Augmenting Semantic Parsing with Word Embeddings

Brendon Boldt

Dec/11/2017

Abstract

Semantic parsing provides a powerful way to demonstrate understanding of a natural language by constructing easily executable, formal representations from natural language utterances. One issue with semantic parsing, though, is that it treats words merely as tokens with no individual semantic content and has no mechanism for inferring the semantics of an unknown word. Using word embeddings (words represented as vector in semantic parsing provides an effective solution to this problem since it allows the semantic parser to infer the meaning of new words by comparing their similarity to known words. Furthermore, vector word embeddings can be learned at a large scale since it is an unsupervised algorithm.

1 Introduction

One of the major subfields of natural language processing is natural language *understanding*, which carries the connotation that the system processing the language is able to find meaning in an utterance rather than simply doing something such as searching for keyword occurences or otherwise. One such method of demonstrating an understanding of a natural language is by having the system in question produce a concrete action or piece of information which has semantic matching that of the utterance.

Semantic parsing, on a general level, is an effective way of demonstrating an understanding of natural language because it maps utterances to formal representations (e.g., lambda-DCS, SQL) based on a formal grammar. Semantic parsing has been applied to tasks such as answering questions, interpreting email-related commands, and building virtual block structures [1] [2] [5].

Like most formal grammars, semantic parsing grammars usually treat each word as either a unique or lemmatized[1] token. In this model, each word has no meaning until it is explicitly assigned one by the semantics of the grammar. This inflexibility is important in applications of formal grammars like programming languages where exactness is expected and required, but this trait is not so helpful when parsing natural language with a formal grammar. For example, a semantic parser might interpret "build large box" while not understanding "create large box" which, for the large part, mean the same thing. Understanding natural language requires flexibility in that words with similar meaning should have *some* degree of interchangeability as opposed to none which is offered by the typical token-based approach.

It would be better, then, to represent words in a way that would group words with similar meanings, and word embeddings, that is, representing words as vectors gives us a very efficient way to do this. One of the primary advantages of augmenting semantic parsing with word embeddings as opposed to dictionary synonyms is that learning the embeddings is a unsupervised task which allows it to be done at scale much more easily than hand-picking sets of words which are semantically similar enough to a given token.

2 Methodology

In order to test the ability of word embeddings to augment semantic parsing, we will first describe a simple grammar consisting of a handful of categories (e.g., $\langle Action \rangle$, $\langle Object \rangle$) which each have 2 or 3 tokens

[1] Lemmatized tokens account for word inflections; for example, "walks", "walked", and "walking" would all match "walk" when lemmatized.

associated with them ("in-grammar tokens"). For the purposes of this paper, we will be using the following grammar:

⟨*Root*⟩ ::= ⟨*Action*⟩

⟨*Action*⟩ ::= ⟨*ActionVerb*⟩ ⟨*Object*⟩

⟨*ActionVerb*⟩ ::= '`build`' | '`remove`' | '`find`'

⟨*Object*⟩ ::= ⟨*ObjectItem*⟩ | ⟨*Size*⟩ ⟨*ObjectItem*⟩

⟨*ObjectItem*⟩ ::= '`box`' | '`bar`'

⟨*Size*⟩ ::= '`large`' | '`medium`' | '`small`'

The semantic aspect of the grammar might refer to a robot which can perform the actions described by the valid sentences formed by the above grammar. While this grammar is not terribly large or expressive, relatively simple grammars can be used to form the basis for a semantic parsing task where the grammar grows through use over the lifetime of the system such as in the case of Voxelurn [5].

Normally, semantic parsing looks for exact matches between the in-grammar tokens and the tokens in the utterance; if one of the tokens in the utterance does not match one of the in-grammar tokens for a specific category, the parse will fail. In the augmented semantic parser, an unknown token is treated as a wildcard token within the expected category so that the parse completes. The unknown token is then compared with the tokens in the grammatical category using the word embeddings to determine which in-grammar token is most suitable to use.

To test the ability of the word embeddings to help with inferring unknown tokens, we replace an in-grammar token with a synonym (loosely-defined); then we compare the synonym, an unknown token, with each in-grammar token for that grammatical category using the cosine similarity between the two word vectors and selecting the in-grammar token with the highest similarity, essentially minimizing the angle between the two vectors [3].

3 Experiment

For this experiment, we are generating synonyms using the RiWordNet[2] interface for WordNet 3.1. The interface allows us to generate the synonyms of each in-grammar token and filter these synonyms by particular senses of the given word. The synonyms for a particular sense of a word specifically refer to the union of its synset, hyponyms, similars, also-sees, and coordinate terms. Senses for each word were selected based on the assigned semantics of the grammar defined above. Table 1 shows the number of synonyms generated for the correct sense of each in-grammar token.

We are using the GloVe word vector model as a source for word embeddings [4]. In particular, we are using the pretrained set of vectors from the "Wikipedia 2014 + Gigaword 5" dataset which results in 6 billion tokens and 400k vocabulary words although only the most frequent 50k were used for our experiment. The word vectors came in dimensionalities of 50, 100, 200, and 300; each were tested in order to observe any effect that dimensionality had on performance.

The results of the experiment are listed in Table 2 and on Figure 1.

