An Active Learning Approach to Finding Related Terms

David Vickrey
Stanford University
dvickrey@cs.stanford.edu

Oscar Kipersztok
Boeing Research & Technology
oscar.kipersztok@boeing.com

Daphne Koller
Stanford Univeristy
koller@cs.stanford.edu

Abstract

We present a novel system that helps non-experts find sets of similar words. The user begins by specifying one or more seed words. The system then iteratively suggests a series of candidate words, which the user can either accept or reject. Current techniques for this task typically bootstrap a classifier based on a fixed seed set. In contrast, our system involves the user throughout the labeling process, using active learning to intelligently explore the space of similar words. In particular, our system can take advantage of negative examples provided by the user. Our system combines multiple preexisting sources of similarity data (a standard thesaurus, WordNet, contextual similarity), enabling it to capture many types of similarity groups ("synonyms of crash," "types of car," etc.). We evaluate on a hand-labeled evaluation set; our system improves over a strong baseline by 36%.

1 Introduction

Set expansion is a well-studied NLP problem where a machine-learning algorithm is given a fixed set of *seed words* and asked to find additional members of the implied set. For example, given the seed set {"elephant," "horse," "bat"}, the algorithm is expected to return other mammals. Past work, e.g. (Roark & Charniak, 1998; Ghahramani & Heller, 2005; Wang & Cohen, 2007; Pantel et al., 2009), generally focuses on semi-automatic acquisition of the remaining members of the set by mining large amounts of unlabeled data.

State-of-the-art set expansion systems work well for well-defined sets of nouns, e.g. "US Presidents," particularly when given a large seed set. Set expansions is more difficult with fewer seed words and for other kinds of sets. The seed words may have multiple senses and the user may have in mind a variety of attributes that the answer must match. For example, suppose the seed word is "jaguar". First, there is sense ambiguity; we could be referring to either a "large cat" or a "car." Beyond this, we might have in mind various more (or less) specific groups: "Mexican animals," "predators," "luxury cars," "British cars," etc.

We propose a system which addresses several shortcomings of many set expansion systems. First, these systems can be difficult to use. As explored by Vyas et al. (2009), non-expert users produce seed sets that lead to poor quality expansions, for a variety of reasons including ambiguity and lack of coverage. Even for expert users, constructing seed sets can be a laborious and time-consuming process. Second, most set expansion systems do not use negative examples, which can be very useful for weeding out other bad answers. Third, many set expansion systems concentrate on noun classes such as "US Presidents" and are not effective or do not apply to other kinds of sets.

Our system works as follows. The user initially thinks of at least one seed word belonging to the desired set. One at a time, the system presents candidate words to the user and asks whether the candidate fits the concept. The user's answer is fed back into the system, which takes into account this new information and presents a new candidate to the user. This continues until the user is satisfied with the compiled list of "Yes" answers. Our system uses both positive and negative examples to guide the search, allowing it to recover from initially poor seed words. By using multiple sources of similarity data, our system captures a variety of kinds of similarity. Our system replaces the potentially difficult problem of thinking of many seed words with the easier task of answering yes/no questions. The downside is a possibly increased amount of user interaction (although standard set expansion requires a non-trivial amount of user interaction to build the seed set).

There are many practical uses for such a system. Building a better, more comprehensive thesaurus/gazetteer is one obvious application. Another application is in high-precision query expansion, where a human manually builds a list of expansion terms. Suppose we are looking for pages discussing "public safety." Then synonyms (or near-synonyms) of "safety" would be useful (e.g. "security") but also non-synonyms such as "precautions" or "prevention" are also likely to return good results. In this case, the concept we are interested in is "Words which imply that safety is being discussed." Another interesting direction not pursued in this paper is using our system as part of a more-traditional set expansion system to build seed sets more quickly.

2 Set Expansion

As input, we are provided with a small set of seed words $\mathbf{s}$. The desired output is a target set of words G, consisting of all words that fit the desired concept. A particular seed set $\mathbf{s}$ can belong to many possible goal sets G, so additional information may be required to do well.

Previous work tries to do as much as possible using only $\mathbf{s}$. Typically $\mathbf{s}$ is assumed to contain at least 2 words and often many more. Pantel et al. (2009) discusses the issue of seed set size in detail, concluding that 5-20 seed words are often required for good performance.

