SW-AG: Local Context Matching for English Lexical Substitution

George Dahl, Anne-Marie Frassica, Richard Wicentowski
Department of Computer Science
Swarthmore College
Swarthmore, PA 19081 USA
{george.dahl, afrassi1}@gmail.com, richardw@cs.swarthmore.edu

Abstract

We present two systems that pick the ten most appropriate substitutes for a marked word in a test sentence. The first system scores candidates based on how frequently their local contexts match that of the marked word. The second system, an enhancement to the first, incorporates cosine similarity using unigram features. The core of both systems bypasses intermediate sense selection. Our results show that a knowledge-light, direct method for scoring potential replacements is viable.

1 Introduction

An obvious way to view the problem of lexical substitution is as a sense disambiguation task. For example, one possible approach is to identify the sense of the target word and then to pick a synonym based on the identified sense. Following Dagan et al. (2006), we refer to this as an *indirect* approach. A system using an indirect approach must have access to a list of senses for each target word, and each sense must have a corresponding list of synonyms. Though one can use a predefined sense inventory, such as WordNet, the granularity of the sense inventory may not be appropriate for the task. If the sense inventory is too fine-grained, then picking the correct sense may be needlessly difficult. Conversely, if it is too coarse, picking the correct sense may not narrow down the list of potential substitutions sufficiently.

To avoid these problems, we propose a *direct* approach, which will break the problem into two steps: for each target word, generate a list of candidate synonyms; then rank each synonym for its quality as a replacement. Although our second system makes use of some sense information, it is used only to rerank candidates generated using a direct approach.

We describe two systems: the first is a purely direct method based on local context matching, and the second is a hybrid of local context matching and wider context bag-of-words matching. Both are knowledge-light and unsupervised.

2 Methods

As mentioned above, we divide the task into two steps: compiling a list of synonyms and then, for each test instance, ranking the list of appropriate synonyms. Both of our systems create lists of candidate synonyms in the same way and only differ in the way they arrive at a ranking for these candidates.

2.1 Compiling a substitution lexicon

We begin by compiling a list of candidate synonyms for each target word. Following Dagan et al. (2006), we will refer to this list of synonyms as our *substitution lexicon*. The performance of our system is limited by the substitution lexicon because it can only pick the correct replacements if they are in the lexicon. The substitution lexicon available to our scoring system therefore determines both the maximum attainable recall and the baseline probability of randomly guessing a correct replacement.

One approach to generating a substitution lexicon is to query WordNet for lists of synonyms grouped by the senses of each word. While WordNet has its advantages, we aimed to create a knowledge-light

Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pages 304–307,
Prague, June 2007. ©2007 Association for Computational Linguistics

system. A more knowledge-free system would have used a machine readable dictionary or a large natural language sample to retrieve its synonyms (see, for example, Lin (1998)), but our system falls short of this, relying on Roget's New Millennium Thesaurus[1] (henceforth RT) as a source of synonyms. Though this thesaurus is similar to WordNet in some ways, it does not contain semantic relationships beyond synonyms and antonyms. One important advantage of a thesaurus over WordNet is that it is easier to obtain for languages other than English.

We used the trial data to ensure that the quality of the list compiled from RT would be satisfactory for this task. We found that by using the synonyms in WordNet synsets[2] as our substitution lexicon, we could achieve a maximum recall of 53% when using an oracle to select the correct synonyms. However, using the synonyms from RT as the substitution lexicon led to a maximum recall of 85%.

Querying RT for the synonyms of a word returns multiple entries. For our purposes, each entry consists of a Main Entry, a part of speech, a definition, and a list of synonyms. Many of the returned entries do not list the query word as the Main Entry. For instance, given the query *"tell"*, RT returns 115 entries: 7 whose Main Entry is "*tell*", an additional 3 that contain "*tell*" (e.g. *"show and tell"*), and the remaining 105 entries are other words (e.g. *"gossip"*) that list *"tell"* as a synonym. Where the Main Entry matches the query, RT entries roughly correspond to the traditional notion of "sense".

In order to reduce the number of potentially spurious synonyms that could be picked, we created a simple automatic filtering system. For each RT query, we kept only those entries whose Main Entry and part of speech matched the target word exactly[3]. In addition, we removed obscure words which we believed human annotators would be unlikely to pick. We used the unigram counts from the Web 1T 5-gram corpus (Brants and Franz, 2006) to determine the frequency of use of each candidate synonym. We experimented with discarding the least frequent third of the candidates. Although this filtering reduced our maximum attainable recall from 85% to 75% on the trial data, it significantly raised our precision.

