Representing the Context of Equivalent Query Words as a Means of Preserving Search Precision

Timothy A. Musgrove

TextDigger, Inc.
235 Second Street, San Francisco, CA 94105 USA
Tim.Musgrove@cnet.com

Abstract. Information retrieval applications utilizing equivalent terms (e.g. synonyms, alternative spellings. expansions of acronyms and abbreviations), face the problem that the appropriateness of polysemous candidate terms cannot be known *a priori*. This paper describes a method of utilizing the context of the query words in order to provide a clue to selecting, from among the possible equivalent terms, the ones that have a higher utility, where utility is defined as likeliness to improve recall while preserving precision. A method of "seeding" the system with thousands of potential contextual relations enables a rapid run-time assignment of context to the user's present query. The result is the broadening of retrieval through equivalent terms, without the typical concomitant decrease in precision.[1]

1. Introduction

One of the main problems in keyword searches is that of missing relevant documents wherein the relevant information is expressed *in different words entirely* than were entered in the original query. Granted, for simple searches with very popular words, and where relevant information is plentiful, this is not much of a problem (e.g. as of this writing, the queries "San Francisco weather" and "Brittney Spears photos," when entered on any major Internet keyword search engine, yield plentiful, relevant results on the basis of an exact keyword match) but for longer queries, and searches where the relevant phrasing is hard to predict, it can be a nightmare.

Examples of the questions that arise in this predicament are: How can a search engine recognize where there are synonymous words for the query words, e.g. that "mother-daughter matching sleeping gowns" matches "adult-child coordinated night gown set"? How can a search engine recognize that "hotel room with a view of the Golden Gate Bridge" matches "suite that provides a panorama of the entire Bay Area skyline?"

[1] The software described herein is the subject of patents pending with the USPTO on behalf of TextDigger, Inc.

1.1 Conventional approaches to the problem of missed candidates

Probably the most straightforward approach to take is to automatically add synonyms to a query. This is easily done by simple look-ups in a machine readable thesaurus or WordNet. Why not just add all the synonyms automatically (or at least the most common ones), and then search for all of them at once? Unfortunately, this approach encounters some very big roadblocks.

1.1.1 The problems in the attempted solution

There are several problems in auto-adding synonyms to queries:

1. Words have many different senses
2. Words have many synonyms in each sense
3. *Worst of all*: most synonyms themselves have other senses which are **not** synonymous with the original word.

For example, while “bank” can mean the turning of an aircraft, the word “turn,” also has other senses (such as in “it’s your turn” or “the turn of the century”, etc.) which have nothing to do with *any of the* senses of “bank.”

This means that automatically adding all the synonyms of every query term, usually creates *more* irrelevant hits, not fewer. While the synonyms do give the benefit of enabling the search engine to find more relevant information, that effect is overshadowed by the creation of a mountain of additional, irrelevant search results. In other words, while increasing recall, it destroys precision.

We should then suppose that the better thing to do, if at all possible, would be to tackle the problem of word sense disambiguation head on, because then we might have a better chance of picking the synonyms that are appropriate. Research in word sense disambiguation has had substantial results, but we must examine the applicability of lexical WSD to search.

1.1.2 Conventional Methods of Word Sense Disambiguation (WSD)

Conventional methods of word sense disambiguation generally proceed along the following lines:

1. Manually sense-tagging a large corpus (marking each word as to its canonical sense). One will use most of this data as the “training data” while saving a minority portion for the “testing data.”
2. Using the training data, for each sense of each word, extract contextual features (e.g. record which words are found frequently occurring next to, or in the same sentence as, or within *n* words distance of the target word).
3. Determine common patterns in the contextual features (e.g. apply any standard machine learning algorithm, whether that be neural nets, or case-based reasoning, or genetic classifiers, or whatever) to enable classification among several senses of a word, and validate the classifier on the testing data.

After the foregoing project is completed, then based on the determined patterns (or feature value-sets, or derived rules concerning them), new occurrences of words (given a surrounding context, i.e. the text before and/or after the word) can be assigned a guess, or a probability, of having certain senses, i.e. be classified according to their canonical sense.

A considerable amount of research and debate has surrounded steps 2 and 3 of this process, and it is no doubt fruitful to investigate and optimize these phases. However, for our present purposes, what is remarkable is that almost all of the conventional methods of WSD agree on Step 1. A large set of manually tagged training data is presumed in the vast majority of methods attempted in word sense disambiguation – it is the point of departure for the various competing methods. This mere fact, by itself, presents the biggest limitation for search applications.

