Context Sensitive Paraphrasing

Michael Connor and Dan Roth
Department of Computer Science
University of Illinois at Urbana-Champaign
connor2@uiuc.edu and danr@cs.uiuc.edu

Abstract

Given a sentence and a specific word or phrase in that sentence, is it possible to replace that word or phrase with another and have the sentence keep the same meaning? Most paraphrasing work develops patterns and templates that can replace other patterns or templates in some context. But what about a specific given context, how do we know if a paraphrase rule applies? Ideally if one knew the correct sense to every word in the sentence, then one would also know what other words or phrases can replace a given word and have the sentence keep its same or entailed meaning. As it is there is no reliable way to tag the sense of every word, and with current sense repositories it is difficult (and perhaps impossible) to determine what senses can replace another for every word and every domain. We will present a new task and formalism that rests somewhere between these two approaches. Our system will allow us to say if A can replace B in a context sensitive way, even on unseen A and B.

1 Introduction

Say you are given a sentence "The general commanded his troops." Is that the same as saying "The general spoke to his troops?" The goal of a paraphrase system should be to detect or generate such asymmetric pairs. Already many systems can generate rules or templates such as "X commanded Y" can be rewritten as "X spoke to Y". The problem with these rules is how does one know when they can be applied and in what order to truly keep the same meaning. In the sentence "The Soloist commands attention.", is it still true that 'commands' can be replaced with 'speaks to'? Alternatively, on a single word level, one can ask, "could 'speak to' replace 'command' in the original sentence and not change the meaning of the sentence." This is really a word sense disambiguation task: is the meaning of 'command' in the sentence the same as the meaning of 'speak to.' With a perfect sense tagger and perfect sense repository our job would be nearly done.

What this project attempts is to learn when one word can replace another in a specific sentence directly instead of learning an intermediate and possibly more difficult word sense assignment or attempting to compile a list of complex rules for when and if one phrase can replace a second. Our approach lies somewhere in-between a template driven approach and straight word sense disambiguation since it takes individual context into account and learns a somewhat weaker, but hopefully still useful, sense function.

The real impetus for this project is the need for a word replacement component as part of a larger entailment system (Ido Dagan and Magnini, 2005), (de Salvo Braz et al., 2005), (R. Raina and Manning, 2005), (Oren Glickman and Koppel, 2005). In this setup one is being asked specifically can a specific word or phrase in the hypothesis replace a word or phrase in the given sentence and keep the same or entailed meaning as the given sentence. Such paraphrasing requires knowledge of the specific context and whether two words fit and do not disturb meaning.

Previous paraphrase work generally viewed it as a generative task, namely given a sentence or phrase, generate paraphrases of that phrase which have the same or entailed meaning. Often this would be represented as a fixed set of rules. Training these systems could require parallel or comparable corpora (Barzilay and Lee, 2003) which ties the systems to very specific topics. Other systems extract rules from dependency trees built over a large corpora or the web (Glickman and Dagan, 2003),(Idan Szpektor and Coppola, 2004),(Lin and Pantel, 2001). These create more general rules, but they still only say that a context exists where the one phrase can replace another. They do not indicate when a rule can be applied.

Alternatively we can treat the single word replacement paraphrase task as a sense disambiguation task. Knowing the sense of the target word will take us most of the way to determining if another word could replace it. But what if in our given sense repository there is no easy to elicit connection between words or senses? In Wordnet 2.1 (Miller, 1995) there is no connection (other than through some analysis of gloss) between 'leave' and 'withdraw', yet few would argue with "The troops withdrew" being rewritten as "The troops left." It may never be possible to build an ontology that captures all appropriate senses or sense relations.

2 Formal Model

Formally, we are learning a binary function $f(S, v, u)$: Given sentence S, can we replace target word or phrase v in S with u and have S keep the same or entailed meaning. Such a function f can be applied either directly to decide if a word in the second sentence of an entailment example can replace a word in the first sentence, or it can be used as a filter for a list of words to form a set of extended synonyms for the target word v. This task captures the usefulness of word sense disambiguation (being able to tell if two words mean the same in a given context) without having to know or rely on any specific set of senses. It also generalizes paraphrasing rules, where a paraphrasing rule is just an instance of a single positive judgment of the system, while adding context sensitivity in the form of given sentence S.

