Experiments with the Use of Syntactic Analysis in Information Retrieval

Markus Mittendorfer, Werner Winiwarter

Software Competence Center Hagenberg
Hauptstrasse 99
A-4232 Hagenberg
markus.mittendorfer@scch.at, werner.winiwarter@scch.at

Abstract: Up to now, the results of applying sophisticated NL techniques to IR have been mostly disappointing. Our research aims at investigating in detail the role of syntactic analysis in IR and at finding answers to the question why it works better for some queries and worse for others. The final goal is a hybrid algorithm that selectively applies syntactic analysis to certain classes of queries while relying on standard statistical techniques otherwise.

1 Introduction

We present an algorithm for information retrieval that makes use of the syntactic structure of a query. While it has always been felt intuitively that natural language processing (NLP) techniques would benefit information retrieval (IR), experimental results have not confirmed that view so far [St99, Sm99, Vo99]. For the time being, we accept this situation as a fact of life; i.e. our main goal is not to improve IR systems with respect to performance measures like average precision, but to analyse the reasons why NLP based systems do not perform as well as hoped for.

We present an algorithm for information retrieval that makes use of the syntactic structure of a query. An overview and details of the implementation are given. Experiments with a standard test collection are described. The reasons why the approach works better in some cases and worse in others are analysed.

2 Description of the Algorithm

2.1 Overview

The input of the algorithm is a query in natural language, the output is a ranked list of documents. There are three main components to the algorithm (see Fig. 1). One for analysing the **structure** of the query, one for processing the **words** of the query, and one for **combining** the results of the other two into one final result. External resources are a **parser** for the structure component, and possibly **dictionaries**, **thesauruses** or **ontologies** for the words component.

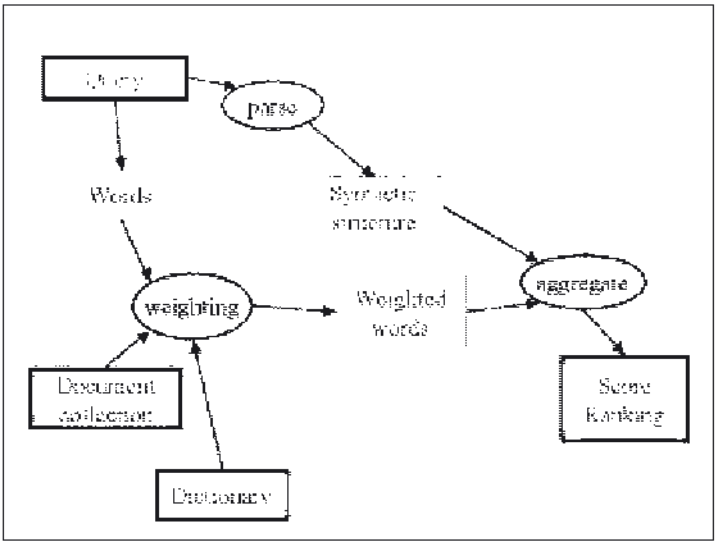

Fig. 1: System architecture

2.2 Details

The first task of the structure component is to build a graph representation of the query, preferably as a tree or a lattice. In our implementation, we use the *Link Grammar Parser* (LGP) [ST93, TSL] for this task. The query is split into sentences by a simple algorithm based on punctuation characters; for each sentence, the LGP provides a set of possible parses. Each parse can be represented as a constituent tree (although this is not the "native" format of the link grammar); the trees are combined into a parse lattice such that identical phrases are shared between trees in order to avoid the computational effort of processing each parse separately (see Fig. 2 for an example).

From this representation, the structure component computes a measure of connectedness c for each pair of words in the query:

$$c(w_1, w_2): \quad \min_a d(N_{w_1}, a) + d(N_{w_2}, a) \tag{1}$$

W is the set of words in the query, N_w is the tree node corresponding to word w, d is a function measuring the distance between two nodes, a is a common ancestor of the nodes. In our implementation, we choose a graph distance measure, namely the sum of the distances to the closest common ancestor of the two words in the parse tree. It does not matter for our purposes if, in the parse lattice, the ancestors or the distances to the ancestors are not unique; in such cases, we choose one of the minimal distances. This quantity measures how strongly two words are bound together by the syntactic structure of the query; the numeric value is used by the combination component.

