High Precision Information Retrieval with Natural Language Processing Techinques

Bronislav Frid
Liza Logounova
Alexander Michailov
Oleg Nusinzon
Leonid Zeltser

A Project Report
Submitted in Partial Fullfillment of the
Requirements for the Course CSE 401

May 1st, 1997

Advisors:
Mitch Marcus
Breck Baldwin
Jeff Reynar

Abstract

We developed a system that can be used to enhance typical information retrieval engines by improving relevancy of documents returned to the user. Our system attempts to recognize relevant documents with very high precision from very short (2-8 word) queries, such as those typically used to search the World Wide Web. We eliminated documents from consideration by using natural language processing techniques, which add to the discriminatory power of very short queries. Simple criteria such as exact string matches for the query in a document tend to be under discriminatory in many cases. We used natural language processing to increase precision of results by utilizing observations about discourse structure, noun phrases and proximity of matches of query terms.

Contents

1. INTRODUCTION

A typical information retrieval (IR) system responds to the user's query by selecting documents from a database and ranking them in terms of relevance. A successful IR system is able to filter out extraneous information and return only relevant documents. Such level of precision of very hard to accomplish. People rarely supply enough information for the system to determine what the user is looking for. For example, queries for a World Wide Web search engine almost never exceed 4 words. An IR system well suited for general use would be able to process very short queries.

If a query is somehow made to resemble a typical relevant document, "everything about this query becomes a valid search criterion: words, collocations, phrases, various relationships, etc." (Strzalkowski et al., 1997) Expanding the query in this way will, therefore, increase accuracy of the system's results. The desired expansion can be achieved by augmenting the original query with documents that are definitely relevant to the user. Such full text expansion may improve the system's performance by as much as 40%. (Strzalkowski et al., 1997)

We designed a system that started with a query 2 to 8 words long, and precisely selected several documents that were relevant to the original query. These documents can be used by an IR system to expand the original query, and choose more documents that accurately match the user's expectations.

We achieved the high precision level by building several independent recognizers and rankers. These modules selected a few documents that matched the original query with very high certainty. Our system then combined these results to produce a slightly larger set of highly relevant documents.

The selection process in IR systems has, traditionally, relied on exact string matching. The document's relevance was a function of the number of times each query word appeared in the document. Unfortunately, such systems were not precise enough in their selections, returning far too many irrelevant documents. We used various natural language processing (NLP) techniques to

improve accuracy of our results.

2. NATURAL LANGUAGE PROCESSING

Most of the natural language processing steps performed were done within the framework of the EAGLE system. (Baldwin et al., 1997) EAGLE allowed us to disambiguate words based on their part of speech, such as *can* used as verb or as a noun. We also eliminated semantically irrelevant words that only added clutter, such as prepositions, conjunctions and pronouns. While these function words are important for proper analysis of complete sentences we can safely ignore them for all models that use single-word or noun-phrase size tokens. We have also used EAGLE to normalize morphological forms of words, so that "*foot*" would match with "*feet*". In addition, we extracted noun phrases and determined sentence boundaries for more effective matching between query terms and words in the document.

We also used the Princeton WordNet package, to expanded query terms to their hyponyms and synonyms. The WordNet provides a network of related terms of English language, in particular it has a network of hyponyms for the majority of English nouns. Hyponym is a term that designates a sub-concept of another word. For instance word *novelist* is a hyponym of *writer*.

3. DATA SETS

In order to comply with short queries typically used to search the World Wide Web, we chose a real-world setting to test our system on. Since we were building a system to work on top of an already existing IR engine, we chose Alta Vista, owned by Digital Equipment Corporation, to provide us with our document set.

Alta Vista provided us with fast query reposonses and returned a data set of manageable size. It also had document relevance ranking that we could use as an initial benchmark for our system. Alta Vista was designed to work with the World Wide Web, and consequently with the kind of short queries we were developing our software for.