We tested two implementations of this function f. The first is closer to the word expert approach used in many WSD systems, a separate $f_u(S, v)$ classifier for every u. Since v is given extra information in the form of its context in S, we may need to know something about u, and a separate f_u classifier will implicitly through training learn relevant senses of u. This approach suffers though because the function cannot be applied to any u it was not trained for. We cannot gather training data for all possible words and phrases, so we would need to rely on some baseline for any unseen and untrained words or phrases if we were to use this implementation.

Instead, we may train a single binary classifier for all word pairs v, u and given context S. If we are able to represent some notion of what v and u have in common, and what they jointly or individually have in common with the target sentence S, then all words and more importantly all phrases do not have to be seen during training. All the learner needs to determine is which shared features indicate replacability.

2.1 Features

As a base set of features we used the usual bag of words (in lemmatized form), collocations, and local syntactic structure (from Minipar (Lin, 1993)). These features contain information about the local context of v in S, but do not say anything about u. For the f_u classifiers this may be enough, these classifiers just need to determine which contexts u can fit in, but they will provide little to no help for a global classifier. If a single classifier is applied for all words, it would need to know the contexts each word can fit in, but these features do not provide that.

What we need is a more general representation of similarity between u and v in S. To that end we extend the local features mentioned above to include features specific to the pair u, v. For each word pair we search a large corpora of text to determine all contexts that both words can appear (above some frequency). Instead of relying on a fixed set of senses to specify meaning of words, we use a more fluid notion of context and word to represent meaning of the word. The idea is that perhaps there are shared contexts that are highly indicative that two words share meaning, or more generally one word or phrase can replace the other: contexts like the same named entities as subject and object of target verbs.

With only the features above, the best our learner could hope for is to learn weights for shared contexts that say two words or phrases can replace each other (Rion Snow, 2005). From pairwise features alone a learner cannot determine directionality or context. With the local features such as bag of words and collocations then the f_u classifiers may learn what local contexts represent 'good' contexts that u would fit into, and the pairwise features would say whether u and v are similar, so hopefully enough information is captured. For the single global classifier, it has no chance to identify what are good local contexts for each u, without basically separating into different u classifiers. For the single global classifier idea to work, it needs to modify the set of features so that they represent when the pair of words are similar to each other, especially within the local context.

To this end, instead of adding more features to indicate sense of local context (perhaps as a set of informative words seen with this context in a large corpus), we filtered the contexts shared by u, v to only those that are also similar to S. Our current notion of similarity is all those context features that have been seen with the same words as the current context. In other words for each local context we find the set of words that have been seen at least N times with this context in a large corpora. Another context is considered similar if it is also seen N times with M of these words. The intuition is that if the same words can be placed in two contexts then their meaning may be related. Now the pairwise features of S, u, v are those contexts that may indicate u could replace v (seen in similar contexts), and that the contexts they can replace are similar to the current local context S.

Having classifier and features defined, we still need to train, and for this we need training data. There is no standard corpus for this task, and it relies on possibly subtle human judgments, so examples must be hand tagged for each sentence and word pair. Trivially positive examples are easy to come by in a large corpus (replacing a word with itself), but still shed some light as to what contexts the replacing word u can appear with. With high probability, u cannot replace an unrelated word from a randomly selected sentence, so we can form a pool of likely negative examples. We still need good positive and negative examples to capture the sort of distinctions we are looking for. For this we use active learning where the pool of example v's (and their sentences S) are drawn from a list of similar words to u (extracted from Lin's dependency-based thesaurus(Lin, 1998)). In this way our training procedure is almost a filtering of this set of possible word replacement rules to find contexts where they do and do not apply for a given u. Training for the single pairwise classifier uses the hand tagged examples from all u.

3 Experiments

For our experiments we restricted the class of words and phrases that can be replaced to verbs and phrasal verbs. Disambiguating verbs is a difficult task because of the fine grained and subtle distinctions between sense (Palmer, 2000). Context for the verb in a sentence is defined as the dependency links given by the Minipar dependency parser (Lin, 1993). Minipar is run over sentences that have been tagged with POS and named entities so links to named entities and pronouns have been collapsed and added. So in the sentence "Richard kicked him" the context for 'kicked' is *subj:Richard, subj:NE:PER, obj:him, obj:PRONOUN*. This way words or pairs can be associated with the more general notion of 'appears often with a person as the subject' as well as the more specific 'appears often with Richard as the subject'.

