Semantic Neighborhoods as Hypergraphs

Chris Quirk and **Pallavi Choudhury**
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{chrisq,pallavic}@microsoft.com

Abstract

Ambiguity preserving representations such as lattices are very useful in a number of NLP tasks, including paraphrase generation, paraphrase recognition, and machine translation evaluation. Lattices compactly represent lexical variation, but word order variation leads to a combinatorial explosion of states. We advocate hypergraphs as compact representations for sets of utterances describing the same event or object. We present a method to construct hypergraphs from sets of utterances, and evaluate this method on a simple recognition task. Given a set of utterances that describe a single object or event, we construct such a hypergraph, and demonstrate that it can recognize novel descriptions of the same event with high accuracy.

1 Introduction

Humans can construct a broad range of descriptions for almost any object or event. In this paper, we will refer to such objects or events as groundings, in the sense of grounded semantics. Examples of groundings include pictures (Rashtchian et al., 2010), videos (Chen and Dolan, 2011), translations of a sentence from another language (Dreyer and Marcu, 2012), or even paraphrases of the same sentence (Barzilay and Lee, 2003).

One crucial problem is recognizing whether novel utterances are relevant descriptions of those groundings. In the case of machine translation, this is the evaluation problem; for images and videos, this is recognition and retrieval. Generating descriptions of events is also often an interesting task: we might like to find a novel paraphrase for a given sentence, or generate a description of a grounding that meets certain criteria (*e.g.*, brevity, use of a restricted vocabulary).

Much prior work has used lattices to compactly represent a range of lexical choices (Pang et al., 2003). However, lattices cannot compactly represent alternate word orders, a common occurrence in linguistic descriptions. Consider the following excerpts from a video description corpus (Chen and Dolan, 2011):

- A man is sliding a cat on the floor.
- A boy is cleaning the floor with the cat.
- A cat is being pushed across the floor by a man.

Ideally we would like to recognize that the following utterance is also a valid description of that event: *A cat is being pushed across the floor by a boy.* That is difficult with lattice representations.

Consider the following context free grammar:

$$
\begin{array}{rcl}
\mathrm{S} & \rightarrow & \mathrm{X}_0\,\mathrm{X}_1 \\
 & | & \mathrm{X}_2\,\mathrm{X}_3 \\
\mathrm{X}_0 & \rightarrow & \textit{a man} \mid \textit{a boy} \\
\mathrm{X}_1 & \rightarrow & \textit{is sliding}\ \mathrm{X}_2\ \textit{on}\ \mathrm{X}_4 \\
 & | & \textit{is cleaning}\ \mathrm{X}_4\ \textit{with}\ \mathrm{X}_2 \\
\mathrm{X}_2 & \rightarrow & \textit{a cat} \mid \textit{the cat} \\
\mathrm{X}_3 & \rightarrow & \textit{is being pushed across}\ \mathrm{X}_4\ \textit{by}\ \mathrm{X}_0 \\
\mathrm{X}_4 & \rightarrow & \textit{the floor}
\end{array}
$$

This grammar compactly captures many lexical and syntactic variants of the input set. Note how the labels act as a kind of multiple-sequence-alignment allowing reordering: spans of tokens covered by the same label are, in a sense, aligned. This hypergraph or grammar represents a ***semantic neighborhood***: a set of utterances that describe the same entity in a semantic space.

Semantic neighborhoods are defined in terms of a grounding. Two utterances are neighbors with respect to some grounding (semantic event) if they are both descriptions of that grounding. Paraphrases, in contrast, may be defined over all possible groundings. That is, two words or phrases

are considered paraphrases if there exists some grounding that they both describe. The paraphrase relation is more permissive than the semantic neighbor relation in that regard. We believe that it is much easier to define and evaluate semantic neighbors. Human annotators may have difficulty separating paraphrases from unrelated or merely related utterances, and this line may not be consistent between judges. Annotating whether an utterance clearly describes a grounding is a much easier task.

This paper describes a simple method for constructing hypergraph-shaped Semantic Neighborhoods from sets of expressions describing the same grounding. The method is evaluated in a paraphrase recognition task, inspired by a CAPTCHA task (Von Ahn et al., 2003).

2 Inducing neighborhoods

Constructing a hypergraph to capture a set of utterances is a variant of grammar induction. Given a sample of positive examples, we infer a compact and accurate description of the underlying language. Conventional grammar induction attempts to define the set of grammatical sentences in the language. Here, we search for a grammar over the fluent and adequate descriptions of a particular input. Many of the same techniques still apply.

