Learning Paraphrase Models from Google New Headlines

Dave Kale
(with thanks to Teg Grenager and Bill MacCartney)
CS 224N Final Project, Spring 2007
davekale@stanford.edu

Abstract

Data sources like the clusters of news headlines at Google News present an exciting opportunity to learn paraphrase models from data automatically. We present both a novel dataset and a novel approach to automatic, unsupervised learning of paraphrase models from that datset. Leveraging existing NLP tools such as the Stanford Parser and lexical resources such as WordNet and Infomap, we constructed a system that first aligns the typed dependency graphs of large numbers of parallel headlines (on the order of hundreds) and then uses aligned paths between corresponding nodes as candidates to a paraphrase extraction system. We present some preliminary results in the form of actual learned paraphrase models. This project serves as a proof of concept for this approach and sheds some light on likely next steps.

1 Introduction

Paraphrases can prove especially troublesome to the task of *recognizing textual entailment* (RTE). Two multiword phrases that possess virtually the same meaning can vary in terms of length, syntactic units and structure, individual words, and more. Many textual entailment systems include an alignment step during which corresponding words and structures are matched up or nodes and edges in *typed dependency graphs* are aligned (MacCartney et al., 2006). Such multiword expressions are particularly challenging because they span multiple nodes and edges and can greatly effect the performance of an alignment algorithm. In many RTE systems this bad performance is tolerated and compensated for in later steps. In other cases RTE systems attempt to collapse common multiword expressions into single nodes; this is done predominantly for named entities, compound nouns, and prepositional phrases. Nevertheless, no RTE systems have presented a comprehensively robust and broad method for handling paraphrasing.

One approach is the creation and use of a phrasal resource similar in practice to many more common lexical resources (WordNet, InfoMap, etc.). Perhaps the most famous is the *Discovery of Inference Rules from Text* (DIRT) database first presented in (Lin and Pantel, 2001), which was constructed automatically and is used most commonly for question answering. (Lin and Pantel, 2001) extracted their paraphrases by matching up typed dependency tree paths with high distributional similarity (i.e., statistically similar contexts). Other work has made similar use of distributional similarity and contexts (Hasegawa et al., 2004). More recent work has leveraged the common existence of Named Entities (NEs) in texts like newspaper articles, detecting, clustering, and utilizing them as anchors for paraphrase candidates (Shinyama and Sekine, 2003). Such systems extract paraphrases from phrases or contexts involving NEs almost exclusively. Both approaches deal with the task of finding (or creating) an appropriate dataset, which is time-consuming and challenging. Paraphrases, by definition, express some sort of common semantic content and so paraphrases should be extracted from passages with similar, if not identical, meaning; however, identifying and corresponding such passages requires some strategy for recognizing this kind of entailment or relation. While both the approaches above find ways around this problem, they do so by trading off the scope and generality of the paraphrases they are able to extract (Sekine, 2005).

Thus, paraphrase modeling, particularly when done automatically on unlabeled texts, is an open problem and remains quite challenging. We present an alternative approach inspired by the availabil-

ity of a novel data source and recent advances in the area of textual entailment and graph alignment. The implemented system acts as proof of concept for both the dataset and the approach, demonstrates promising results, and opens the door to promising future work.

1.1 Google News Dataset

The main problem with the task of learning paraphrase models from data automatically is the very nature of the data itself. By nature the selection and preparation of appropriate data "begs the question." To discover and extract paraphrases requires the correspondence of semantically similar passages and phrases; however, detection of semantically similar passages requires some understanding of notions of paraphrase and entailment. Many systems get around this question with a variety of approximations (like bag-of-words keyword matching, distributional similarity, restricting data to named entities, etc.), but this places restrictions on the eventual results. Another possibility would be to compile and annotate a large dataset by hand (rather, by the hands of unpaid undergraduate research assistants), but this is tedious and annoying.