There are several problems with the fixed seed set approach. It is not always easy to think of even a single additional seed word (e.g., the user is trying to find "German automakers" and can only think of "Volkswagen"). Even if the user can think of additional seed words, time and effort might be saved by using active learning to find good suggestions. Also, as Vyas et al. (2009) show, non-expert users often produce poor-quality seed sets.

3 Active Learning System

Any system for this task relies on information about similarity between words. Our system takes as input a rectangular matrix M. Each column corresponds to a particular word. Each row corresponds to a unique *dimension of similarity*; the j^{th} entry in row i m_{ij} is a number between 0 and 1 indicating the degree to which w_j belongs to the i^{th} similarity group. Possible similarity dimensions include "How similar is word w_j to the verb jump?" "Is w_j a type of cat?" and "Are the words which appear in the context of w_j similar to those that appear in the context of boat?" Each row r_i of M is labeled with a word l_i. This may follow intuitively from the similarity axis (e.g., "jump," "cat," and "boat", respectively), or it can be generated automatically (e.g. the word w_j with the highest membership m_{ij}).

Let θ be a vector of weights, one per row, which correspond to how well each row aligns with the goal set G. Thus, θ_i should be large and positive if row i has large entries for positive but not negative examples; and it should be large and negative if row i has large entries for negative but not positive examples. Suppose that we have already chosen an appropriate weight vector θ. We wish to rank all possible words (i.e., the columns of M) so that the most promising word gets the highest score. A natural way to generate a score z_j for column j is to take the dot product of θ with column j, $z_j = \sum_i \theta_i m_{ij}$. This rewards word w_j for having high membership in rows with positive θ, and low membership in rows with negative θ.

Our system uses a "batch" approach to active learning. At iteration i, it chooses a new θ based on all data labeled so far (for the 1^{st} iteration, this data consists of the seed set $\mathbf{s}$). It then chooses the column (word) with the highest score (among words not yet labeled) as the candidate word w_i. The user answers "Yes" or "No," indicating whether or not w_i belongs to G. w_i is added to the positive set $\mathbf{p}$ or the negative set $\mathbf{n}$ based on the user's answer. Thus, we have a labeled data set that grows from iteration to iteration as the user labels each candidate word. Unlike set expansion, this procedure generates (and uses) both positive and negative examples.

We explore two options for choosing θ. Recall that each row i is associated with a label l_i. The first method is to set $\theta_i = 1$ if $l_i \in \mathbf{p}$ (that is, the set of positively labeled words includes label l_i), $\theta_i = -1$ if $l_i \in \mathbf{n}$, and $\theta_i = 0$ otherwise. We refer to this method as "Untrained", although it is still *adaptive* — it takes into account the labeled examples the user has provided so far.

The second method uses a standard machine learning algorithm, logistic regression. As before, the final ranking over words is based on the score z_j. However, z_j is passed through the logistic function to produce a score between 0 and 1, $z'_j = \frac{1}{1+e^{-z_j}}$. We can interpret this score as the probability that w_j is a positive example, $P_\theta(Y|w_j)$. This leads to the objective function

$$L(\theta) = log(\prod_{w_j \in \mathbf{p}} P_\theta(Y|w_j) \prod_{w_j \in \mathbf{n}} (1-P_\theta(Y|w_j))).$$

This objective is convex and can be optimized using standard methods such as L-BFGS (Liu & Nocedal, 1989). Following standard practice we add an L_2 regularization term $-\frac{\theta^T\theta}{2\sigma^2}$ to the objective. This method does not use the row labels l_i.

Data	Word	Similar words
Moby	arrive	accomplish, achieve, achieve success, advance, appear, approach, arrive at, arrive in, attain,...
WordNet	factory	(plant,-1.9);(arsenal,-2.8);(mill,-2.9);(sweatshop,-4.1);(refinery,-4.2);(winery,-4.5);...
DistSim	watch	(jewerly,.137),(wristwatch,.115),(shoe,0.09),(appliance,0.09),(household appliance,0.089),...

Table 1: Examples of unprocessed similarity entries from each data source.

4 Data Sources

We consider three similarity data sources: the Moby thesaurus[1], WordNet (Fellbaum, 1998), and distributional similarity based on a large corpus of text (Lin, 1998). Table 1 shows similarity lists from each. These sources capture different kinds of similarity information, which increases the representational power of our system. For all sources, the similarity of a word with itself is set to 1.0.