4 Discussion

Raw accuracies are displayed for the individual in-grammar tokens in Table 2 while adjusted accuracy is displayed in the "mean" column of Table 2 and in Figure 1. The mean is evenly weighted among in-grammar tokens, that is, an in-grammar token that had more synonyms does not influence the mean more than others. Adjustment to the accuracy was done by taking the raw accuracy and subtracting from it the probability of

[2] `https://rednoise.org/rita/reference/RiWordNet.html`

Table 1: Number of synonyms generated by WordNet for each token.

Word	Synonyms
build	123
remove	295
find	81
box	151
bar	25
large	88
medium	4
small	54

Table 2: The raw accuarcies achieved on test set of data. Categories are separated by vertical lines.

Dimensions	*build*	*remove*	*find*	*box*	*bar*	*large*	*medium*	*small*	Mean Adj.
50	0.278	0.528	0.686	0.769	0.231	0.691	0.750	0.400	0.167
100	0.316	0.463	0.784	0.231	0.788	0.764	0.500	0.700	0.194
200	0.342	0.444	0.745	0.788	0.385	0.855	0.500	0.850	0.239
300	0.380	0.519	0.706	0.808	0.462	0.855	0.500	0.850	0.260

randomly guessing the correct in-grammar token (thus 1/2 is subtracted from the accuracies of *box* and *ball* and 1/3 from the rest).

Looking at the accuracy, some synonyms were more often matched correctly in lower dimensions, but the mean of the adjusted accuracies showed a steady increase as vector dimensions increased. This is to be expected as a higher dimension account would reasonably give a more semantically nuanced representation of the word.

With the 300 dimension set of word vectors, we were able to achieve an average boost of 0.260 above randomly guessing the closest in-grammar match to unknown tokens. In instances of relatively short utterances and small grammars, applying this model would a "better than nothing" situation since an increased ability to recognize unknown tokens comes only at the cost of training the word vectors once (or using a pre-trained model). The computational costs of calculating similarities is relatively small (since it primarily dot and scalar vector products). Grammars with large numbers of in-grammar tokens for a given grammatical category or with large parse trees would be more likely to see a slow-down related to computing similarities, but definitive answers to this require further investigation.

Further experimentation would involve using human subjects (e.g., using Amazon Mechanical Turk) to directly test whether the robustness added by the word embeddings is effective or not. The effectiveness would be determined by the ability of the semantic parser to address the natural variations in language as used by humans rather than just the strict synonym replacement demonstrated in this paper.

5 Conclusion

The core method of representing words as vector is looking at the context that words are in, and taking this into account in future research would be a promising way to boost the accuracy of inferring unknown tokens with word embeddings. Namely, the words which surround the unknown token should directly factor into the similarity metric between the various pre-defined tokens and the one in question.

While the augmentation of the semantic parsing grammar ranged from significant to absent, the technique shows on a basic level that it is possible to allow to semantic parsing models to make inferences on unseen tokens without the need for more test data or supervised learning learning procedures. Especially in semantic parsing applications where the system *must* make a decision of some sort, an educated guess is better than no guess at all.

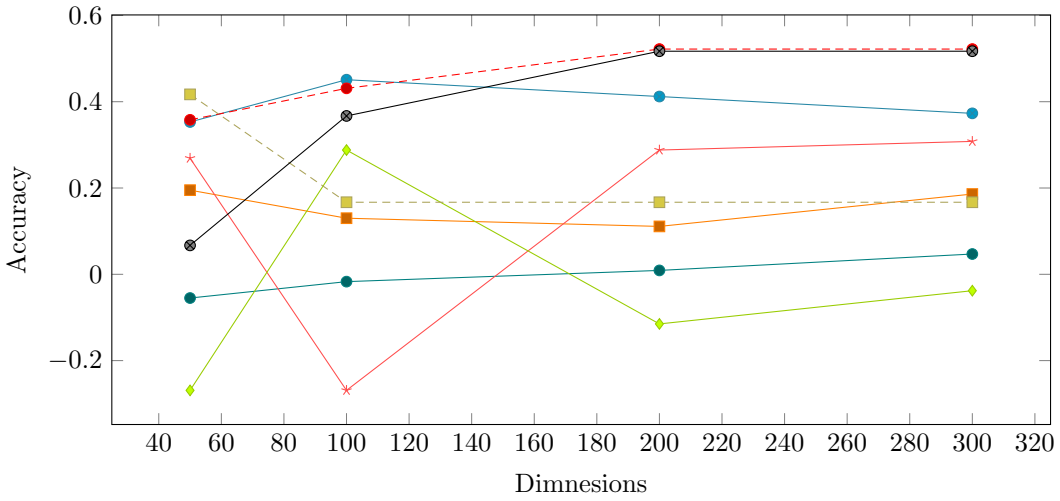

Figure 1: Change in accuracy shown at different dimension spaces. The change in accuracy is defined as the raw accuracy minus 1 over number of tokens within the category.

References

[1] Amos Azaria, Jayant Krishnamurthy, and Tom Mitchell. Instructable intelligent personal agent, 2016.

[2] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL, 2013.

[3] Kaname Kasahara, Tsuneaki Kato, and Christopher Manning. Synonym retrieval using word vectors from text data.

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[5] Sida I. Wang, Samuel Ginn, Percy Liang, and Christopher D. Manning. Naturalizing a programming language via interactive learning. *CoRR*, abs/1704.06956, 2017.

6 Notes

I originally intended to have a working version of the modified semantic parser completed by the end of the semester, but I ended up not having enough time to complete my modifications of the SEMPRE semantic parser. You can find my fork of the repository with my partial progress at `https://github.com/brendon-boldt/sempre/tree/embeddings`.