Finding ARG1 of Partitive Nouns in NomBank with DistilBERT

Ziyang Zeng
Dept. of Computer Science
New York University
251 Mercer Street, New York, NY
zz2960@nyu.edu

Abstract

This document is a supplement to the general instructions for *ACL authors. It contains instructions for using the LaTeX style files for ACL conferences. The document itself conforms to its own specifications, and is therefore an example of what your manuscript should look like. These instructions should be used both for papers submitted for review and for final versions of accepted papers.

1 Introduction

NomBank (Meyers et al., 2004c) is a databank project at New York University that adds annotation layer of argument structure for instances in the Penn Treebank II corpus. Apart from the more commonly researched nominalizations of verbs and nominalizations of adjectives, it also covers relational nouns, partitive nouns and several other types of argument-taking nouns.

More recently, the NLP community has seen huge popularity in heavily adopting pretraining-based langauge models. BERT (Devlin et al., 2018) is a large-size language representation model trained on a large corpus of English text that can achieve state-of-the-art results on a variety of natural language processing tasks. BERT came a decade later than the release of NomBank and there's been little previous work using it to conduct NomBank-based tasks and related experiments. However, as powerful as BERT is, its large parameters size makes it significantly more computationally expensive to train and use. On the other hand, DistilBERT is a distilled student model of BERT that retains 97% of BERT's language understanding capabilities while being 40% smaller and 60% faster. This makes it a great candidate to train when facing with limited resources.

This paper presents a token classification approach using pre-trained DistilBERT to find ARG1 of partitive nouns in NomBank. It focuses on partitive nouns (nouns that are used to describe a part or quantity of something) and the partitive task. The task is described as finding one argument (ARG1) of % (the percent sign), or in other words, finding the none group that is being sub-divided or quantified over. For example, for the sentence "Output in the energy sector rose 3.8%.", the ARG1 to be found is "Output" since it is the partitive noun that "3.8%" is referring to. This task can be translated into a binary classification problem on each token in the sentence where the model predicts whether the token is an ARG1 or not.

In addition to the token classification approach above, this paper also experiments with question-answer task, transforming the original ARG1 finding task into the task of make the model answer the question "What is the ARG1 of this sentence?". The advantage of this adaptation is that the model would stably output exactly one ARG1 for each sentence whereas token classification could give each sentence 0 to N ARG1s.

2 Related Work

NomBank (Meyers et al., 2004c), as a databank extending the frames of NOMLEX and PropBank, annotates argument structures of common nouns similar to how PropBank annotates predicating verbs. The development of the NomBank corpus has made several other work on argument structure extraction more accessible. Jiang and Ng (2006) attempts the first NomBank-based automatic semantic role labeling system after the databank's release. Ping (2006) adapts a PropBank-based SRL system to the SRL task of NomBank, achieving an overall F1 score of 72.73 on section 23 of the NomBank corpus.

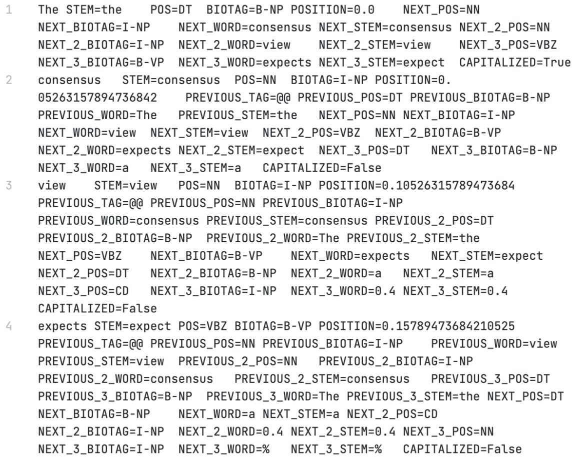
BERT, or Bidirectional Encoder Representations from Transformers (Devlin et al., 2018), has shown state-of-the-art results on many NLP tasks. There have been many attempts to compress BERT into a smaller model, among which DistilBERT (Sanh et al., 2019) successfully reduces the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster, leveraging knowledge distillation (Bucila et al., 2006, Hinton et al., 2015) during the pre-training phase.