The strategy we have adopted is to allow large corporations with nearly unlimited resources and generally public products to do this on our behalf. We are, of course, referring to Google and, in particular, to its Google News service. Google News (in its own words) is "a computer-generated news site that aggregates headlines from more than 4,500 English-language news sources worldwide, groups similar stories together and displays them according to each reader's personalized interests" (Google News website, 2007). They cluster articles based not only on text-based features of the headlines and articles themselves but also on characteristics of their respective publications, publication time, web statistics, etc. The end result is a website that is updated more than once per day and that at any one time has article headlines numbering in the hundreds of thousands. The articles are partitioned into one of seven topics (World, U.S., Business, Sci/Tech, Sports, Entertainment, and Health) and, within each topic, then more precisely assigned to clusters of very similar articles. Within each topic, the top 20 most salient (according to some Google measure) clusters are displayed, each containing between 300 and 1500 articles and headlines.

The end result is a large, publicly available database of news headline clusters. Within each cluster the headlines similarity is consistent and somewhat uncanny – syntactic structure, word choice, and style vary but it is clear that the large majority of headlines refer to the same event and contain very similar semantic content. These clusters are a fertile field for the harvesting of large classes of paraphrases grounded in a variety of topics. Headlines are short and concise, so that procedures can be designed to handle many at once.

This dataset has drawbacks certainly. Using a dataset that is strictly headlines-based (not to mention news based) will bias and restrict the set of paraphrases we can learn. Also, headlines tend to have funky grammars, would could affect the quality and usefulness of any paraphrases we extract. Also, online headlines tend to be messy (filled with inconsistent capitalization, misspellings, anomalous syntax and punctuation, HTML tags, etc.). Nevertheless, the benefits of the dataset far outweigh its drawbacks, and furthermore, we think procedures effectively applied to this dataset can be bootstrapped to work on other more general datasets.

To date we have downloaded and prepared nearly 13,000 clusters of headlines, and this number grows on a daily basis, creating a (hopefully) richer and more diverse dataset. **Figure 1** an example of some headlines from a single cluster.

1.2 Related work

There is a great deal of related work taking place in the Stanford NLP group here at Stanford. First of all, we leveraged a number of JavaNLP tools, in particular the Stanford Parser and the large number of electronic lexical resources. Of great relevance is the work the group has done for both the RTE2 and RTE3 Challenges (MacCartney et al., 2006; de Marneffe et al., 2007; Chambers et al., 2007). In particular the algorithm we use to align typed dependency parses is based on the one used most recently in the RTE3 Challenge for aligning dependency parses of the text and hypothesis sentences. We also borrow shamelessly from a number of ideas that have been bouncing around. Finally, Teg Grenager conceived of this project in its orig-

```
TSA eases carry-on restrictions.
US to Ease Ban on Liquids, Gels on Airplanes.
US eases ban on air travelers carrying liquids.
US relaxes travel restrictions on liquids.
Feds adjust travel restrictions.
US Set to Relax Ban on Liquids on Airliners.
Air travel restrictions eased in the US.
Liquids Ban on Airlines is Not as Strict Now.
US to allow some liquids on airliners.
Ban on liquids modified.
```

Figure 1: Examples of Google News headlines cluster.

inal form and contributed heavily to its framework and early-stage code, as did Bill MacCartney.

2 Brief System Overview

We will give a brief overview of the complete paraphrase extraction system before talking in greater detail about the alignment stage.

2.1 Preprocessing

The Google News headlines data required a number of important preprocessing steps before it was ready to be passed to the Stanford Parser. These preprocessing steps include (but are not limited to):

- *Extracting actual headlines from text of webpage*
- *Cleaning up HTML tags and other junk*
- *Fixing inconsistent capitalization* (with the goal of leaving only words at the beginning of sentences and NEs capitalized)
- *Removing duplicate headlines from within each cluster*

In the future this step could also include running the headlines through a NER or part-of-speech (POS) tagger to aid parsing.

2.2 Parsing

We run all headlines of interest through the Stanford Parser to create lexicalized phrase-structure parses, which are then converted via rules into typed dependency (or semantic) graphs. We found that the syntax of headlines differed enough from that of the Wall Street Journal (WSJ) treebank used to train the Stanford Parser that we needed to modify the training of the parser subtly to properly handle those nuances. Therefore, we hand labeled 40 representative headlines, added them to the training set (along with the WSJ trees), and trained a new parser that work significantly better. Most of the problems the parser has now regard improper handling of things like NEs.