The most direct comparison to this work is that of Glickman and Dagan (2003). They too experimented with paraphrasing individual verbs and verb phrases, based largely on common context as defined by Minipar dependency links. The results of their system though are existential verb pairs: they can only say that there exists a sentence where one of these verbs can replace another.

To compare our approach to theirs we randomly selected 10 pairs that their system had output and that humans had judged as being correct (where correct means some sentence exists where this rule can be applied) (Table 1). Each verb pair was ordered such that there was a reasonable direction (u replace v). For each v, the goal was to have 10 sentences

$v \leftarrow u$
grow ← increase *
derail ← disrupt
offer ← pay
enforce ← police *
head ← move
cut ← trim
tighten ← toughen
appoint ← name
strike ← attack *
join ← support *

Table 1: Selection of verb replacement rules generated by Glickman and Dagan, judged by humans to be correct for some context

either created or extracted from newswire as a test set (100 total sentences). Each sentence for a given v was hand tagged by two humans who determined whether u could replace it and keep the same or entailed meaning. Inter-annotator agreement was 85%, with disputed examples adjudicated by a third annotator. Each test set was then pruned down to 10 sentences such that there would be an equal number of positive and negative examples. This pruning does bias the results since the distribution of senses is most likely not uniform, but this way there is no advantage to always guessing a fixed positive or negative classification.

Using this test set we compare their rules (which would say v can be replaced by corresponding u in all cases) to both separate f_u classifiers trained on each u and single f classifier trained with examples from all u. We use active learning to acquire training data for each f_u classifier. Over this set of examples we train a SNoW based classifier (Roth, 1998), tuned using 10 fold cross validation on the training data. SNoW uses a variation of the winnow update rule(Littlestone, 1988) that has shown to be useful for tasks with a large number of features per example, but relatively few useful features. This is ideal for our situation where there can be over 1000 pairwise features representing contexts shared by two verbs, but only a few may be indicative of similarity. We want to be able to identify those features as fast as possible.

3.1 Implicit classifier

Table 2 shows performance of the individual f_u classifiers as compared to the Glickman and Dagan rules. If used as a simple decision list style classifier (if v is seen, it can be replaced with u), the rules would judge all examples to be positive, it has no notion of context. For this reason, the Glickman results are exactly equivalent to a system that says yes to all of these given examples. The trained classifiers do better than the paraphrase rules overall, but in only a few cases do they do something different or better than making a constant decision for a given pair.

The starred rules in table 1 represent systems that never saw an example to replace v when training a u classifier. These classifiers have to generalize about the shared features of this new v, compared to the v seen in training. For the most part these classifiers seem to do a good job of it. In fact, these f_u classifiers are the only ones that improve over the Glickman rule baseline. It is possible that these classifiers are doing better than those that have been trained with these specific v's because when a f_u classifier gets training data for a target v, its weights may be set such that all examples with this v are fixed at positive or negative. With a classifier f_u such as for *increase*, it must be making decisions on context and shared context alone because it has never seen the target *grow* during training.

3.2 Global Classifier

Individual f_u classifiers should 'know' more about the u they are trained for than a single global classifier can be capable of. This is simply because in training a single global classifier does not see u in the features of the example as much as it sees the features that represent similarity between u, v and S. The word expert classifiers do not need this shared feature representation as much, since it implicitly learns it during training, and as we describe above these features may in fact hurt the f_u. On the other hand, a global classifier needs such a representation of shared features to learn a general similarity measure. After learning this, it can be applied to new unseen pairs (both u and v unseen, not just unseen v). It is assumed that global classifier may not be able to make such fine distinctions that an expert f_u

	# Test examples	Glickman	f_u classifier	Global f
Total	100	50	53	59

Table 2: Classifier performance on 10 examples of each verb replacement rule. f_u classifier uses a different classifier for each rule and the Global f applies a single trained classifier to all examples.