1.1.3 Limitations of Conventional WSD

The need to manually tag a corpus containing numerous example sentences for each word in a variety of contexts, presents not one but several problems to the designer of an open-ended search application[2]:

1. The manual labor cost, in number of hours, is mind-boggling. It can require several person-months to manually tag the several thousand example sentences that are required as training data for disambiguating *one single word in the English language* as an example of an algorithm.
2. The labor in question is not just any sort of labor, but linguistically trained labor. The tagging must be performed by those who understand grammar, parts of speech and canonical word senses, and are very literate. This skill requirement extends far beyond that of the worker typically employed to do standard data processing.
3. Many word senses simply do not have enough examples in the corpus, to provide a sufficient baseline for subsequent disambiguation, even if the data were all tagged.
4. Given that for a search application, what is to be disambiguated is a query word, and since queries are usually short (four words or less), there is not enough context for most WSD methods to perform their analysis.

Given these formidable roadblocks to scalable WSD for generalized search applications, we have sought an alternative approach to distinguishing those among numerous synonyms or equivalent terms that are most appropriate to a given query. This method utilizes the text corpus and the history of past queries, on an automated basis, and does not require manually tagged training data. We will now describe this method in greater detail.

2 An Alternative Approach

The essence of the TextDigger approach is to determine, before the actual keyword search, which equivalent terms are most likely to preserve precision while increasing

[2] For an example of the type of approach here described, see (Bruce and Weibe 1994). There are also a few WSD systems that do not require manual tagging of training data, such as (Yarowsky 1995), but they still have limitations (such as recognizing only canonical senses), and they typically do not address problems 3and 4 on this list.

recall. The chief features of the approach being discussed here, are (1) to assign a prior probability to every WordNet synonym reflecting its probability of having indeed the same sense as the original word and (2) to determine which of the various potential equivalent terms, when substituted one at a time in the context of the remaining, unaltered query terms, yields multiple hits in semantically continuous segments of the hit documents. A final contextual synonymy confidence score is then determined as a function of both these measures.

In addition to requiring that candidate synonyms pass a threshold on this score, they are subjected also to an absolute requirement of multiple contextual attestations in the source documents. Only those equivalent terms that meet these conditions are deemed to "pass" with regard to the context represented by the other terms in the original query. The hope is that some meaningful subset of query terms will pass this test, which are far more likely to preserve precision when used together in a disjunctively expanded query, than would be the case with the entire, unaltered set of potentially equivalent terms.

2.1 Related work

Our approach is similar in some ways to (Ramakrishnan, 2003) in that it does not "pin down" a single word sense for every query term, but involves rather the demoting of certain senses while qualifying others as more viable, and thus can be seen as a "soft" WSD. It differs however in the manner in which the probabilities of each word sense are calculated, and also in its granularity, as our method assigns scores not at the level of WordNet senses, but at the level of every synonym within each sense.

In other ways, our approach is similar to that of (Agirre and Martinez, 2000) in that it uses the Web as a resource to help automate the disambiguation task. However, our analysis of Web content is more intensively concerned with the context of co-occurring query words (and their candidate synonyms), and does not employ a Bayesian network, but rather a context validation model that combines statistical and rule-based approaches as described herein.

2.2 Outline of method

By "prior probability" we mean the probability of relevance of a candidate synonym, based purely on (1) the distribution of senses listed in WordNet, and (2) the bias of our application as to part-of-speech (POS), but without any reference yet to the actual source documents. We calculated this prior probability for a word by examining its sense numberings in all four parts-of-speech in WordNet, and giving each one a weighting inversely proportional to the longest sense-depth, e.g. where the largest number of senses in any POS was 6, the 6th sense was weighted a 1, and the 1st sense in every POS was weighted 6 (even in a POS having less than 6 senses). We then scored each sense of each POS as a function of this sense weight divided by the total of all sense weights across all POS. We then factored this against both an internal and external POS-bias. The "internal POS bias" is the one derivable from WordNet by counting to see which POS has the most senses listed, and calculating a proportional bias accordingly. The "external POS bias" comes from a usage profile of our search application, showing that users most frequently search for nouns, and least frequently for adverbs, with the frequency of their searching for verbs and adjectives falling in the middle.

By "semantically continuous" segments of text we mean the standard approach within NLP (e.g. Hearst 1997) of checking semantic distance between content words in contiguous text blocks, and noticing when there is a marked shift in semantic space. If the substitute term is found together with the original query terms in the text corpus, but not in a semantically continuous block of text, it is disqualified for our present purposes. This rules out a great number of would-be synonyms.

By requiring multiple attestations, we mean simply to reflect that finding one hit is not a good enough indicator of the equivalent term's appropriateness. We look for two or more hits within the corpus. Our early prototyping showed that this greatly improved precision, whereas asking for broader attestation (say, three or four examples) did not substantially increase the eventual precision beyond what was gained with our double-attestation threshold.