2.3 Alignment

The goal of the alignment stage is to take the typed dependency graphs for a single headlines cluster and align them all to one another. Equivalently, we align them to a global graph structure (called ambiguously a *supergraph*) that underlies all of the headline graphs. This is done by assigning IDs to all of the supergraph's nodes (called *supernodes*) and then using those IDs to associate nodes in the typed dependency graphs to the supernodes. When two nodes in a typed dependency graph are assigned, the path between them becomes a path between their respective supernodes. Once the alignment of all graphs to the global graph is completed, all of the paths between two supernodes become candidates from which to create a paraphrase model.

2.4 Paraphrase Extraction

The final stage, to be glossed over here, concerns the process of taking a bunch of dependency graphs and the underlying global graph to which they are aligned and then extracting decent paraphrases from them. As mentioned above, the set of paths between

two supernodes becomes a candidate for a paraphrase model. We do a variety of postprocessing, including filtering extremely short or trivial paths (e.g., paths of length 0 are simply synonyms, paths of length 1 are often basic grammar rules or dependency relationships imposed by the parsers) and merging overlapping path sets. Paraphrase models are then created by replacing some supernodes along a set of paths with variables (or *slots* in the DIRT sense), creating a template of sorts that specifies a particular dependency relationship between two or more variables, other words or parts of speech, and edges in a dependency tree.

3 Alignment

The focus of this project was the further development of the alignment stage. As noted above, the goal of the alignment stage is to assign global (supernode) IDs to each node in each typed dependency graph (of a headline) in such a way as to obtain the optimal alignment of all headline graphs to one another (or equivalently, to the global graph in a manner that is consistent). The supergraph is a structure that underlies the dependency parses. If we think of this as a generative model, then the supergraph is a set of hidden variables and parameters that generate typed dependency graphs as observations: supernodes spit out nodes that take the form of words while *superedges* between them spit out paths (of one or more edges, possibly including multiple words and dependency relationship) between those nodes. In any case the underlying global graph is unknown and so is estimated (or constructed, depending on how you want to look at it) incrementally as we align each dependency graph to it.

Finding the optimal alignment amounts to a search through the space of alignments. This space is prohibitively large: it has upwards of 50 headlines per cluster, most with 5-10 words to be assigned global IDs, and any number of nodes in the global graph. Thus, some strategy for making the search tractable is necessary, as will be explained below.

Another way to understand the alignment process is as a clustering process in which graph nodes as assigned to clusters (supernodes) according to some scoring metric. However, the number of clusters is unknown at the outset; in other words, we can make no assumptions about the actual size or structure of the hidden supergraph. To address this, as we search for the optimal global ID (alignment) for a particular node, we allow that node to align with some small probability to a previously "empty" supernode; this allows the supergraph's structure to be augmented when appropriate. This allowance for new clusters to be added in a probabilistic way is similar to a Dirichlet process.

3.1 Scoring

Our scoring metric is very similar to that used in the Stanford RTE3 work mentioned earlier. We define the score of an aligned cluster to be the score of the supergraph "built" by the alignments of all the typed dependency graphs. In the same way that (de Marneffe et al., 2007) decompose alignment scores, we do the same, making the score of a supergraph to be defined to be the sum the individual scores of its supernodes and superedges:

$$s(G, \mathcal{G}) = \sum_{i \in G} s_{\text{node}}(G_i) + \sum_{(i,j) \in e(G)} s_{\text{edge}}((i,j))$$

where G is the supergraph, $\mathcal{G}$ is the set of dependency graphs aligned to it, G_i is a supernode (set of dependency graph nodes with ID i). The score of a supernode is defined to be a measure of the "consistency" of all dependency graph nodes (in most cases, words) assigned to it. We define it to be the average pairwise consistency of two nodes in the set:

$$s_{\text{node}}(G_i) = \alpha \sum_{n_1, n_2 \in G_i} c_{\text{node}}(n1, n2)$$

where α is a normalizing constant (equal to 1 over the number of distinct pairs of nodes). The consistency score (c_{node}) of two nodes can be defined in any way we choose; in our most basic implementation we simply used the score returned by WordNet for the word/pos pair. We have also experimented with expanding this to include other lexical resources.

In the first version of our system, we ignored the edge score, letting $s_{\text{edge}}((i,j)) = 0 \; \forall i, j \in e(G)$.