2.2 Ranking substitutions

We created two systems (and submitted two sets of results) for this task. The first system is fully described in Section 2.2.1. The second system includes the first system and is fully described in the remainder of Section 2.2.

2.2.1 Local context matching (LCM)

Our first system matches the context of target words to the context of candidate synonyms in a large, unannotated corpus. If the context of a candidate synonym exactly matches the context of a target word, it is considered a good replacement synonym. Context matches are made against the Web 1T corpus' list of trigrams. Though this corpus provides us with a very large amount of data[4], to increase the likelihood of finding an appropriate match, we mapped inflected words to their roots in both the corpus and the test data (Baayen et al., 1996).

The context of a target word consists of a set of up to 3 trigrams, specifically those trigrams in the test sentence that contain the target word. For example, the context of *"bright"* in the sentence[5] "... who was a *bright* boy only ..." is the set {"was a *bright*", "a *bright* boy", "*bright* boy only"}.

Once we identified the set of context trigrams, we filtered this set by removing all trigrams which did not include content words. To identify content words, we used the NLTK-Lite tagger to assign a part of speech to each word (Loper and Bird, 2002). We considered open class words (with the exception of the verb *to be*) and pronouns to be content words. We call the filtered set of trigrams the test trigrams. From the above example, we would remove the trigram "was a *bright*" since it does not contain a content word other than the target word.

We match the test trigrams against trigrams in the Web 1T corpus. A corpus trigram is said to match one of the test trigrams if the only difference between them is that the target word is replaced with a candidate synonym.

A scoring algorithm is then applied to each candidate. The scoring algorithm relies on the test tri-

[1] `http://thesaurus.reference.com`

[2] excluding the extended relations such as hyponyms, etc.

[3] Since RT was not always consistent in labeling adjectives and adverbs, we conflated these in filtering.

[4] There are over 967 million unique trigrams in this corpus

[5] Excerpted from trial instance 1.

grams, denoted by T, the set of candidate synonyms, C, and the frequencies of the trigrams in the corpus. Let $m(t, c)$ be the frequency of the corpus trigram that matches test trigram t, where the target word in t is replaced with candidate c. The score of a candidate c is given by:

$$\text{score}(c) = \sum_{t \in T} \frac{m(t, c)}{\sum_{x \in C} m(t, x)}.$$

The normalization factor prevents high frequency test trigrams from dominating the score of candidates. The candidates are ranked by score, and the top ten candidates are returned as substitutions.

2.2.2 Nearest "synonym" neighbor

In some cases, the words in the local context did not help identify a replacement synonym. For example, in test instance 391 the trigrams used in the local context model were: "by a *coach*", "a *coach* and", and "*coach* and five". The first two trigrams were removed because they did not contain content words. The final trigram does not provide conclusive evidence: the correct synonym in this case can be determined by knowing whether the next word is "players" (*coach* = *instructor*) or "horses" (*coach* = *vehicle)*. Without backoff, extending the local context model to 5-grams led to sparse data problems.

Rather than match exact n-grams, we use a nearest neighbor cosine model with unigram (bag of words) features for all words in the target sentence. For each instance, an "instance vector" was created by counting the unigrams in the target sentence.

Since the Web 1T corpus does not contain full sentences, we matched each of the instance vectors against vectors derived from the British National Corpus[6]. For each candidate synonym, we created a single vector by summing the unigram counts in all sentences containing the candidate synonym (or one of its morphological variants). We ranked each candidate by the cosine similarity between the candidate vector and the instance vector.

2.2.3 Nearest "sense" neighbor

Manual inspection of the trial data key revealed that, for many instances, a large majority (if not all) of the human-selected synonyms in that instance were found in just one or two RT entries. This not altogether unexpected insight led to the creation of a second nearest neighbor cosine model.

We first created instance vectors, following the method described above. However, instead of creating a single vector for each candidate synonym, we created a single vector for each "sense" (RT entry): for each RT entry, we created a single vector by summing the unigram counts in all BNC sentences containing any of that entry's candidate synonyms (or morphological variants). We ranked each candidate sense by the cosine similarity between the sense vector and the instance vector.

