Improving search engine Query Expansion techniques with ILP

José Carlos Almeida Santos and Manuel Fonseca de Sam Bento Ribeiro

Microsoft Language Development Center
Tagus Park, 2744-010 Porto Salvo, Oeiras, Portugal
ISCTE-Lisbon University Institute, Portugal
Email: {t-josant,t-manrib}@microsoft.com

Abstract. Query Expansion is the process in which a query is augmented so that it matches more documents, thus potentially increasing the number of relevant results. This is frequently done by spell correcting the query and adding synonyms, morphological variations or other type of relevant data. Query Expansion would, for example, expand 'automobile' with 'car' or 'car' with 'cars'.
Given a concrete set of queries and particular words in these, it is relatively simple to generate lists of candidates that would later reflect as a good or a bad alterations. However, generalizing from a ground truth set, which is a list of alterations considered to be good, to a model which allows the generation of good alterations to an unseen set of words is a challenging task.
In the present work we provide such a model for the English language, discovered with Inductive Logic Programming (ILP). The model induced by ILP is a set of Prolog rules. An important aspect of having the model as rules in Prolog is that, instead of merely being able to classify a given pair $\langle term, candidate \rangle$ as good or bad, we are now able to generate good alterations for a given word, which is the main goal of this investigation.

Keywords: Inductive Logic Programming application, Query Expansion, Word alteration generator

1 Introduction and Motivation

Having a search engine return relevant links for an arbitrary query is a complex task involving the work of several components, most importantly: indexing, query expansion, matching and ranking.

The indexing component crawls the web and updates and annotates a database of valid urls. The query expansion component focuses on spell correcting and augmenting the query with synonyms, morphological variations and other related terms. The matching component is given an augmented query and has to find all the documents in the index that match at least one of the query words. Finally, the ranking component is given a list of documents that match the query

and ranks them according to their relevance, trying to maximize the Normalized Discounted Cumulative Gain, NDCG [2], of the results list.

Since the matching component simply provides the ranker with the set of documents that match a subset of the words in the expanded query, it is important that the query expansion component performs well by adding additional words that are not present in the original query but that are likely to be relevant. For instance, the query 'used automobiles' may be expanded to 'used word:(automobiles cars)', meaning that the words 'automobiles' and 'cars' are equivalent when returning documents to the ranker.

In this paper we consider *Term* to be a token which may be extended with another token. *Candidate* refers to a token which will augment an original token and *Alteration* is a $\langle Term, Candidate \rangle$ pair that may be used to extend a query. Using the previous example, the term *automobile* can be expanded with the candidate *cars*, thus producing the alteration $\langle automobiles, cars \rangle$.

An important aspect of query expansion is thus to be able to generate good alterations for certain terms in a query. An alteration is considered good if it increases the NDCG value of the query. Generating good alterations for an arbitrary word in a query is not a trivial task. The current approach uses a range of techniques, from manually supervised lists of alterations to synonyms extracted from online repositories such as WordNet [1].

In this work we explore a machine learning approach in which we aim to learn rules that identify common patterns for good alterations. Since the creation of new good alterations is a costly process, it is our goal to use the identified rules to generate good candidates for a given term.

From the query expansion component perspective, the aspect of being able to generate new candidates from the learned rules is more important than predictive accuracy. An Inductive Logic Programming approach is thus particularly suitable for this problem as the model an ILP system learns, a set of Prolog rules, can also be easily used to construct new alterations.

2 Experiments

In this section we describe the whole experimentation procedure, starting from processing the raw data to learning an ILP model and finally using the model to construct putative new good alterations.

2.1 Materials

We have gathered from internal resources five data files that were used for these experiments. The first is an English lexicon consisting of roughly 340,000 words. The other four files are query sets gathered from Bing search engine users in Great Britain, which were named *Head1, Tail1, Head2, Tail2*. The numerical suffix refers to the period from which the queries were sampled.

Head queries are randomly sampled from the top 100,000 queries performed by users, whilst Tail queries are randomly sampled from queries which have been issued less than 500 times during the time period 1 or 2.

Each of the 4 query sets is a tab separated file where each line has the format *Query*<*tab*>*Term*<*tab*>*Candidate*. Previous experimentation has determined that changing the word *Term* to *Candidate* in query *Query* increases the overall NDCG value and is, therefore, a good alteration.

In this investigation we are considering query-independent alterations only. Thus, we processed the four query sets to consider only the $\langle Term, Candidate \rangle$ pair. The alteration list was restricted to those in which both *Term* and *Candidate* occur in the English lexicon. We combined the query sets from the same period (Head1+Tail1 and Head2+Tail2) into two datasets *HT1* and *HT2*. The number of positive alterations in these datasets is 6,508 and 5,832, respectively.