Rather than starting from scratch, we bootstrap from an existing English parser. We begin by parsing the set of input utterances. This parsed set of utterances acts as a sort of treebank. Reading off a grammar from this treebank produces a grammar that can generate not only the seed sentences, but also a broad range of nearby sentences. In the case above with *cat*, *man*, and *boy*, we would be able to generate cases legitimate variants where *man* was replaced by *boy* as well as undesired variants where *man* is replaced by *cat* or *floor*. This initial grammar captures a large neighborhood of nearby utterances including many such undesirable ones. Therefore, we refine the grammar.

Refinements have been in common use in syntactic parsing for years now. Inspired by the result that manual annotations of Treebank categories can substantially increase parser accuracy (Klein and Manning, 2003), several approaches have been introduced to automatically induce latent symbols on existing trees. We use the split-merge method commonly used in syntactic parsing (Petrov et al., 2006). In its original setting, the refinements captured details beyond that of the original Penn Treebank symbols. Here, we capture both syntactic and semantic regularities in the descriptions of a given grounding.

As we perform more rounds of refinement, the grammar becomes tightly constrained to the original sentences. Indeed, if we iterated to a fixed point, the resulting grammar would parse only the original sentences. This is a common dilemma in paraphrase learning: the safest meaning preserving rewrite is to change nothing. We optimize the number of split-merge rounds for task-accuracy; two or three rounds works well in practice. Figure 1 illustrates the process.

2.1 Split-merge induction

We begin with a set of utterances that describe a specific grounding. They are parsed with a conventional Penn Treebank parser (Quirk et al., 2012) to produce a type of treebank. Unlike conventional treebanks which are annotated by human experts, the trees here are automatically created and thus are more likely to contain errors. This treebank is the input to the split-merge process.

Split: Given an input treebank, we propose refinements of the symbols in hopes of increasing the likelihood of the data. For each original symbol in the grammar such as NP, we consider two latent refinements: NP_0 and NP_1. Each binary rule then produces 8 possible variants, since the parent, left child, and right child now have two possible refinements. The parameters of this grammar are then optimized using EM. Although we do not know the correct set of latent annotations, we can search for the parameters that optimize the likelihood of the given treebank. We initialize the parameters of this refined grammar with the counts from the original grammar along with a small random number. This randomness prevents EM from starting on a saddle point by breaking symmetries; Petrov et al. describe this in more detail.

Merge: After EM has run to completion, we have a new grammar with twice as many symbols and eight times as many rules. Many of these symbols may not be necessary, however. For instance, nouns may require substantial refinement to distinguish a number of different actors and objects, where determiners might not require much refinement at all. Therefore, we discard the splits that led to the least increase in likelihood, and then reestimate the grammar once again.

(a) Input:

- the man plays the piano
- the guy plays the keyboard

(b) Parses:

- (S (NP (DT the) (NN man))
  (VP (VBZ plays)
  (NP (DT the) (NN piano)))
- (S (NP (DT the) (NN guy))
  (VP (VBZ plays)
  (NP (DT the) (NN keyboard)))

(c) Parses with latent annotations:

- (S (NP_0 (DT the) (NN_0 man))
  (VP (VBZ plays)
  (NP_1 (DT the) (NN_1 piano)))
- (S (NP_0 (DT the) (NN_0 guy))
  (VP (VBZ plays)
  (NP_1 (DT the) (NN_1 keyboard)))

(d) Refined grammar:

$$
\begin{aligned}
S &\rightarrow NP_0\ VP \\
NP_0 &\rightarrow DT\ NN_0 \\
NP_1 &\rightarrow DT\ NN_1 \\
NP &\rightarrow VBZ\ NP_1 \\
DT &\rightarrow \textit{the} \\
NN_0 &\rightarrow \textit{man} \mid \textit{guy} \\
NN_1 &\rightarrow \textit{piano} \mid \textit{keyboard} \\
VBZ &\rightarrow \textit{plays}
\end{aligned}
$$

Figure 1: Example of hypergraph induction. First a conventional Treebank parser converts input utterances (a) into parse trees (b). A grammar could be directly read from this small treebank, but it would conflate all phrases of the same type. Instead we induce latent refinements of this small treebank (c). The resulting grammar (d) can match and generate novel variants of these inputs, such as *the man plays the keyboard* and *the buy plays the piano*. While this simplified example suggests a single hard assignment of latent annotations to symbols, in practice we maintain a distribution over these latent annotations and extract a weighted grammar.

Iteration: We run this process in series. First the original grammar is split, then some of the least useful splits are discarded. This refined grammar is then split again, with the least useful splits discarded once again. We repeat for a number of iterations based on task accuracy.