We also wanted to compare our results with an ideal relavance ranking. In order to do that we had to use some manually benchmarked data set. Our choice fell on Tipstser data collection. This collection also used short queries and already had manual benchmarks for relevance for each document. It was also a standard training set for many competitions in IR.

Tipster is a large corpus of newswire services, released by the Linguistic Data Consortium. This collection has been used by TREC-5 information retrieval competition, in course of which the queries and relevance judgements were produced.

For each query we used the union of all the documents retrieved by TREC participant engines, restricted to Associated Press 1988 subcorpus. This way we got an average of 180 documents per query, which is typical for results of a World Wide Web retrieval engine. We assumed TREC relevance judgements for both the development and evaluation data sets.

4. VECTOR SPACE

Vector Space is a mathematical model that is often used by IR systems. It attempts to determine how similar retrieved documents are to the user's query by constructing an N-dimensional token space, where N is the number of tokens in the query. In the "vanilla" version of Vector Space tokens constitute words from the query. Each document is then weighed according to how frequently it mentions tokens from the query.

Query was broken into tokens and an N-dimensional space was created. Each dimension is referenced by a token derived from the query and is equal to the number of occurrences of that token. In later versions of Vector Space, tokens were derived from morphological roots, part of speech tags, disambiguated meaning, and other NLP operations. These oeprations were carried out using Camp, as well as WordNet software packages. Various models were later created basing on the original, "vanila" Vector Space model. They employed one or more of the following techniques: disanmbiguation, morphological analysis, thesaurus expansion.

Disambiguation means that words like *can* are distinguished in the meaning of "*can* of soda" and "I *can* do it". It must be noted that we only do part of the jobs, as we still cannot determine whether, for instance, a noun *trade* means *profession* or *exchange*. Morphological analysis means that roots of the words are presented as tokens rather than words themselves. For example, *foot's* and *feet* are presented as token "*foot*". Thesaurus expansion means that words like "*writer*" are expanded to include "*novelist*", "*poet*", etc.

We adopted Vector Space as our initial relevance ranking model because it is straight-forward to implement and fast to compute. The basic version of Vector Space was easy to expand by adding various NLP modules for processing terms in the query.

Once the query is processed as described above, we have a collection of tokens that can be used to construct an N-dimensional array, where N is the number of tokens in the query. Algorithm then goes through each document in the collection and adds one to an appropriate location in the array if a matching token is found. After all the documents are processed, we are left with an array of N-dimensional arrays. This new array has a cardinality of the document set.

Lastly, to compute the Vector Space coefficient, we used the following formula: (Gerard Salton, 1983)

$$\textbf{relevance} = \frac{\sum^{\text{tokens}}(\textbf{query}_{\text{token}} * \textbf{document}_{\text{token}})}{\sqrt{(\sum^{\text{tokens}} \textbf{query}^2_{\text{token}})} * \sqrt{(\sum^{\text{tokens}} \textbf{document}^2_{\text{token}})}}$$

These numbers represent cosines of the angle between the vector constructed from the query and the vector of the document. As the angle approaches 0, cosines approahc 1. The closer cosines are together, the more similar documents are judged to be.

This method produces a total ordering on the document set based on each document's similarity ot the query. Various flavors of this model perform progeressively better as the complexity of NLP processing increases. However, increases are in a logarithmic relationship to the complexity, i.e. a small increase in precision, requires a very large increase in complexity and computing necessary.

As you can see in **Figure 1**, various Vector Space models outperformed Alta Vista's search engine rankings by a solid margin. In this figure, X-axis is the number of total documents in the document set and Y-axis is the number of relative documents. We compared all graphs to an ideal performance, where all documents in the document set were ranked manually.

