|  |
| --- |
| **Using spaCy Pipeline Components to Improve Embeddings for DrQA Model** |
|  |
| **Yijun He, Xuan Ting Liu** |
|  |
|  |
|  |
|  |

Abstract

In this paper, we discuss the improvements to the model in the DrQA (Chen et al., 2017) paper with SQuAD data using additional *parser* and *dependency* tags from the spaCy library using the pretrained pipeline, with customized tokenizer. Inserting the information regarding parts of the passage to the embeddings in the model, we see incremental improvements in both EM and F1 scores that reflect the supplementary data given.

Credits

The results from this document have been adapted by Yijun He and Xuan Ting Liu from the instructions and starter code for C S 388, Natural Language Processing, at the University of Texas at Austin. The SQuAD dataset was also provided as a download link.

Introduction

The Stanford Question Answering Dataset (SQuAD) dataset is a reading comprehension dataset that contains questions about Wikipedia articles and their answers as (a passage of) text.

The goal of the model is to accurately predict each word of the answers.

Existing model components from the DrQA paper:

* Embedding
* Embedding-spy
* Aligned attention
* Passage RNN
* Question RNN
* Dropout
* Question attention
* Output

Scope

The improvements in the existing model are based on additional linguistics constraints for a more robust model. Specifically, we are integrating the spaCy library in order to take advantage of the language processing pipeline. Using spaCy, the text is converted from tokens to part-of-speech tags and dependency labels using the Tagger and Parser respectively. This information is then passed into the embeddings for the model.

Implementation

Timeline

Description automatically generated

Figure 1: SpaCy NLP trained pipeline

We use the spaCy “en\_core\_web\_sm” pre-trained model in order to generate the part-of-speech tags and dependency labels. The “en” is the English language, “core” for a general-purpose pipeline, “web” is from web data, and “sm” is the small size.

Tokenizer

The original base code includes a tokenizer class. We found that the tokenizer class in the code is more accurate at splitting tokens compared to the tokenizer from the spaCy library. For example, when the text reads “St. Joseph”, the spaCy tokenizer splits “st”, “.” and “joseph” as 3 tokens, whereas the original code base splits the passage more accurately into 2 tokens, “st.” and “joseph”, since we do not want “.” to be later tagged as punctuation to signify the end of a sentence. Therefore, we remove and replace the default tokenizer in the spaCy pipeline.

spaCy Tagger

The spaCy tagger assigns part-of-speech tags (token.tag\_) to each input token, using the default trainable model. For example, each input will be tagged noun, verb, or adverb. The actual spaCy tagger has 49 tags.

spaCy Parser

The spaCy dependency parser assigns dependency labels (token.dep\_) to each input token. As the spaCy model learns the labels of the sentence segments, we capture the transitional probabilities of subsequent words and predict a tag. The default parser is also used. See Figure 2 for illustration of the parser.

Embeddings

As the Tokenizer is customized (See 4.1 for detail), the Tagger and dependency Parser detected by spaCy English trained pipeline “en\_core\_web\_sm” are one-to-one correspondent to the “token\_context” and “question\_token” from the source data, which implies the time steps of the Tagger and Parser are in the same length as the “token\_context” and “question\_token”. This correspondence smooths out the ingestion process of tagger and parser information to the DrQA model without losing any content or question token information.

In order to pass the tagger or parser information to the model, we first assign an index to each of the possible outputs of the Tagger or Parser. These indexers would be translated to correspondent embeddings with a new embedding layer. Different from the “token\_context” and “question\_token”, the parser/tagger embeddings are not pre-trained from the Glove set. They are randomly initialized and trained as the model updates.

After indexing the dependency Parsers or the Taggers, we concatenate the correspondent embeddings with the word embeddings from Glove and parse the newly created embeddings to the existing DrQA model with some changes on the input dim of the encoders.

## **Training**

The DrQA model with parser/tagger takes more time to train than the baseline model. The majority of the extra time is spent on the parser/tagger generation by spaCy English pipeline. As we customized the tokenizer for each instance, it takes extra 10 minutes for the data preparation to transform all tokens to indexer with the customized token we created.