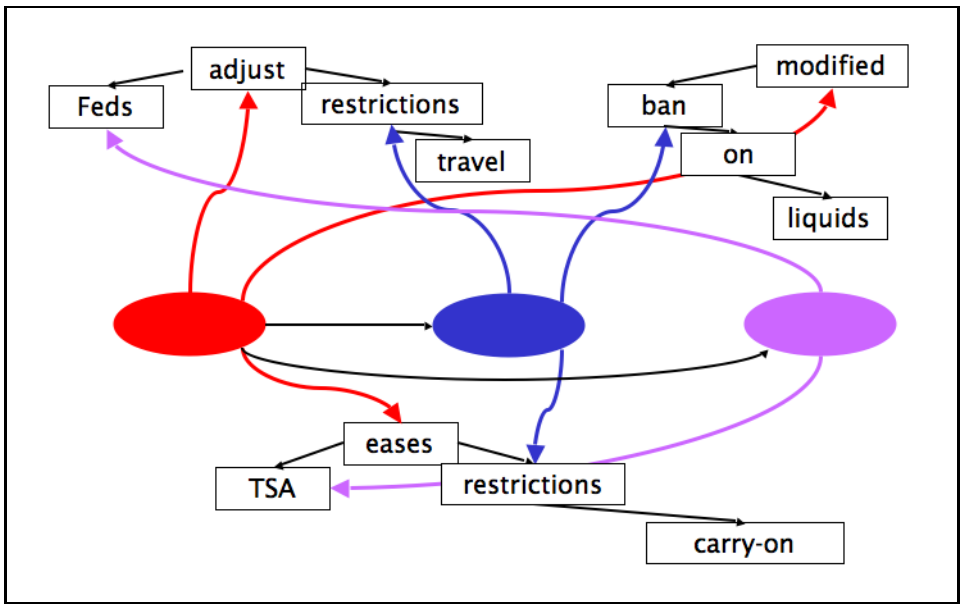

Figure 2: Visualization of three headline dependency graphs aligned to global supernodes.

This is also a good way to initialize the search state. In later versions we did something similar to the node score, letting the score of a superedge be the average pairwise consistency of all dependency paths corresponding to this edge. However, designing an edge score, even in this decomposed fashion, proved to be difficult. It requires developing a way to compare and score dependency relations, among other things. Of particular difficulty is the fact that a superedge can correspond to a multi-edge path in a dependency parse. Thus, our consistency score must be able to compare a single dependency relation (one edge) to path passing through multiple dependency relations and one or more nodes. To date we chose to do the simplest thing, using hand-designed scoring functions in much the same way as (de Marneffe et al., 2006). We designed a consistency function that penalized large differences in length and preferred "similar" dependency relations. The performance of this approach was mediocre at best; a better, more principled approach is needed, possibly one that uses a variety of weighted features and perhaps is learned from data.

3.2 Local search algorithm

Due to the size of the search space, an exhaustive search is extremely infeasible. Motivated by the Stanford RTE team's work, we chose to address this with a local search in which state space is a complete assignment of global IDs to all dependency graph nodes and a transition between states is change to only one node's assignment. In each step of the search, we select one node examine all possible assignments to its global ID, generating a score (based on the resulting change to all affected consistency scores) for each. At this point we treat the scores as unnormalized log probabilities, normalize them, and create a valid posterior probability distribution from which we sample the next state. As noted in (de Marneffe et al., 2007), this is a form of *Gibbs sampling*. For most transitions we constrain the next state to be a high probability one but allow the search to take low probability states on occasion, a form of randomized annealing designed to help avoid local optima. We iterate through the nodes in all dependency graphs in a semi-fixed order, changing one at a time per transition. We run this algorithm for a large number of iterations until some convergence criterion is met.

4 Experiments and Evaluation

Our basic experimental setup was to parse a large number of headline clusters and then run them all through the entire system, aligning one cluster at a time, extracting paraphrase models from each, and

then attempting to merge classes of paraphrases that are very similar. The end result are paraphrase models (resembling templates) that are very similar to those produced by DIRT; they resemble pieces of dependency graphs, defining typed relations between a number of nodes, some of which are variables (or slots) to be filled by multiple possible words. Paraphrase models are grouped into equivalence classes, in which all members of a class (purportedly) represent paraphrases of one another.

```
X <nsubj hit >dobj Y
X <prep_by hit >nsubj Y

X <nsubj strike >dobj Y
X <prep_by hit >nsubj Y

X <dobj rejoin <xcomp Y
X <prep_to return <xcomp Y

X <nsubj get >dobj
              year >num Y
X >dep sentence >prep_to
              year >num Y
```

Figure 3: Examples of learned paraphrase models.