Opposing the positive alterations, the negative ones will decrease or maintain the NDCG score of a query. Since our original sets do not contain negative alterations, we have randomly generated them from all the lexicon entries. Considering there are no more than 10 to 20 good alterations per *Term*, a randomly selected pair $\langle Term, Candidate \rangle$ is highly likely to be a bad alteration.

We used a ratio of 100 negatives to 1 positive to ensure the rules found would be specific. To generate the 100 negative examples per positive example, we fixed the *Term* to be the same as in the positive example and selected the 100 *Candidates* randomly from the full English lexicon.

2.2 Problem Modeling

To model this problem we used the features described in Table 1, whose names should be self explanatory.

Predicate type	Background Knowledge Predicates
unary word properties	word_length/2, num_vowels/2, num_glides/2, num_consonants/2, is_possessive/2
$\langle word, alteration \rangle$ properties	edit_distance/3, len_common_prefix/3, len_common_suffix/3, len_longest_common_substr/3, len_longest_common_subseq/3, is_substr/3, is_prefix/3, is_suffix/3
integer comparison	lteq/2, gteq/2

Table 1. Background knowledge predicates

The unary word properties features take a word as input and output an integer. An important binary feature is the edit distance, also known as the Levenshtein-distance [3]. The edit distance between two strings is the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character.

On a previous experiment we focused on purely linguistic features, looking at grammatical and lexical (part-of-speech) categories. This, however, did not produce relevant results, since ILP was attempting to identify ways of expressing the relation of the alteration beyond these categories. Stating that altering "singular to plural" or "present tense to past tense" is not enough, since there are many ways to express these relations. As a simple example, one can form a plural in English by adding "s", "en" or "es" to the singular form, by changing

the vowel sound of the singular (mutated plurals) or by null affixation (adding a null morpheme).

Therefore, we opted to focus on features that work at the pure string level, disregarding more language specific or linguistic information. This also simplifies the process, limiting the input data to a valid word list of a given language and not being dependent on any type of linguistic annotation.

So instead of simply looking for general linguistic rules to define good alterations, such as the referred "singular to plural", these features will attempt to look deeper at the way the alteration is being built. This approach allows the ILP system not to be dependent on linguistic restraints and be free, for example, to identify specific plural morphemes as good alterations, while ignoring others.

2.3 Rule learning

We have implemented a program to generate the ILP background knowledge file containing ground facts for all the features of Table 1 applied to all the positive and negative alterations in our two data sets, *HT1* and *HT2*.

Note that all the predicates in Table 1 are determinate. That is, given the input, there is only one possible output. This determinism makes the hypothesis space relatively small and the coverage computation efficient, thus well suited to ILP systems like Aleph [5] and FOIL [4]. We employed both Aleph 5 and FOIL 6.4, with default settings, using *HT1* as training set and *HT2* as test set.

We selected the top 2 rules of each system in the training set. Table 2 presents the rules and respective precision and recall on the training and test sets. An initial analysis of these results show that the precision and recall in the test set are identical to the ones in the training set, signalling that the rules generalize well. Also, Aleph rules tend to be more general and FOIL more specific.

Rule	An alteration from a term A to a candidate B is good if :	Training set		Test set	
		Precision	Recall	Precision	Recall
1	edit_distance(A,B,C), lteq(C, 3), is_substr(A,B,1)	98.9%	56.6%	98.4%	55.6%
2	edit_distance(A,B,C), lteq(C, 3), len_common_prefix(A,B,D), gteq(D,3), is_possessive(A,0)	98.5%	86.6%	98.2%	84.9%
3	edit_distance(A,B,C), C≤2, len_longest_common_subseq(A,B,D), D≥2, is_possessive(A,0), num_consonants(B,E), E≥1	97.7%	77.5%	97.4%	75.5%
4	num_vowels(B,C), edit_distance(A,B,D), len_common_prefix(A,B,E), D≤E, D≤2, C>D	99.4%	59.5%	99.5%	58.3%

Table 2. Rules found by ILP. First 2 rules were found by Aleph, last 2 by FOIL.

A further analysis of the positive matches proved to be interesting in the sense that the rules grouped several linguistic processes. Thus, we were able identify alterations that fall under specific linguistic rules. The most frequent cases are alterations that expand singular to plural ("ace-aces", "captain-captains") and

add the possessive clitic *-'s* ("car-car's", "video-video's"). But we also note that these matches are not absolute, meaning that they do not cover all that a *singular to plural* or a *form possessive* rule would.