Final grammar estimation: The EM procedure used during split and merge assigns fractional counts $c(\cdots)$ to each refined symbol X_i and each production $X_i \rightarrow Y_j\ Z_k$. We estimate the final grammar using these fractional counts.

$$P(X_i \rightarrow Y_j\ Z_k) = \frac{c(X_i, Y_j, Z_k)}{c(X_i)}$$

In Petrov et al., these latent refinements are later discarded as the goal is to find the best parse with the original coarse symbols. Here, we retain the latent refinements during parsing, since they distinguish semantically related utterances from unrelated utterances. Note in Figure 1 how NN_0 and NN_1 refer to different objects; were we to ignore that distinction, the parser would recognize semantically different utterances such as *the piano plays the piano*.

2.2 Pruning and smoothing

For both speed and accuracy, we may also prune the resulting rules. Pruning low probability rules increases the speed of parsing, and tends to increase the precision of the matching operation at the cost of recall. Here we only use an absolute threshold; we vary this threshold and inspect the impact on task accuracy. Once the fully refined grammar has been trained, we only retain those rules with a probability above some threshold. By varying this threshold t we can adjust precision and recall: as the low probability rules are removed from the grammar, precision tends to increase and recall tends to decrease.

Another critical issue, especially in these small grammars, is smoothing. When parsing with a grammar obtained from only 20 to 50 sentences, we are very likely to encounter words that have never been seen before. We may reasonably reject such sentences under the assumption that they are describing words not present in the training corpus. However, this may be overly restrictive: we might see additional adjectives, for instance. In this work, we perform a very simple form of smoothing. If the fractional count of a word given a pre-terminal symbol falls below a threshold k, then we consider that instance rare and reserve a fraction of its probability mass for unseen words. This accounts for lexical variation of the grounding, especially in the least consistently used words.

Substantial speedups could be attained by using finite state approximations of this grammar: matching complexity drops to cubic to linear in the length of the input. A broad range of approximations are available (Nederhof, 2000). Since the small grammars in our evaluation below seldom exhibit self-embedding (latent state identification

tends to remove recursion), these approximations would often be tight.

3 Experimental evaluation

We explore a task in description recognition. Given a large set of videos and a number of descriptions for each video (Chen and Dolan, 2011), we build a system that can recognize fluent and accurate descriptions of videos. Such a recognizer has a number of uses. One example currently in evaluation is a novel CAPTCHAs: to differentiate a human from a bot, a video is presented, and the response must be a reasonably accurate and fluent description of this video.

We split the above data into training and test. From the training sets, we build a set of recognizers. Then we present these recognizers with a series of inputs, some of which are from the held out set of correct descriptions of this video, and some of which are from descriptions of other videos. Based on discussions with authors of CAPTCHA systems, a ratio of actual users to spammers of 2:1 seemed reasonable, so we selected one negative example for every two positives. This simulates the accuracy of the system when presented with a simple bot that supplies random, well-formed text as CAPTCHA answers.[1]

As a baseline, we compare against a simple tf-idf approach. In this baseline we first pool all the training descriptions of the video into a single virtual document. We gather term frequencies and inverse document frequencies across the whole corpus. An incoming utterance to be classified is scored by computing the dot product of its counted terms with each document; it is assigned to the document with the highest dot product (cosine similarity).

Table 2 demonstrates that a baseline tf-idf approach is a reasonable starting point. An oracle selection from among the top three is the best performance – clearly this is a reasonable approach. That said, grammar based approach shows improvements over the baseline tf-idf, especially in recall. Recall is crucial in a CAPTCHA style task: if we fail to recognize utterances provided by humans, we risk frustration or abandonment of the service protected by the CAPTCHA. The relative importance of false positives versus false negatives

[1] A bot might perform object recognition on the videos and supply a stream of object names. We might simulate this by classifying utterances consisting of appropriate object words but without appropriate syntax or function words.

Total videos		2,029
Training	descriptions	22,198
	types	5,497
	tokens	159,963
Testing	descriptions	15,934
	types	4,075
	tokens	114,399

Table 1: Characteristics of the evaluation data. The descriptions from the video description corpus are randomly partitioned into training and test.

(a)

Algorithm	S	k	Prec	Rec	F-0
tf-idf			99.9	46.6	63.6
tf-idf (top 3 oracle)			99.9	65.3	79.0
grammar	2	1	86.6	51.5	64.6
	2	4	80.2	62.6	70.3
	2	16	74.2	74.2	74.2
	2	32	73.5	76.4	74.9
	3	1	91.1	43.9	59.2
	3	4	83.7	54.4	65.9
	3	16	77.3	65.7	71.1
	3	32	76.4	68.1	72.0
	4	1	94.1	39.7	55.8
	4	4	85.5	51.1	64.0
	4	16	79.1	61.5	69.2
	4	32	78.2	63.9	70.3