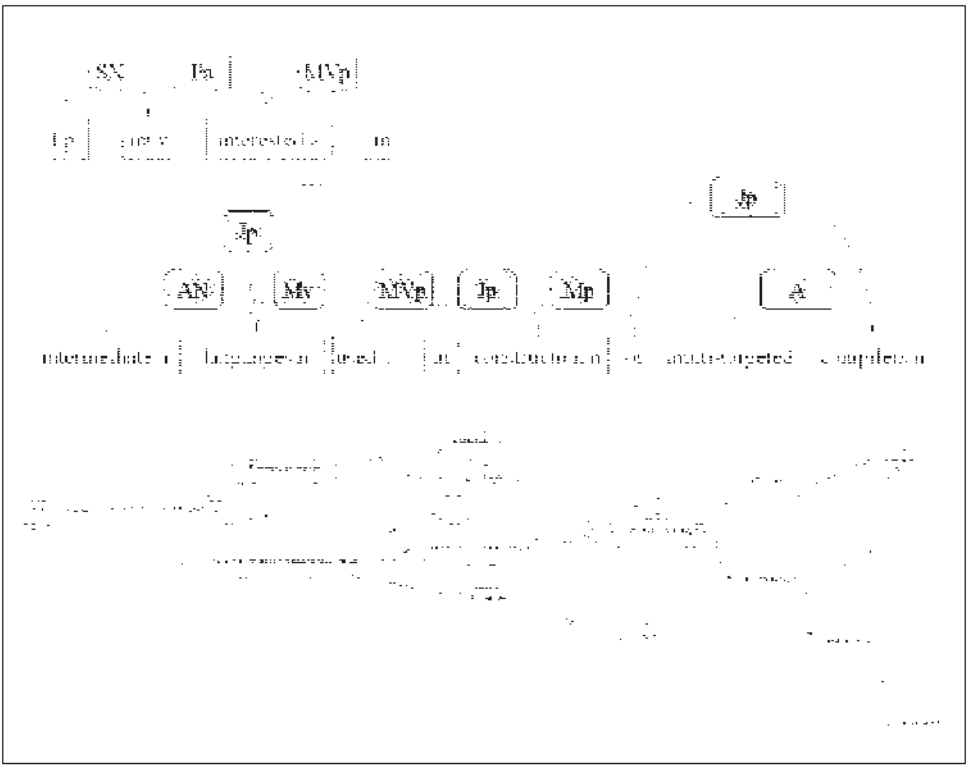

Fig. 2: LGP native output and constituent tree

The task of the words component is to provide candidates for filling the terminal nodes of the syntactic representation, and a weight or score for each of those candidates.

The set of candidates is the same as the set of words W in this version. The function s assigns real numbers to the candidates. In our current implementation, the set of candidates remains the same as the original set of words from the query, i.e. no new words are added; a planned extension for a future version is to allow synonyms for each word.

As for the weights, we currently use the conventional IDF [BR99] method of rating the importance of a word:

$$s(w) := idf(w) \tag{2}$$

As a future modification, we plan to incorporate other resources into the weighting process, such as collection-independent frequency tables, or possibly even resources that allow to modify a weight based on qualitative restrictions like context. In any case, the

flexible architecture of the algorithm permits an arbitrary assignment of weights to the candidates.

Whenever two candidate words are both present in a document, the difference of their positions in the document is calculated (in the case of several occurrences, as the minimal difference); this difference is transformed into the *positional score factor* psf_{doc}, in our implementation by an exponentially decaying function. The reasoning behind this approach is to approximate syntactic connectedness in the document by spatial closeness; the transformation function should be continuous (for graceful behaviour in practice), monotonically decreasing, and tuned experimentally such that it agrees reasonably with the typical (for the language under consideration) spatial distance of words which are in fact syntactically connected:

$$psf_{doc}(w_1, w_2) := e^{-c_{psf} d_{min}(w_1, w_2)} \tag{3}$$

$d_{min}(w_1,w_2)$ is the positional difference of the closest occurrences of w_1 and w_2, measured in words. $psf_{doc}(w_1,w_2)$ is 0 if either w_1 or w_2 do not occur in the document.