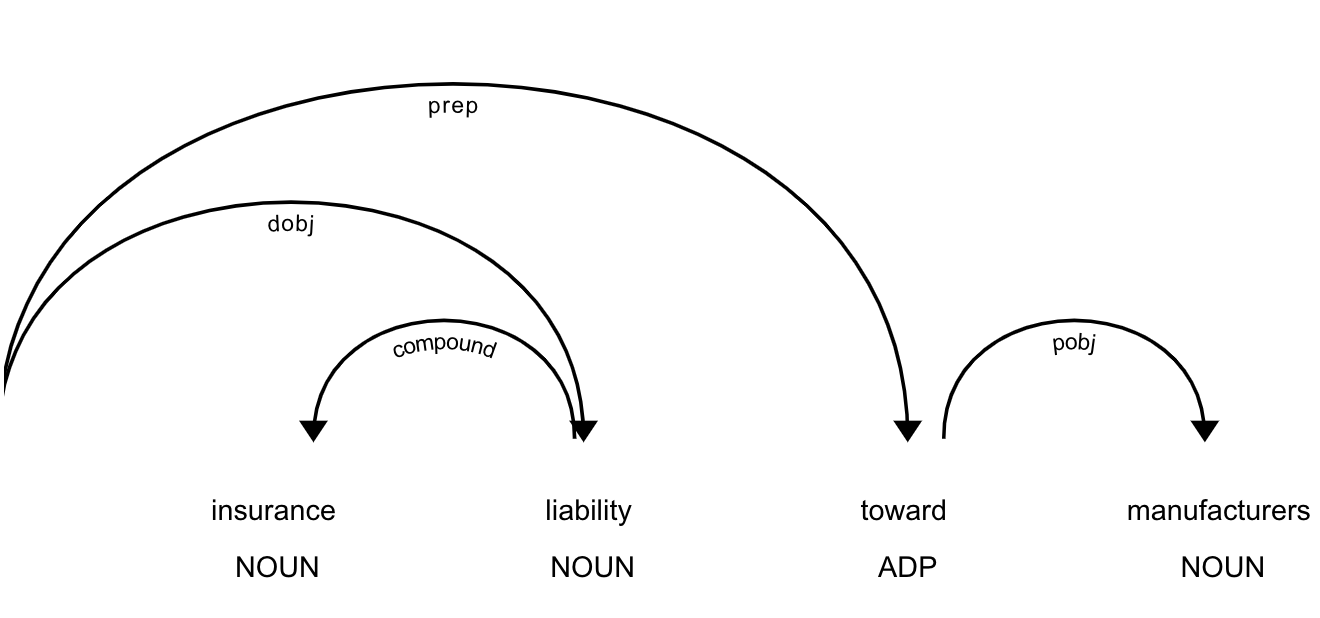
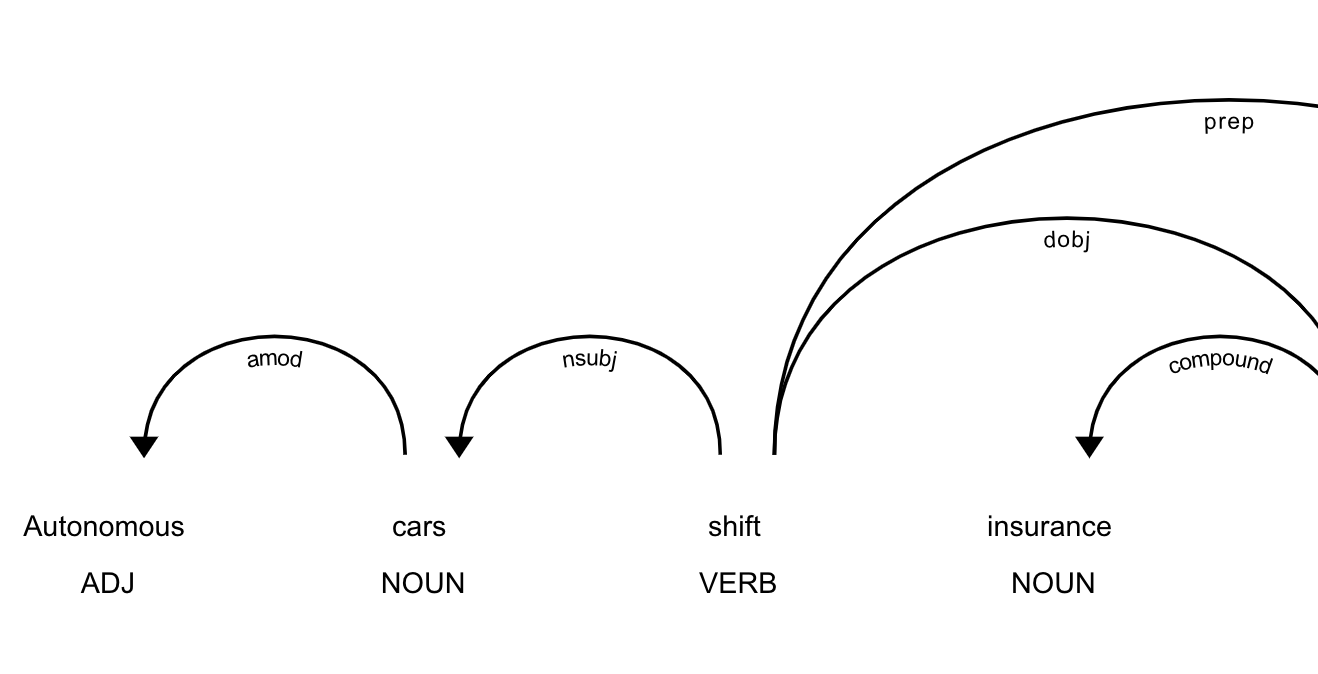
The rest of the components of the modeling takes almost the same time compared to the baseline model.

Results

|  |  |  |  |
| --- | --- | --- | --- |
| Model  (x + y hidden dim) | EM | F1 | Time |
| Baseline (300) | 48.98 | 61.33 |  |
| With tags (300 + 50) | 51.67 | 64.35 |  |
| With dependency  (300 +50) | 50.76 | 63.40 |  |

Table 1: Results Comparison with 300 hidden dim baseline

Reasons



References

TAGS, arrows show dependencies

Figure 2: SpaCy Tagger (capital letters) and Parser (arrows) when translating from spaCy documentation

References