It is worth noting that our system is not strictly limited to choosing from pre-existing groups. For example, if we have a list of luxury items, and another list of cars, our system can learn weights so that it prefers items in the intersection, luxury cars.

Moby thesaurus consists of a list of word-based thesaurus entries. Each word w_i has a list of similar words sim^i_j. Moby has a total of about 2.5 million related word pairs. Unlike some other thesauri (such as WordNet and thesaurus.com), entries are not broken down by word sense.

In the raw format, the similarity relation is not symmetric; for example, there are many words that occur only in similarity lists but do not have their own entries. We augmented the thesaurus to make it symmetric: if "dog" is in the similarity entry for "cat," we add "cat" to the similarity entry for "dog" (creating an entry for "dog" if it does not exist yet). We then have a row i for every similarity entry in the augmented thesaurus; m_{ij} is 1 if w_j appears in the similarity list of w_i, and 0 otherwise. The label l_i of row i is simply word w_i. Unlike some other thesauri (including WordNet and thesaurus.com), the entries are not broken down by word sense or part of speech. For polysemic words, there will be a mix of the words similar to each sense and part of speech.

WordNet is a well-known dictionary/thesaurus/ ontology often used in NLP applications. It consists of a large number of synsets; a synset is a set of one or more similar word senses. The synsets are then connected with hypernym/hyponym links, which represent IS-A relationships. We focused on measuring similarity in WordNet using the hypernym hierarchy.[2]. There are many methods for converting this hierarchy into a similarity score; we chose to use the Jiang-Conrath distance (Jiang & Conrath, 1997) because it tends to be more robust to the exact structure of WordNet. The number of types of similarity in WordNet tends to be less than that captured by Moby, because synsets in WordNet are (usually) only allowed to have a single parent. For example, "murder" is classified as a type of killing, but not as a type of crime.

The Jiang-Conrath distance gives scores for pairs of word senses, not pairs of words. We handle this by adding one row for every word sense with the right part of speech (rather than for every word); each row measures the similarity of every word to a particular word sense. The label of each row is the (undisambiguated) word; multiple rows can have the same label. For the columns, we do need to collapse the word senses into words; for each word, we take a maximum across all of its senses. For example, to determine how similar (the only sense of) "factory" is to the word "plant," we compute the similarity of "factory" to the "industrial plant" sense of "plant" and to the "living thing" sense of "plant" and take the higher of the two (in this case, the former).

The Jiang-Conrath distance is a number between $-\infty$ and 0. By examination, we determined that scores below -12.0 indicate virtually no similarity. We cut off scores below this point and linearly mapped each score x to the range 0 to 1, yielding a final similarity of $\frac{min(0,x+12)}{12}$. This greatly sparsified the similarity matrix M.

Distributional similarity. We used Dekang Lin's dependency-based thesaurus, available at `www.cs.ualberta.ca/~lindek/downloads.htm`. This resource groups words based on the words they co-occur with in normal text. The words most similar to "cat" are "dog," "animal," and "monkey," presumably because they all "eat," "walk," etc. Like Moby, similarity entries are not divided by word sense; usually, only the dominant sense of each word is represented. This type of similarity is considerably different from the other two types, tending to focus less on minor details and more on broad patterns.

Each similarity entry corresponds to a single

[1]Available at `icon.shef.ac.uk/Moby/`.

[2]A useful similarity metric we did not explore in this paper is similarity between WordNet dictionary definitions

word w_i and is a list of *scored* similar words sim_j^i. The scores vary between 0 and 1, but usually the highest-scored word in a similarity list gets a score of no more than 0.3. To calibrate these scores with the previous two types, we divided all scores by the score of the highest-scored word in that list. Since each row is normalized individually, the similarity matrix M is not symmetric. Also, there are separate similarity lists for each of nouns, verbs, and modifiers; we only used the lists matching the seed word's part of speech.

5 Experimental Setup

Given a seed set $\mathbf{s}$ and a complete target set G, it is easy to evaluate our system; we say "Yes" to anything in G, "No" to everything else, and see how many of the candidate words are in G. However, building a complete gold-standard G is in practice prohibitively difficult; instead, we are only capable of saying whether or not a word belongs to G when presented with that word.

