspectSurveyComplete.pdf.

Kunz, C., & Botsch, V. (2002). Visual representation and contextualization of search results-list and matrix browser. *Proceedings of the ICDC Conference* (pp. 229–234).

Liu, F., Yu, C., & Meng, W. (2002). Personalized web search by mapping user queries to categories. *Proceedings of the 11th CIKM Conference* (pp. 558–565). McLean, Virginia, USA: ACM Press.

Rasolofo, Y., Abbaci, F., & Savoy, J. (2001). Approaches to collection selection and results merging for distributed information retrieval. *Proceedings of the 10th ACM CIMK Conference*.

Sahami, M., Mittal, V. O., Baluja, S., & Rowley, H. A. (2004). The happy searcher: Challenges in web information retrieval. *Proceedings of the 8th PRICAI Conference* (pp. 3–12).

Shahabi, C., & Chen, Y.-S. (2003). Web information personalization: Challenges and approaches. *Proceedings of the 3rd DNIS Workshop*.

Towell, G., Voorhees, E. M., Gupta, N. K., & Johnson-Laird, B. (1995). Learning collection fusion strategies for information retrieval. *Proceedings of the 12th ICML Conference*.

Witten, I. H., Moffat, A., & Bell, T. C. (1999). *Managing gigabytes: Compressing and indexing documents and images*. Morgan Kaufmann Publishers. 2nd edition.