Another transformation function is applied to the measure of connectedness prepared by the structure component in order to obtain the *structure score factor ssf*. Again, this function should be continuous and monotonically decreasing. In our implementation, we choose another exponential function:

$$ssf_{doc}(w_1, w_2) := e^{-c_{ssf} c_{min}(w_1, w_2)} \tag{4}$$

The *basic word score bws* is computed from the two individual word scores by an arbitrary two-place function; we suggest to interpret this function as an averaging operator; it is not clear at this moment whether such an operator should be biased towards the greater or the smaller of the two values. In our current implementation, we choose the geometric mean:

$$bws(w_1, w_2) := \sqrt{s(w_1)s(w_2)} \tag{5}$$

Finally, the combination component calculates an *overall score* os_{doc} from the measure of
connectedness, the candidate weights, and the actual presence of candidate words in a document:

$$os_{doc}(w_1) := \max_{w_2 \neq w_1} bws(w_1, w_2) ssf(w_1, w_2) psf_{doc}(w_1, w_2) \tag{6}$$

The final score *ds* of a document *doc* is the sum of the overall scores for the document of all query words:

$$ds(doc) := \sum_{w \in W} os_{doc}(w) \tag{7}$$

3 Experiments

3.1 Data

For our experimental data, we use the well-known CACM test collection (3204 documents, 64 queries) [BR99].

3.2 Results

We compare our approach to a baseline of a simple term weighting approach (basic vector space model without modifications like query expansion, etc.) [BR99].

The current version of the system does not perform significantly different from the baseline: 34.8 % average precision for the baseline, 34.6 % for our system. However, these overall figures are not the result of generally equivalent behaviour; rather, our system outperforms the baseline system in some cases, and is in turn outperformed in others (see Fig. 3).

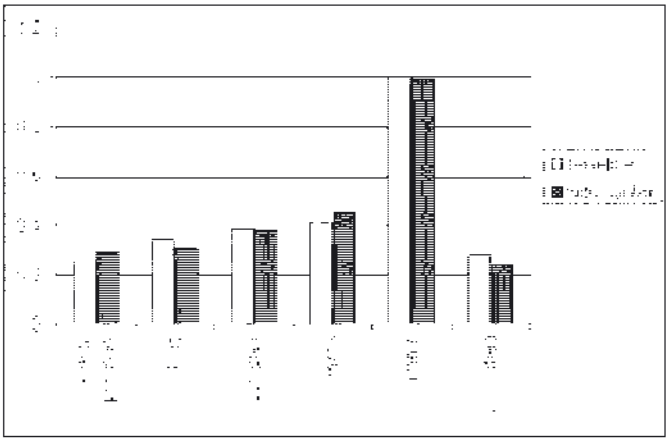

Fig. 3: Average precision by query category

We will present a detailed analysis in which we attempt to categorize queries and reveal some of the reasons why some queries are more or less suited for being handled by our approach.

Table 1 shows two examples of very short queries. Our approach rewards co-occurring words which are in the same region of the parse lattice. In the case of such short queries, however, there is only one region; in the vector space model, there is also a reward for co-occurring words in the same document via the summation of term weights. In other words, in these cases there is no conceptual difference between our approach and the usual vector space model. Since the exact parameterisations of the latter have been opti-

mised over the years, it is not surprising that it should outperform our approach for queries of this type.

Table 1: Very short queries

q#	query text	BL	−syntax
10	*Parallel languages: languages for parallel computation*	71.9%	62.7%
12	*portable operating systems*	43.8%	44.0%

Syntactically complex or ungrammatical sentences like in Table 2 make it difficult for the parser to produce useful output. Furthermore, the complex structures generate larger lattices where semantically related items have a greater distance from each other than in an equivalent simpler formulation.

Table 2: Syntactically complex or ungrammatical sentences

q#	query text	BL	−syntax
4	*I'm interested in mechanisms for communicating between disjoint processes, possibly, but not exclusively, in a distributed environment. I would rather see descriptions of complete mechanisms, with or without implementations, as opposed to theoretical work on the abstract problem. Remote procedure calls and message-passing are examples of my interests.*	7.5%	7.5%
6	*Interested in articles on robotics, motion planning particularly the geometric and combinatorial aspects. We are not interested in the dynamics of arm motion.*	23.7%	37.1%

Some queries have meta-level content (see Table 3), i.e. phrases or whole sentences that do not describe a topic, but the search process itself or the user's preferences. Many queries also have phrases *I'm interested in…*, *find all descriptions of …*, or *the role of …* We believe that functional constructs like *the use of* which are essentially semantically empty hurt our approach more than the vector space approach, because our approach considers such tight syntactic couplings important and assigns disproportionately (in this case) large bonuses to any occurrences thereof. For instance, given the query phrase *the use of information retrieval techniques*, it would also consider as important common occurrences of the word pairs *use – information* and *use – technique*, which can be expected to be rather common, with many occurrences unrelated to information retrieval.