An example of this process can be seen in Table 1, where the query "hotel with activities for kids" was tested on a major keyword search engine, in several forms. One will notice very many synonyms in the unaltered WordNet synset (twenty-six), and then a much shorter list (four) after our contextual processing. This is a 15% synonym retention rate, for this example. Across forty sample queries, the average synonym retention rate was 28% (i.e. 76% of synonym candidates were discarded).

For this test, hundreds of search results were manually graded, comprised of the top twenty results for each of forty sample queries, at four different phases: unexpanded, expanded but unprocessed, processed with context analysis, and processed with both context and proper-name analysis.

2.3 Measuring recall

Since recall is difficult to judge when searching the entire Web, and since it is non-controversial in this case that recall will be improved, only the precision was measured as a strict percentage, while recall was monitored in relative or "pragmatic" terms, as the number of new, relevant hits appearing in the top twenty that did not (and never could) come up on the unaltered query (due to their using different words). For example, on the aforementioned query, after expansion we found in the top twenty hits a web page for the Denali Bluffs Hotel, describing a "Tundra Wilderness Tour" recommended for families with "children" (and no mention of the original query word "kid"), and we found another web page explaining that the Grand Hyatt in Beijing has an affiliation with the Imperial Tours Company, offering a "family and child-oriented luxury tour of China." In total, four relevant hits in the top twenty were found that would not have satisfied the original query on a strict keyword-matching basis, and when viewed against the nineteen relevant hits found originally in the top twenty, we call this a "relative" increase in recall of 21% (i.e., 4/19).

Across forty sample queries, the average relative increase was higher – about 38%. However, the increase varied dramatically from a low of 7% (1/14) to a high of 300% (9/3), and the standard deviation was rather large (.18), suggesting that a larger set of sample queries should be tested. Granted, this measure is suggestive rather than precise, but the main point is, expecting recall will generally improve in most cases, to see if precision can be recovered after query expansion to near the point where it was before expansion.

Table 1. Example of contextual synset analysis of "hotel with activities for kids"

Results of expanding with synonyms	Unexpanded	Unprocessed [full list from WordNet 2.0]	After Processing with context = [hotel, activities, kids]	After Proper-Name Pruning
Synsets	N.A.	[hotel] [activity, action, activeness, bodily process, body process, bodily function, natural process, natural action] [kid, banter, chaff, child, fry, jolly, josh, kidskin, Kyd, minor, nestling, nipper, pull leg, shaver, small fry, tiddler, tike, tyke, young goat, youngster]	[hotel] [activity] [kid, child, jolly, nestling, youngster]	[hotel] [activity] [kid, child, nestling, youngster]
Precision [top twenty hits]	.95	.60	.90	.95

2.4 Measuring precision

As the following table shows (Table 2), the answer appears to be that context processing alone cannot quite restore the same level of precision of an unexpanded query, but when proper-name pruning is added (which could be considered also a type of context analysis which brings in additional background knowledge, i.e. lists of established proper names), the answer is Yes.

These results were arrived at by our manually tuning the threshold of the final confidence score until we simultaneously enjoyed substantial recall improvements and no loss of precision. Specifically, the final confidence score was normalized on a 0-1.0 scale, and we ended up placing the threshold at 0.68. Precision at this threshold actually ended up slightly higher than that of unaltered queries, though our goal was only to preserve it, and the slight increase we won is probably not statistically significant, given our relatively small sample size.

Table 2. Summary of search precision rates of results (top twenty hits)

Original query	Not expanded	Expanded but not processed	After context processing	After proper name pruning
"hotels with activities for kids"	95%	60%	90%	95%
"parking near Stratford Court hotel"	75%	40%	70%	80%
"winter hiking trails in the Bay area"	95%	55%	85%	90%
"guided tours of caves in France"	80%	65%	75%	95%
"nightgown with buttons all the way down"	35%	40%	60%	65%
"replacement parts for microwave oven"	70%	50%	65%	65%
"major earthquakes in Asia"	80%	30%	70%	75%
"conservative criticism on Bill of Rights"	95%	45%	55%	90%
"missile treaties with China"	75%	40%	75%	75%
"laws on growing marijuana in Oregon"	85%	65%	90%	90%
AVERAGE (of the above 10 queries)	79%	49%	74%	82%
AVERAGE (of all 40 sample queries)	76%	41%	72%	80%

3 A limitation of the approach, and a method of addressing it

The chief limitation of the approach is its computational intensiveness and latency of results. The user must wait for extensive analysis of the query terms. The system literally issues many dozens of slightly varying queries and post-processes all the results, in order to determine the contextually qualified synonyms.