We observe that, together with the high occurrence of *singular to plural* and the formation of the possessive, we also encounter matches with *denominal adjectives* ("intelligence-intelligent", "gnosticism-gnostic", "apocalypse-apocalyptic, "angola-angolan"). Other interesting results extend the lemma of a verb to its *present continuous* ("access-accessing", "edit-editing") or to the *regular past tense* ("eye-eyed") or match contractions ("cannot-can't"). So each rule the system found is not specific to one phenomenon and covers different linguistic processes and that is what makes these rules relevant.

The relevance of these rules can therefore be justified by being independent of any linguistic background. We can, of course, always write this information in such a way that it would generate alterations based on the word formation rules of a given language, but by applying these rules, we identify specific occurrences of the patterns that were matched. Also relevant is the fact that the patterns that these rules match also allow the quick generation of alterations based on a simple list of words, instead of being dependent on annotated lexica. To find alterations with these rules, all that is required is a lexicon of valid words of a given language. This process will not only find alterations based on morphological rules, but others that would not necessarily be annotated within a list, such as orthographical variants ("ann-anne", "whisky-whiskey", "majorca-mallorca") or even, if it is the case, of misspelled words.

2.4 From ILP rules to new alterations

To test the rules found by ILP on the full lexica, we compiled a fresh list of 36 sample words and provided these as terms to the rules of Table 2 so that new candidates would be generated. The list of sample words compiled attempted to match lexical and grammatical categories: *adjectives, nouns (loan words, singular, plural, possessive singular, possessive plural) and verbs (present participles, past tenses, infinitives)*. We have also selected test words by length, ranging from two to eight characters. While analyzing the results, a generated alteration was considered to be relevant if the candidate maintained a semantic approximation to the term. Due to space restrictions, Table 3 shows only the coverage of one word category per rule. Not all the alterations are sensible but most are.

Rule	Word	Candidates
1	financer	financer's,financers,financers',refinancer,refinancers
2	acrylic	acrid,acrolect,acrolith,acromia,acronymic,acrostic,acryl,acrylate, acrylic's,acrylics,acrylics'
3	Moldavian	moldavia,moldavians
4	pianos'	pianism,pianist,piannas',piano,piano's,pianola,pianolas',pianos

Table 3. Candidates generated from all lexicon for a sample list of words

It was noted that rule behavior is not specific to lexical or grammatical categories. Word length; however, seems to be important when it comes to generating new candidates. Lengthier words (i.e. ≥ 5 characters) will return more relevant alterations than smaller words. This is explained by all rules computing the edit distance of the current term to a low value (2 or 3). The larger the word, the more likely it is that an arbitrary string at an edit distance of 1 to 3 will be an invalid word and thus not belong to the lexicon. It is interesting though, that neither Aleph nor FOIL captured the constraint on the word length. This is because the remaining constraints of the rule were enough to discriminate between the positive and negatives. However, if we were to further increase the ratio of negatives to positives, the more likely the word length constraint would be learned by the ILP systems.

3 Conclusions and future work

In this work we showed how ILP was used to learn a model which can generate good alterations. The learned model has both high predictive accuracy and recall and, more importantly, has shown to be easily reversed in order to generate new good alterations. In future work we would attempt lifting the no-context restriction and consider the neighboring tokens of the query term. Improving the quality of both rules and generated alterations might pass from manipulating the lexicon so that it can contain more morphological variations or so that it can be stripped of specific lexical categories (such as articles, preposition or other groups of entries that could be considered stop-words).

A more in-depth linguistic analysis would also be of relevance with the generated data, in which we would attempt to understand the relation between the rules and the linguistic coverage they support.

Currently we are able to construct the candidates for a given term by running the rule against the background knowledge file. However, this process takes about 1 minute of cpu time per term, for all the 4 rules, which limits scalability. We should look at the ILP rules as constraints over the good alterations and, using Constraint Logic Programming techniques and a constructive Prolog implementation of the predicates in Table 1, we would be able to generate the alterations in a more efficient way.

References

1. Christiane Fellbaum, editor. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May 1998.
2. Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20:422–446, October 2002.
3. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.
4. J. Ross Quinlan and R. Mike Cameron-Jones. Induction of logic programs: Foil and related systems. *New Generation Computing*, 13(3&4):287–312, 1995.
5. A. Srinivasan. *The Aleph Manual*. University of Oxford, 2007.