To evaluate a particular active learning algorithm, we can just run the algorithm manually, and see how many candidate words we say "Yes" to (note that this will not give us an accurate estimate of the recall of our algorithm). Evaluating several different algorithms for the same $\mathbf{s}$ and G is more difficult. We could run each algorithm separately, but there are several problems with this approach. First, we might unconsciously (or consciously) bias the results in favor of our preferred algorithms. Second, it would be fairly difficult to be consistent across multiple runs. Third, it would be inefficient, since we would label the same words multiple times for different algorithms.

We solved this problem by building a labeling system which runs all algorithms that we wish to test in parallel. At each step, we pick a random algorithm and either present its current candidate to the user or, if that candidate has already been labeled, we supply that algorithm with the given answer. We do NOT ever give an algorithm a labeled training example unless it actually asks for it – this guarantees that the combined system is equivalent to running each algorithm separately. This procedure has the property that the user cannot tell which algorithms presented which words.

To evaluate the relative contribution of active learning, we consider a version of our system where active learning is disabled. Instead of retraining the system every iteration, we train it once on the seed set $\mathbf{s}$ and keep the weight vector θ fixed from iteration to iteration.

We evaluated our algorithms along three axes. First, the method for choosing θ: Untrained and Logistic (U and L). Second, the data sources used: each source separately (M for Moby, W for WordNet, D for distributional similarity), and all three in combination (MWD). Third, whether active learning is used (+/-). Thus, logistic regression using Moby and no active learning is L(M,-). For logistic regression, we set the regularization penalty σ^2 to 1, based on qualitative analysis during development (before seeing the test data).

We also compared the performance of our algorithms to the popular online thesaurus `http://thesaurus.com`. The entries in this thesaurus are similar to Moby, except that each word may have multiple sense-disambiguated entries. For each seed word w, we downloaded the page for w and extracted a set of synonyms entries for that word. To compare fairly with our algorithms, we propose a word-by-word method for exploring the thesaurus, intended to model a user scanning the thesaurus. This method checks the first 3 words from each entry; if none of these are labeled "Yes," it moves on to the next entry. We omit details for lack of space.

6 Experimental Results

We designed a test set containing different types of similarity. Table 2 shows each category, with examples of specific similarity queries. For each type, we tested on five different queries. For each query, the first author built the seed set by writing down the first three words that came to mind. For most queries this was easy. However, for the similarity type Hard Synonyms, coming up with more than one seed word was considerably more difficult. To build seed sets for these queries, we ran our evaluation system using a single seed word and took the first two positive candidates; this ensured that we were not biasing our seed set in favor of a particular algorithm or data set.

For each query, we ran our evaluation system until each algorithm had suggested 25 candidate words, for a total of 625 labeled words per algorithm. We measured performance using mean average precision (MAP), which corresponds to area under the precision-recall curve. It gives an overall assessment across different stopping points.

Table 3 shows results for an informative subset of the tested algorithms. There are many conclusions we can draw. Thesaurus.Com performs poorly overall; our best system, L(MWD,+), outscores it by 164%. The next group of al-

Category Name	Example Similarity Queries
Simple Groups (SG)	car brands, countries, mammals, crimes
Complex Groups (CG)	luxury car brands, sub-Saharan countries
Synonyms (Syn)	syn of {scandal, helicopter, arrogant, slay}
Hard Synonyms (HS)	syn of {(stock-market) crash, (legal) maneuver}
Meronym/Material (M)	parts of a car, things made of wood

Table 2: Categories and examples

Algorithm	MAP
Thesaurus.Com	.122
U(M,-)	.176
U(W,-)	.182
U(D,-)	.211
L(D,-)	.236
L(D,+)	.288
U(MWD,-)	.233
U(MWD,+)	.271
L(MWD,-)	.286
L(MWD,+)	.322

Table 3: Comparison of algorithms

	SG	CG	Syn	HS	M
Thesaurus.Com	.041	.060	.275	.173	.060
L(D,+)	.377	.344	.211	.329	.177
L(M,-)	.102	.118	.393	.279	.119
U(W,+)	.097	.136	.296	.277	.165
U(MWD,+)	.194	.153	.438	.357	.213
L(MWD,-)	.344	.207	.360	.345	.173
L(MWD,+)	.366	.335	.379	.372	.158

Table 4: Results by category

gorithms, U(*,-), add together the similarity entries of the seed words for a particular similarity source. The best of these uses distributional similarity; L(MWD,+) outscores it by 53%. Combining all similarity types, U(MWD,-) improves by 10% over U(D,-). L(MWD,+) improves over the best single-source, L(D,+), by a similar margin.