Table 3: "Meta-baggage"

q#	query text	BL	−syntax
22	*I am interested in hidden-line and hidden-surface algorithms for cylinders, toroids, spheres, and cones. This is a rather specialized topic in computer graphics.*	66.2%	64.9%
28	*What is the type of a module? (I don't want the entire literature on Abstract Data Types here, but I'm not sure how to phrase this to avoid it. I'm interested in questions about how one can check that a module "matches" contexts in which it is used.)*	26.7%	42.2%

Queries with several two-or-three word phrases (see Table 4) seem to be well suited for our approach. The effect is the greater the more phrases there are in the query, and the more probable it is for words in the query to co-occur randomly; the latter point is especially important for very frequent words like *time* or *system*, which occur too often for them to be useful as standalone keywords, but which are significant as parts of a phrase like *time sharing* or *operating system*.

Table 4: Queries with several two-or-three word phrases

q#	query text	BL	−syntax
1	*What articles exist which deal with TSS (Time Sharing System), an operating system for IBM computers?*	27.7%	34.3%
9	*Security considerations in local networks, network operating systems, and distributed systems.*	13.3%	14.7%

Finally, some queries include a dominating term (see Table 5), a very specific term that selects exactly the relevant documents, such that the effects of all other terms are negligible. In these cases, there is no difference between the baseline and our system.

Table 5: Dominating terms

q#	query text	BL	−syntax
57	*Abstracts of articles: J. Backus, "Can programming be liberated from the Von Neumann style? A functional style and its algebra of programs", CACM 21 (1978), 613-641, R.A.De Millo, R.J. Lipton, A.J. Perlis, letter to ACM Forum, CACM 22 (1979), 629-630*	100%	100%

4 Conclusion and Future Work

We have described an approach for exploiting the syntactic structure of a query for an IR system that performs better than a standard vector space model in some cases; we have attempted to define these cases by categorizing queries and to analyze the underlying causes for varying results. The approach has been designed with practical application in mind; all weighting factors are continuous, syntactic structure in the documents is approximated by proximity, and only pairs of words are considered in order to avoid combinatorial explosions.

Things that remain to be done are:

- Experiments with different values for the parameters and with different weighting functions: The currently used parameter sets and functions have been found by heuristics and experimentation. Systematic tuning may lead to improved results.

- Analysis of the meta structure of a query: A separate module for eliminating meta-level content and the associated syntactic structure.

- Improved segmentation and tokenisation of a query to make it easier for the parser to handle.

- Finding a criterion for deciding whether to apply the algorithm or not: Based on the insights gained from the present experiments, features correlating with the suitability of a query for the approach, like query length or parse lattice complexity, should be formulated and used to find an appropriate weight for blending the results into the overall system.

- Experiments with other collections.

Bibliography

|BR99| Baeza-Yates, R.; Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, 1999.

|Sm99| Smeaton, A.: Using NLP or NLP Resources for Information Retrieval Tasks. In (In (Strzalkowski, T. Ed.): Natural Language Information Retrieval. Kluwer, 1999; pp. 99-112.

|St99| Strzalkowski, T. et. al.: Evaluating Natural Language Processing Techniques in Information Retrieval. In (Strzalkowski, T. Ed.): Natural Language Information Retrieval. Kluwer, 1999; pp. 113-146.

|ST93| Sleator, D.; Temperley, D.: Parsing English with a Link Grammar. In: Proc. 3rd Intl. Workshop on Parsing Technologies, 1993.

[TSL] Temperley, D.; Sleator, D.; Lafferty, J.: Carnegie Mellon Link Grammar Web Site. http://www.link.cs.cmu.edu/link/index.html.

[Vo99] Voorhees, E.M.: Natural Language Processing and Information Retrieval. Lecture Notes in Computer Science Series, Springer-Verlag, 1999.