(b)

t	S	Prec	Rec	F-0
$\geq 4.5 \times 10^{-5}$	2	74.8	73.9	74.4
$\geq 4.5 \times 10^{-5}$	3	79.6	60.9	69.0
$\geq 4.5 \times 10^{-5}$	4	82.5	53.2	64.7
$\geq 3.1 \times 10^{-7}$	2	74.2	75.0	74.6
$\geq 3.1 \times 10^{-7}$	3	78.1	64.6	70.7
$\geq 3.1 \times 10^{-7}$	4	80.7	58.8	68.1
> 0	2	73.4	76.4	74.9
> 0	3	76.4	68.1	72.0
> 0	4	78.2	63.9	70.3

Table 2: Experimental results. (a) Comparison of tf-idf baseline against grammar based approach, varying several free parameters. An oracle checks if the correct video is in the top three. For the grammar variants, the number of splits S and the smoothing threshold k are varied. (b) Variations on the rule pruning threshold t and number of split-merge rounds S. > 0 indicates that all rules are retained. Here the smoothing threshold k is fixed at 32.

(a) Input descriptions:

- A cat pops a bunch of little balloons that are on the groung.
- A dog attacks a bunch of balloons.
- A dog is biting balloons and popping them.
- A dog is playing balloons.
- A dog is playing with balloons.
- A dog is playing with balls.
- A dog is popping balloons with its teeth.
- A dog is popping balloons.
- A dog is popping balloons.
- A dog plays with a bunch of balloons.
- A small dog is attacking balloons.
- The dog enjoyed popping balloons.
- The dog popped the balloons.

(b) Top ranked yields from the resulting grammar:

+0.085	A dog is popping balloons.
+0.062	A dog is playing with balloons.
+0.038	A dog is playing balloons.
0.038	A dog is attacking balloons.
+0.023	A dog plays with a bunch of balloons.
+0.023	A dog attacks a bunch of balloons.
0.023	A dog pops a bunch of balloons.
0.023	A dog popped a bunch of balloons.
0.023	A dog enjoyed a bunch of balloons.
0.018	The dog is popping balloons.
0.015	A dog is biting balloons.
0.015	A dog is playing with them.
0.015	A dog is playing with its teeth.

Figure 2: Example yields from a small grammar. The descriptions in (a) were parsed as-is (including the typographical error "groung"), and a refined grammar was trained with 4 splits. The top k yields from this grammar along with the probability of that derivation are listed in (b). A '+' symbol indicates that the yield was in the training set. No smoothing or pruning was performed on this grammar.

may vary depending on the underlying resource. Adjusting the free parameters of this method allows us to achieve different thresholds. We can see that rule pruning does not have a large impact on overall results, though it does allow yet another means of tradiing off precision vs. recall.

4 Conclusions

We have presented a method for automatically constructing compact representations of linguistic variation. Although the initial evaluation only explored a simple recognition task, we feel the underlying approach is relevant to many linguistic tasks including machine translation evaluation, and natural language command and control systems. The induction procedure is rather simple but effective, and addresses some of the reordering limitations associated with prior approaches.(Barzilay and Lee, 2003) In effect, we are performing a multiple sequence alignment that allows reordering operations. The refined symbols of the grammar act as a correspondence between related inputs.

The quality of the input parser is crucial. This method only considers one possible parse of the input. A straightforward extension would be to consider an n-best list or packed forest of input parses, which would allow the method to move past errors in the first input process. Perhaps also this reliance on symbols from the original Treebank is not ideal. We could merge away some or all of the original distinctions, or explore different parameterizations of the grammar that allow more flexibility in parsing.

The handling of unseen words is very simple. We are investigating means of including additional paraphrase resources into the training to increase the effective lexical knowledge of the system. It is inefficient to learn each grammar independently. By sharing parameters across different groundings, we should be able to identify Semantic Neighborhoods with fewer training instances.

Acknowledgments

We would like to thank William Dolan and the anonymous reviewers for their valuable feedback.

References

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

David Chen and William Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200, Portland, Oregon, USA, June. Association for Computational Linguistics.

Markus Dreyer and Daniel Marcu. 2012. Hyter: Meaning-equivalent semantics for translation evaluation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 162–171, Montréal, Canada, June. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo,

Japan, July. Association for Computational Linguistics.

Mark-Jan Nederhof. 2000. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1):17–44, March.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Chris Quirk, Pallavi Choudhury, Jianfeng Gao, Hisami Suzuki, Kristina Toutanova, Michael Gamon, Wentau Yih, Colin Cherry, and Lucy Vanderwende. 2012. Msr splat, a language analysis toolkit. In *Proceedings of the Demonstration Session at the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 21–24, Montréal, Canada, June. Association for Computational Linguistics.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147, Los Angeles, June. Association for Computational Linguistics.

Luis Von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. 2003. Captcha: Using hard ai problems for security. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer Berlin Heidelberg.