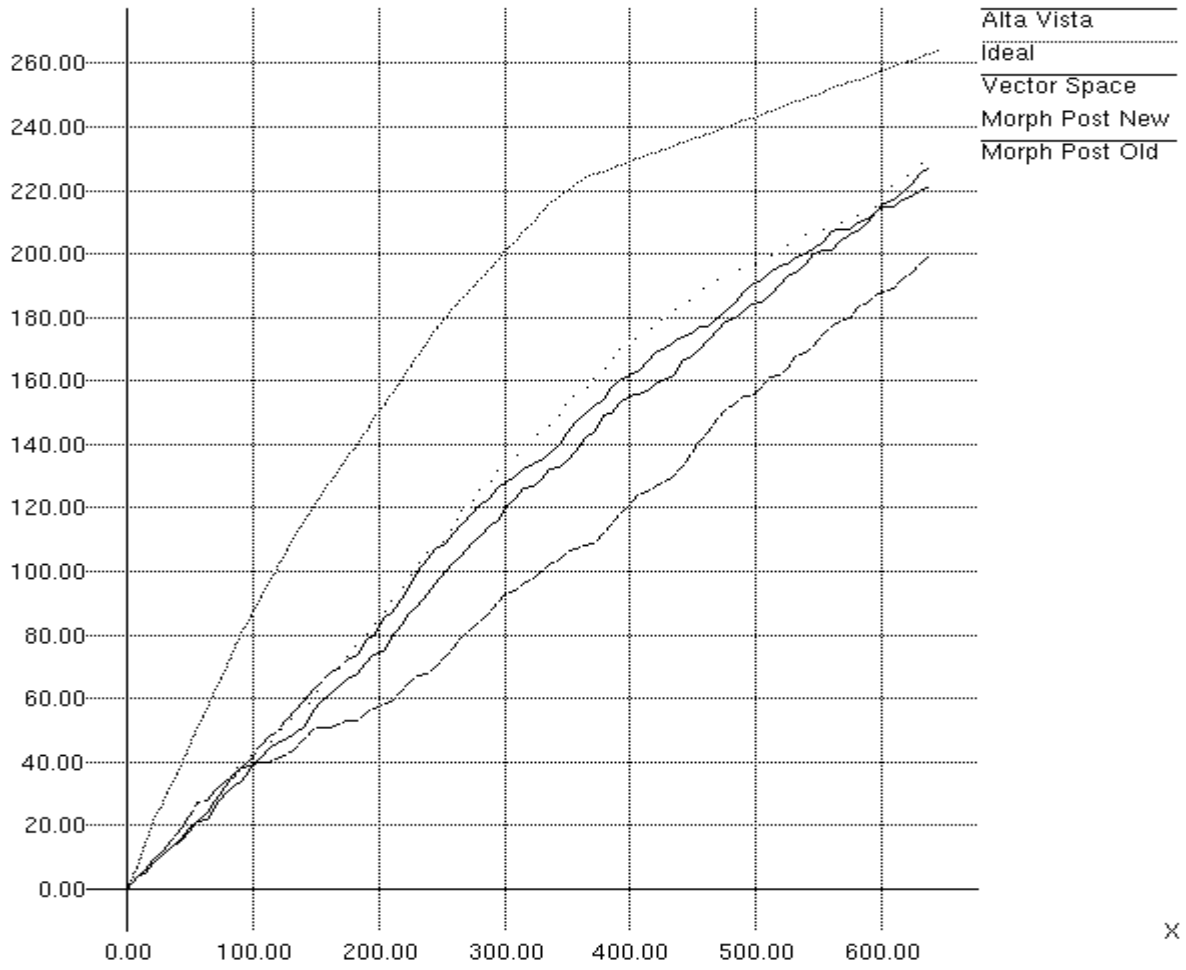

Figure 1: Relative Performance of Vector Space Models

5. PROXIMITY

We designed the Proximity model as an alternative ranking system to the Vector Space model, and based it on the following assumptions:

1. Given any two words from the query, the closer they are in the document, the more relevant the document is.

2. More frequently the pair appears in the document, the more relevant that document is.

3. Word pairs that appear in the document in the same order as in the query (forward pairs), should rank higher than the ones that appear in the inversed order (backward pairs).