At this time we have no quantitative way to evaluate our system's performance so we resort to human evaluation of the paraphrases discovered by the system. Some representative example pairs are in **Figure 3**. Some examples are compelling, but this is tempered by the fact that the system returns many more questionable paraphrases as well. We list here a number of problems or issues and possible ways to address them in the future:

- *Redundant equivalence classes:* A number of equivalence classes are redundant, which we attribute to redundant headline clusters. This is an artifact of the way we download clusters (download all clusters present on a daily basis) and the nature of news - big stories (and even identical articles) persist in the headlines for days and sometimes weeks, generating Google News clusters every day. We are considering ways to deal with the redundancy, but it poses more of a nuisance rather than an actual problem.

- *Named Entities creating bad parses:* In doing the simplest thing first, we decided to forego doing any sort of tagging before parsing. However, we found that headlines are saturated heavily with NEs, and the Stanford Parser didn't always label or parse them correctly. This problem was compounded by the presence of multiword NEs and words with improper capitalization. Nearly all of these problems would likely be addressed by running our headlines through a standard NER system before parsing.

- *Bad or wrong alignments:* We are confident in the search algorithm we chose for the alignment stage, so we suspect that bad alignment examples result from the scoring function we designed. The node-based scoring function works OK and probably just requires greater thought and care as to which lexical resources are used and how to most effectively combine them. However, the edge-based scoring requires a great deal of additional work, both in design and in implementation. We think that a more fine-grained feature-based approach would yield superior performance. We also want to explore the possibility of learning a scoring function from data, as in (de Marneffe et al., 2007; Chambers et al., 2007).

- *Better evaluation needed:* Obviously we need a method of evaluation better than "inspection." The most straightforward thing to do would be to take a number of clusters and enumerate as completely as possible all of the paraphrases models we can find and then do a precision/recall performance measure on the coverage our system gets for these clusters. However, this is trickier than it seems as it is not a truly comprehensive measurement of our system's performance.

5 Conclusion

This project demonstrates a number of important contributions. First, we (with the help of Teg Grenager) aided in the discovery, harnessing, and exploitation of a novel, publicly available dataset, the Google News headline clusters dataset. Our project has demonstrated this dataset's promise of usefulness and fertility. Our second contribution is an effective extension to an established Gibbs-sampling-based alignment algorithm; our extension applies the algorithm to the alignment of an arbitrarily large set of dependency graphs in an effective, flexible way. We have done some promising preliminary work on meaningful feature selection and scoring function design. Furthermore, our approach has interesting theoretical pinnings related to generative models and Dirichlet processes. Finally, we demonstrate promising preliminary results and clearly enumerated several avenues for strong improvement.

References

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D. Manning. *Learning Alignments and Leveraging Natural Logic*. 2007. ACL Workshop on Textual Entailment and Paraphrase, 2007.

Marie-Catherine de Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D. Manning. *Learning to distinguish valid textual entailments*. 2006. Second Pascal RTE Challenge Workshop.

Marie-Catherine de Marneffe, Trond Grenager, Bill MacCartney, Daniel Cer, Daniel Ramage, Chloé Kiddon, and Christopher D. Manning. *Aligning semantic graphs for textual inference and machine reading*. 2007. AAAI Spring Symposium at Stanford 2007.

Google News. *http://news.google.com*. ©2007 Google.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. *Discovering relations among named entities from large corpora*. 2004. Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL04).

Dekang Lin and Patrick Pantel. *DIRT - Discovery of inference rules from text*. 2001. Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. *Learning to recognize features of valid textual entailments*. 2006. Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2006).

Satoshi Sekine. *Automatic paraphrase discovery based on context and keywords between ne pairs*. 2005. Proceedings of 3rd International Workshop on Paraphrasing (IWP2005).

Yusuke Shinyama and Satoshi Sekine. *Paraphrase acquisition for information extraction*. 2003. Proceedings of 1st International Workshop on Paraphrasing (IWP2003).