$v \leftarrow u$
create $\leftarrow$ set up
carry out $\leftarrow$ conduct
call for $\leftarrow$ want
take up $\leftarrow$ accept

Table 3: Unseen phrasal verb pairs, generated and judged correct by Glickman and Dagan

classifier can, but it hopefully will capture a more general notion of similarity and paraphrasing.

To test the single f global classifier we trained a single SNoW network over all the training data collected for each f_u. We compared the single classifier to each f_u on the test set in table 2, but we also extend the test set with 4 additional totally unseen rules (table 3). Each of these rules also incorporates simple phrasal verbs, the sort that exist in WordNet and that Minipar can recognize (when contiguous). For these phrases the f_u cannot be applied because such an f_u was not created. It is not clear whether it would ever be meaningful to try to use an f_u classifier on a u it wasn't trained for. Perhaps the function f_u classifiers learn can be applied to some set of similar words outside of just u; perhaps there is a better partitioning of classifiers between single words and a single classifier overall.

Curiously, the global classifier does better overall on the trained u than the individual f_u classifiers, when trained with same data. One reason the global classifier outperforms individual f_u classifiers is simply that it has more training data since it has access to the training data for all the u. To test this an extra 100 examples were tagged with the active learning procedure for 4 u, which doubled the training set size for these classifiers. With the extra training data the accuracy of these classifiers remained constant, so it can't be the case that it was only the extra training data that helped.

The global classifier is using the extra data to in fact not overfit, as opposed to the individual f_u classifiers. The problem the global classifier must learn is much broader than what each f_u is presented with, and the training data more varied. Since global classifier is presented many different pairs it is unable to overfit any one.

Table 4 shows that on this small set of total unseen pairs the global classifier does reasonably, but perhaps to be expected for unseen u and v, its not spectacular. The one pair that the classifier performs well on, *take up* $\leftarrow$ *accept*, is possibly most similar to some of the pairs the global classifier was trained on (notably *offer* $\leftarrow$ *pay*). The results are reasonable though, showing that with even more, and more varied, training data the global classifier can fulfill the promise of generalizing to unseen.

3.3 Active Learning

Since training data was to be tagged by hand, active learning was used to facilitate selecting what were hoped to be good examples. Regularized perceptron was used as the core classifier for the active learning system, an approximation of the support vector machine approach in (Tong and Koller, 2001). New examples were selected from a pool of unlabeled examples and presented to the user to be tagged. This selection was based on the closeness to the current linear hyperplane. 100 examples would be tagged this way, and these examples would be saved and used to train a separate SNoW classifier.

To use this active learning setup, for each u, three pools of initially unlabeled examples were generated. The first pool are a small number (10) of sentences that u actually occurs in. These are trivially positive examples, used to prime the linear classifier. The second pool are sentences with a verb from Lin's similarity list for the given u. The third pool contains sentences with verbs of around the same frequency (in the corpus used to draw examples) as u, but are not related (either through Wordnet hypernym hierarchy or on Lin's list). It is generally safe to assume the pool of random examples are almost all negative, so we use a small random selection of

	# Examples	Glickman	Global f
Total	35	19	22
Accuracy		54.3%	62.9%

Table 4: Global classifier on new phrasal verb pairs

these along with the trivially positive to initialize the linear classifier used for active learning. The rest of the random examples are combined with the similar examples to form the pool of unlabeled examples that active learning selects from.

The hope with active learning is that good examples will be drawn more often and we'll need to tag fewer examples to learn a classifier. In this case the good examples are assumed to come from the set of similar examples, attempting to determine when the notion of similarity from the Lin list applies. It appears this does happen in execution with approximately 94% of examples selected being drawn from the similar pool. The expected proportion of random selections from the similar pool is about 85%, because the method of extracting these pools of data is already biased to form a larger set of similar verbs than random.

There is a qualitatively interesting procession of examples presented to the human annotator by the active learner. The similarity measure used to create Lin's list is based on similar shared context features as we are using in our classifier, so it makes sense that these are presented as good examples, they were put in the similarity list because they already have more in common. The problem is there is still some overly common verbs included in the list that at the beginning dominate the examples presented. Words such as 'have' and 'say' occur often enough in the corpora that they will have very high context overlap with target word. For the first 50 or 60 examples these words appear very frequently and are overwhelmingly negative for the set of instances we are training for. The end result for this is that pairwise features are diminished in weight and so local context features begin to factor more into the decision, and better examples start to be presented and eventually pairwise features for real positive examples get boosted.