Using logistic regression instead of the untrained weights significantly improves performance. For example, L(MWD,+) outscores U(MWD,+) by 19%. Using active learning also significantly improves performance: L(MWD,+) outscores L(MWD,-) by 13%. This shows that active learning is useful even when a reasonable amount of initial information is available (three seed words for each test case). The gains from logistic regression and active learning are cumulative; L(MWD,+) outscores U(MWD,-) by 38%.

Finally, our best system, L(MWD,+) improves over L(D,-), the best system using a single data source and no active learning, by 36%. We consider L(D,-) to be a strong baseline; this comparison demonstrates the usefulness of the main contributions of this paper, the use of multiple data sources and active learning. L(D,-) is still fairly sophisticated, since it combines information from the similarity entries for different words.

Table 4 shows the breakdown of results by category. For this chart, we chose the best setting for each similarity type. Broadly speaking, the thesauri work reasonably well for synonyms, but poorly for groups. Meronyms were difficult across the board. Neither logistic regression nor active learning always improved performance, but L(MWD,+) performs near the top for every category. The complex groups category is particularly interesting, because achieving high performance on this category required using both logistic regression and active learning. This makes sense since negative evidence is particularly important for this category.

7 Discussion and Related Work

The biggest difference between our system and previous work is the use of active learning, especially in allowing the use of negative examples. Most previous set expansion systems use bootstrapping from a small set of positive examples. Recently, the use of negative examples for set expansion was proposed by Vyas and Pantel (2009), although in a different way. First, set expansion is run as normal using a fixed seed set. Then, human annotators label a small number of negative examples from the returned results, which are used to weed out other bad answers. Our method incorporates negative examples at an earlier stage. Also, we use a logistic regression model to robustly incorporate negative information, rather than deterministically ruling out words and features.

Our system is limited by our data sources. Suppose we want actors who appeared in Star Wars. If we only know that Harrison Ford and Mark Hamill are actors, we have little to go on. There has been a large amount of work on other sources of word-similarity. Hughes and Ramage (2007) use random walks over WordNet, incorporating information such as meronymy and dictionary glosses. Snow et al. (2006) extract hypernyms from free text. Wang and Cohen (2007) exploit web-page structure, while Pasca and Durme (2008) examine query logs. We expect that adding these types of data would significantly improve our system.

References

Fellbaum, C. (Ed.). (1998). *Wordnet: An electronic lexical database*. MIT Press.

Ghahramani, Z., & Heller, K. (2005). Bayesian sets. *Advances in Neural Information Processing Systems (NIPS)*.

Hughes, T., & Ramage, D. (2007). Lexical semantic relatedness with random graph walks. *EMNLP-CoNLL*.

Jiang, J., & Conrath, D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *Proceedings of International Conference on Research in Computational Linguistics*.

Lin, D. (1998). An information-theoretic definition of similarity. *Proceedings of ICML*.

Liu, D. C., & Nocedal, J. (1989). On the limited memory method for large scale optimization. *Mathematical Programming B*.

Pantel, P., Crestan, E., Borkovsky, A., Popescu, A., & Vyas, V. (2009). Web-scale distributional similarity and entity set expansion. *EMNLP*.

Pasca, M., & Durme, B. V. (2008). Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. *ACL*.

Roark, B., & Charniak, E. (1998). Noun-phrase co-occurrence statistics for semiautomatic semantic lexicon construction. *ACL-COLING*.

Snow, R., Jurafsky, D., & Ng, A. (2006). Semantic taxonomy induction from heterogenous evidence. *ACL*.

Vyas, V., & Pantel, P. (2009). Semi-automatic entity set refinement. *NAACL/HLT*.

Vyas, V., Pantel, P., & Crestan, E. (2009). Helping editors choose better seed sets for entity expansion. *CIKM*.

Wang, R., & Cohen, W. (2007). Language-independent set expansion of named entities using the web. *Seventh IEEE International Conference on Data Mining*.