In order to implement these assumptions, for each document we counted the number of forward and backward pairs in the whole document, and as well as the number of forward and backward pairs within the three sentence window intervals, throughout the document. We chose the the three sentence window interval instead of one, because experimentally, it gave better results.

We computed four values for each word pair within a document:

- *N1:* number of forward pairs within a three sentence window
- *N2:* number of forward pairs in the whole document
- *N3:* umber of backward pairs within a three sentence windows
- *N4:* number of backward pairs considering the whole document

We then combined these values to compute the document's score, assigning a higher coefficient to word pairs within a three sentence window than to word pairs within the whole document. Also, forward pairs received a higher coefficient than backward pairs:

$C_1 \sum N1 + C_2 \sum N2 + C_3 \sum N3 + C_4 \sum N4$

To implement the previously mentioned design, we designed a matrix, where each entry would be the number of times that a particular pair occured in the document or sentence window. We also used a set of registers where each entry would be the number of times that word appears in that sentence window or that document.

For example, given a query: "US welfare reform" the following set of registers would be created:

US	Welfare	Reform
0	0	0

and the following matrix:

	US	Welfare	Reform
US	x		
Welfare		x	
Reform			x

If given the following setup:

US	Welfare	Reform
2	1	0

	US	Welfare	Reform
US	x	1	0
Welfare	1	x	0
Reform	0	0	x

we encountered a word "reform," the matrix and the registers would change as follows:

US	Welfare	Reform
2	1	0+1

	US	Welfare	Reform
US	x	1	0+2
Welfare	1	x	0+1
Reform	0	0	x

Using the above described technique, we were able to go through the whole document only once. We had one matrix for the pairs in the centence windows, and one for the whole document.

6. BACKING OFF

We used the Backing Off technique when matching query and document tokens. This approach is based on the assumption that NLP algorithms are imperfect and that some information is not important for determining relevance of a document. For example, morphological information is not very important, while the discorse and part of speech information can be helpfull. The less information matches between the query item and the document item, the less certainty that match has and thus the less weight we assign to it.

The Backing Off algorithm used in our system is a 5 stage matching process. If a query term can be successfully matched in the document at any particular stage, it is assigned the appropriate weight. The stages of matching are described below. The weight assigned to the match decreases with each stage.

1. Match query term exactly, including morphology (number, case or person), discorse information (e.g. whether the word is a part of proper nown phrase) and part of speech.

2. Use only root and part of speech information.

3. Account for NLP mistakes in part of speech assignments: EAGLE software sometimes confuses adjectives with proper nouns ("*International*" as an adjective vs. "*International*" as a

proper noun), and gerunds with nouns ("*Frying* pan" vs. "We are *frying* chicken").

4. Match word roots only ignoring morphology and POS tags.

5. Match to a synonym or hyponym. We used the Princeton Wordnet package to provide a list of synonyms and hyponyms and then tred to match given document to a query term expansion from that list, recursively following stages 1 through 4.

7. NOUN PHRASE RECOGNIZER

The Backing Off algorithm can be applied to larger objects, in particular to noun phrases. This approach constitutes the core of our Noun Phrase Recognizer. This module operates as follows:

- Check for an exact match of noun phrases from the query within the first 15 noun phrases of the document. A hit means that the document is accepted.

- Check for matches of geographical terms. If the query contains a geographical term, make sure that it is mentioned in the document. If none of the geographical terms in the query are met in the document, such document is removed from further consideration.

- Count noun phrase matches between the query (including noun phrases its thesaurus expansion.) Use the following Backing Off steps:

  - morphology, POS & Discorse info and word roots
  - only POS information and word roots
  - consider NLP errors (as described above)
  - Try to find a match using a strings of roots a as items (complete or substring match). Also use expansion terms and phrases.