3.4 Case Studies

A positive example for *grow* $\leftarrow$ *increase* that f_u gets right is the sentence "Last quarter, company profits grew to 100." If we look at the feature weights that the linear classifier learns for *increase* we see that a positive judgment is associated with seeing context features such as subject congress, budget, force, or objects such as attractiveness, spending, rate, or more specifically, a passive verb with object profit, as is the case here. Taking context into account, the f_u classifier also gets the negative example correct: "I grow grapes on the south slope." f_u for *increase* was able to distinguish between *grow* as in "get larger" and *grow* as in "age and mature" yet it had never seen an example of *grow* in training.

Taking a look at a classifier that doesn't do so well, yet has seen this exact v in training, we can look at the classifier for *toughen*, which only gets 4 of its 10 test examples correct. If we examine the feature weight for the *toughen* classifier we see that the target word being *tighten* has high weight, along with words like security and policy, which are common contexts for when *toughen* can replace *tighten*. So when in the test data the sentence "The recent robberies made us reconsider tightening our security a bit" is presented the classifier correctly predicts positive. Yet, when given sentence "He tightened the cinches over the saddle basket" there are not many contextual clues that the *toughen* classifier has seen before, so it falls back to the judgment that tighten can be replaced. In effect, by the nature of the overlap features, when the f_u classifiers are trained with a specific v, they have a tendency to overfit, making future context based decisions difficult.

An interesting case study for the global classifier is on a case where it is not only applied to an unseen word pair, but to a structure slightly more complex than a single verb or phrasal verb. In the sentence "I sent a request for a typewriter", can "sent a request" be replaced by "asked". In this case, yes it can, and in fact the global classifier agrees. What it

send a request ← ask
I sent a request for a typewriter.
The groom sent a request to the DJ.

Table 5: Small example of replacing slightly more complicated structure than simple verbs or verb phrases.

is really looking at is if "ask" can replace the target verb "send" which has object "request". If the same query is made on the sentence "The groom sent a request to the DJ", the answer this time is no, its a different sense of request and "send a request" in general, and again the global classifier agrees.

4 Conclusion and Future Work

We have presented here a formalism that attempts to abstract away notions of meaning and sense that seem necessary to reasoning about any sort of paraphrasing. It is not necessary to rely on a fixed notion of what word can replace another or what one word means. Instead it is better to ask what do these words share, and can that be placed in our given context. At this point the problem is pushed onto the features. Can we capture shared meaning that is appropriate, non-symmetric and context sensitive. We demonstrated a notion of common features and local context here that despite being trained on a relatively small number of training examples showed promise in our testing.

In the future we would like to see this work extended to more complicated structures, as hinted in the "send a request" example. At some point phrases become too rare to ever generate training data, but we also hypothesize they have fewer possible meanings. We will need to capture a notion of similarity of structure as well as the words inside, plus expand what context would mean in such cases.

References

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings HLT-NAACL*, pages 16–23.

R. de Salvo Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. 2005. An inference model for semantic entailment in natural language. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1678–1679.

Oren Glickman and Ido Dagan. 2003. Identifying lexical paraphrases from a single corpus: A case study for verbs. In *Recent Advantages in Natural Language Processing (RANLP-03)*.

Ido Dagan Idan Szpektor, Hristo Tanev and Bonaventura Coppola. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP 2004*.

Oren Glickman Ido Dagan and Bernardo Magnini. 2005. The pascal recognizing textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1993. Principal-based parsing without overgeneration. In *Proceedings of ACL-93*, pages 112–120.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING-ACL-98*.

Nick Littlestone. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318.

George Miller. 1995. Wordnet: A lexical database. *Communication of the ACM*, 38(11).

Ido Dagan Oren Glickman and Moshe Koppel. 2005. A probabilistic classification approach for lexical textual entailment. In *Proceedings of AAAI 2005*.

Martha Palmer. 2000. Consistent criteria for sense distinctions. *Computers and the Humanities*, 34.

A. Ng R. Raina and C. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI 2005*.

Andrew Y. Ng Rion Snow, Daniel Jurafsky. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS 17*.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 806–813.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, pages 45–66.