This method is not used on its own but rather to filter the results (Section 2.2.4) of the nearest "synonym" neighbor method. Also note that while we used the "senses" provided by RT for this method, we could have used an automatic method, e.g. Lin and Pantel (2001), to achieve the same goal.

2.2.4 Filtering by sense

The nearest synonym neighbor method underperformed the local context matching method on the trial data. This result led us to filter the nearest neighbor results by keeping only those words listed as synonyms of the highest ranked senses, as determined by the nearest sense neighbor model. This proved successful, increasing accuracy from .41 to .44 (for instances which had a mode) when we kept only those synonyms found in the top half of the senses returned by the nearest sense model.[7]

We attempted the sense filtering method on the local context model but found that it was less successful. No matter what threshold we set for filtering, we always did best by not doing the filtering at all. However, applying the filtering to only *noun* instances, keeping only those synonyms belonging to the single most highly ranked sense, increased our accuracy on nouns from .51 to .57 (for instances which had a mode). This surprising result, used in the following section, requires further investigation which was not possible in the limited time provided.

2.2.5 Model Combination

A straightforward model combination using the relative ranking of synonyms by the filtered local

[6] http://www.natcorp.ox.ac.uk/

[7] We rounded up if there were an odd number of senses, and we always kept a minimum of two senses.

	P	R	Mode P	Mode R
all	35.53	32.83	47.41	43.82
Further Analysis				
NMWT	37.49	34.64	49.11	45.35
NMWS	38.36	35.67	49.41	45.70
RAND	36.94	34.52	48.94	45.72
MAN	33.83	30.85	45.63	41.67

Table 1: SWAG1:OOT results

	P	R	Mode P	Mode R
all	37.80	34.66	50.18	46.02
Further Analysis				
NMWT	39.95	36.51	52.28	47.78
NMWS	40.97	37.75	52.25	47.98
RAND	39.74	36.36	53.61	48.78
MAN	35.56	32.79	46.34	42.88

Table 2: SWAG2:OOT results

context matching (FLCM) model[8] and the filtered nearest neighbor (FNN) model yielded results which were inferior to those provided by the FLCM model on its own. Examination of the results of each model showed that the FLCM model was best on nouns and adjectives, the FNN model was best on adverbs, and the combination model was best on verbs. Though limited time prohibited us from doing a more thorough evaluation, we decided to use this unorthodox combination as the basis for our second system.

3 Results

We submitted two sets of results to this task: the first was our local context matching system (SWAG1) and the second was the combined FLCM and FNN hybrid system (SWAG2).

Our systems consistently perform better when a mode exists, which makes sense because those are instances in which the annotators are in agreement (McCarthy and Navigli, 2007). In these cases it is more likely that the most appropriate synonym is clear from the context and therefore easier to pick.

It is hard to say exactly why SWAG2 outperforms SWAG1 because we haven't had enough time to fully analyze our results. Our decision to choose different systems for each part of speech may have been partially responsible. For example, both LCM (used in SWAG1 and SWAG2) and the nearest neighbor cosine comparison algorithm (used in SWAG2) performed poorly on verbs on the trial data. The voter described in the SWAG2 discussion always performed better on verbs than either system did individually, so this may account for part of the higher precision and recall.

[8]Filtering was done only on nouns as described above.

4 Conclusions

Our results show that direct methods of lexical substitution deserve more investigation. It does indeed seem possible to successfully do lexical substitution without doing sense disambiguation. Furthermore, this task can be accomplished in a knowledge-light way. Further investigation of this method could include generating the list of synonyms using a completely knowledge-free approach.

References

R.H. Baayen, R. Piepenbrock, and L. Gulikers. 1996. CELEX2. LDC96L14, Linguistic Data Consortium, Philadelphia.

T. Brants and A. Franz. 2006. Web 1T 5-gram, ver. 1. LDC2006T13, Linguistic Data Consortium, Philadelphia.

I. Dagan, O. Glickman, A. Gliozzo, E. Marmorshtein, and C. Strapparava. 2006. Direct word sense matching for lexical substitution. In Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics.

D. Lin and P. Pantel. 2001. Induction of semantic classes from natural language text. In Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics.

E. Loper and S. Bird. 2002. NLTK: The Natural Language Toolkit. In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics.

D. McCarthy and R. Navigli. 2007. SemEval-2007 Task 10: English lexical substitution task. In Proceedings of SemEval-2007.