When applying Backing Off techniques to the NP-matches, instead of simply assigning matches a particular coefficient, if at least one match occurred, we counted the number of occurrencies of each query noun phrase in the document. We then normalized this number to receive the density of the queries noun phrases in the document.

8. COMBINING MODELS

We currently have 13 different models. Eight different flavors of Vector Space Model, Words Match with Backoff, Noun Phrases with Backoff, Proximity, String Match and Textzilla(elaborate ?). Out of these, seven are rankers, they assign weight to each document according to its relevance, and the other eight are recognizers, they recognize whether a document is relevant or not, without giving it a rank.

In the beginning, we have desided to use the statistical method to combine the models together. We did some preliminary analysis using the correlation matrix. With that, we could see how much the models are independent of each other. The results were very similar across queries. It turned out, as expected, that the vector space models are very similar to each other.

8.1. Statistical Model

At first, we wanted to use regular regression, but after some research, we desided to use logistic regression, because it is better suited for binary responce data. Our documents are ranked relevant/irrelevant, so the model seemed like the best fit. At first, we ran it on regular coefficients, returned by different models. But then, we have realized that coefficients returned for one query, are incomarable to the ones of the other. We have desided to use the ranks of the coefficients. However, another problem came up, the data sets where so different, that it was impossible to predict a single formula, such that all the coefficients would fit, and the prediction would give some significant result. We were getting one equation for one query, and a totally different for another. Such rigidity made us consider another approach.

8.2. Logical Model

To combine the rankers and recognizers, we have come up with a logical method to determine the relevance of each document from these 13 models, by giving it a proper rank. For each document we first determine the models that rank that document in the top 20. For each such model return a coefficient (C inversely proportional to the rank of that document multiplied by the weight assigned to that model. We took the correlation data mentioned previously to assign the weight of the model. Recognizers that have a very small negative error, receive a very high coefficient. All the Vector Space based models, have a small coefficient, because they are highly correlated between each other, and therefore could be considered as one model. The coefficient is multiplied by the number which is proportional to the normalized inverse rank assigned to the document by a particular model. The final equation for the relevance value is as follows:

$$relevance = \sum [C_i(1 + \frac{3}{\sqrt{RankByModel_i}}) + ... + C_k(1 + \frac{3}{\sqrt{RankByModel_k}})]$$

9. CONCLUSIONS

In conclusion we would like to emphasize that high precision information retrieval is an extremely hard problem and we did not achieve any breakthroughs. The main problem lies in the fact that we do not completely understand the fast processing techniques that humans use in order to process language, and we are forced to rely on the heruistics. It is also extremely hard to formalize the notion of being relevant.

We have also encountered a problem of disambiguating the meanings of the words and identification of the semantic keyu words in the sentences. There does not appear to be a solution to this short of building a knowledge base of *common sense* facts that humans automatically use when processing language.

We seem to achieve pretty good performance (80 - 90%) on the unambiguous queries with uncommon words, such as legal or financial terms. The more general are the query terms the less

is the precision. Our average precision on 10 top documnets is about 32 % (or upto 42% if we account for some irregularities of our evaluation set.) This is notably better than average 14We could not find a comparable experiments in the scientific community as our task is more specific than generic IR engines and our queries are shorter than ones normally used.

REFERENCES

Baldwin, Breck, Christine Doran, Jeffrey C. Reynar, Michael Niv, B. Srinivas, and Mark Wasson. 1997. EAGLE: An extensible architecture for general linguistic engineering. In *To appear in the Proceedings of RIAO-97*, Montreal.

Gerard Salton, Michael J. McGill. 1983. *Introduction to modern information retrieval.* McGraw-Hill, New York.

Strzalkowski, Tomeky, Fang Lin, Jose Perez-Carballo, and Jin Wang. 1997. Building effective queries in natural language information retrieval. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 299–306, April.