To get around this, we utilized a list of the top queries from a major online search site, and used it to determine frequently co-occurring query words. We then pre-processed these word sets for context, and stored them in a "context cache". When a new query comes in that matches one of these wordsets (as a subset match), the cached context is used instantaneously instead of generating the context from scratch.

As of this writing we are not at the point of deploying a run-time system on enterprise-class servers, and therefore, any measuring of the actual time-savings is not informative or useful. However, a study of the ten queries above showed the following combinations (Table 3) would have been pre-processed:

Table 3. Pre-seeding context with frequently combined query words

Pre-processed word combinations from a list of previously inputted user queries
“activities” + “kids”
“hiking” + “trails”
“Bay” + “area”
“missile” + “treaties”
“treaties” + “China”
“laws” + “marijuana”

The word combinations in Table 3 account for 11 of 39, or 28%, of the content words (i.e. words kept after “stop words” were discarded) across the first ten sample queries, and suggests that a substantial real-time savings could be won through pre-seeding the context base. There is no reason not to suppose that a larger collection of query word combinations would lead to an even greater match rate between the pre-established context base and the stream of incoming, novel queries.

3.1 Matching Context and Situation

There is one complication of using the clever trick of pre-seeding a context base: it now becomes non-trivial to match a stored context with a novel, incoming search. For we are seeking to save computation time by finding one or more partial matches. Note in Table 3 the occurrence of both [“missile” + “treaties”] and [“treaties” + “China”], which intersect in respect of the word “treaties.” In this case, we take the intersection (not the union) of the synonyms in the context base for these two vectors, which is the more conservative approach for guarding precision.

It is informative in this regard to examine what a context vector in the context base really means in practical terms. To take our familiar example, one set of vectors produced from our tests recommends to us that :

1. “Kids” in the context of “hotel” is likely to mean the same as any of “children, youngster, infant, nestling” would mean in the same context, and each of these latter words is not likely to mean anything *other* than “kid” in that same context.
2. “Kid” in the context of “activity” is likely to mean the same as any of “children, youngster, tiddler, infant, jolly” would mean in the same context, and each of these latter words is not likely to mean anything *other* than “kid” in that same context.
3. Taking “kid” in the presence of both “hotel” and “activity”, it is likely to mean the same as any of “children, youngster, infant” (the intersection of the previous two lists) – and likewise, these latter words are not likely to mean anything besides “kid” in the same context (i.e. the context of “hotel” and “activity”).

Note that the intersection of these vectors in line 3, above, does not produce exactly the same list of final synonyms as our direct approach did earlier, but it is rather close, and has the benefit of eliminating a good deal of real-time latency for the user. We presume that a large context base of such vectors, easily joinable, could prove useful to many applications in the areas of information retrieval and presentation, including but not limited to our general search application.

3.2 Forthcoming work

Our further development involves using meta-tags and ontological classification information that is embedded in many of the search documents, to further refine the assignment of context. This is why we speak of matching the context to the "situation" rather than merely to the "query"; the user's profile and session history are also additional clues to the selection of the appropriate synonym context. In view of this fact, we call the combination of the query along with the user's profile and history, the "situation" of a query. We also hope to run tests on larger sets of sample queries and on some additional standard test data, such as TREC.

4 Conclusion

Representing a "synonym context" as the subset of possible synonyms which are multiply attested in the presence of contextual query words in semantically continuous segments of the source documents, has been shown to be a powerful technique for increasing recall of keyword searches without a loss in the final precision of the search. The increased computation that is required can be at least partly reduced by pre-seeding the process with a "context base" of the pruned synsets of frequently co-occurring query words. Additional clues from the overall situation of the query (e.g. the user's interest profile and ontological navigation history) may enable the precision to reach an even higher level in subsequent stages of this project.

References

Agirre, E. and Martinez, D. Exploring automatic word sense disambiguation with decision lists and the Web. In *Procedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content*. Saarbrcken, Germany (2000)

Bruce, R. and Weibe, J. Word-sense disambiguation using decomposable models. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics,* 139-146. (1994)

Hearst, M. Texttiling: Segmenting text into Multi-Paragraph Subtopic Passages. *Computational Linguistics* 23(1): 33-64. (1997)

Ramakrishnan, G., Prithviraj, B., Deppa, A., Bhattacharya, P., and Chakrabarti, S. Soft Word Sense Disambiguation. In *Proceedings of the Second International WordNet Conference – GWN 2004*, 291-298. Masaryk University Brno, Brno, Czech Republic, December (2003).

Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods, *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics,* 189-196. (1995)