More recent development on semantic role labeling (SRL) has shifted from feature engineering to architecture-based modeling leveraging deep neural networks (Collobert et al., 2011). Several recent notable approaches suggest only using raw tokenized text as input and let the model learn the features from the text itself (He et al., 2017; Sahin and Steedman, 2018; Marcheggiani et al., 2017; Strubell et al., 2018). This end-to-end approach allows the model to easily generalize from one SRL task to another without very task-specific feature engineering effort.

3 ARG1 Finding Pipelines

3.1 Maximum Entropy Baseline Pipeline

The baseline we use is a maximum entropy machine learning method in a token-by-token classification manner. Maximum entropy is a general technique for estimating probability distributions from data. Labeled training data is used to derive a set of constraints for the model that characterize the class-specific expectations for the distribution. To perform well, the model relies on the good representation of the features we feed it. In this paper, we select features including the word stem, the neighboring 2 words, POS tags, NG-BIO tags, whether capitalized and the position of the word in the sentence. Figure 1 shows the features we selected for the maximum entropy baseline.



```
The STEM=the    POS=DT  BIOTAG=B-NP POSITION=0.0    NEXT_POS=NN
NEXT_BIOTAG=I-NP    NEXT_WORD=consensus NEXT_STEM=consensus NEXT_2_POS=NN
NEXT_2_BIOTAG=I-NP  NEXT_2_WORD=view    NEXT_2_STEM=view    NEXT_3_POS=VBZ
NEXT_3_BIOTAG=B-VP  NEXT_3_WORD=expects NEXT_3_STEM=expect  CAPITALIZED=True
consensus   STEM=consensus  POS=NN  BIOTAG=I-NP POSITION=0.
05263157894736842    PREVIOUS_TAG=@@ PREVIOUS_POS=DT PREVIOUS_BIOTAG=B-NP
PREVIOUS_WORD=The   PREVIOUS_STEM=the   NEXT_POS=NN NEXT_BIOTAG=I-NP
NEXT_WORD=view  NEXT_STEM=view  NEXT_2_POS=VBZ  NEXT_2_BIOTAG=B-VP
NEXT_2_WORD=expects NEXT_2_STEM=expect  NEXT_3_POS=DT   NEXT_3_BIOTAG=B-NP
NEXT_3_WORD=a   NEXT_3_STEM=a   CAPITALIZED=False
view    STEM=view   POS=NN  BIOTAG=I-NP POSITION=0.10526315789473684
PREVIOUS_TAG=@@ PREVIOUS_POS=NN PREVIOUS_BIOTAG=I-NP
PREVIOUS_WORD=consensus PREVIOUS_STEM=consensus PREVIOUS_2_POS=DT
PREVIOUS_2_BIOTAG=B-NP  PREVIOUS_2_WORD=The PREVIOUS_2_STEM=the
NEXT_POS=VBZ    NEXT_BIOTAG=B-VP    NEXT_WORD=expects   NEXT_STEM=expect
NEXT_2_POS=DT   NEXT_2_BIOTAG=B-NP  NEXT_2_WORD=a   NEXT_2_STEM=a
NEXT_3_POS=CD   NEXT_3_BIOTAG=I-NP  NEXT_3_WORD=0.4 NEXT_3_STEM=0.4
CAPITALIZED=False
expects STEM=expect POS=VBZ BIOTAG=B-VP POSITION=0.15789473684210525
PREVIOUS_TAG=@@ PREVIOUS_POS=NN PREVIOUS_BIOTAG=I-NP    PREVIOUS_WORD=view
PREVIOUS_STEM=view  PREVIOUS_2_POS=NN   PREVIOUS_2_BIOTAG=I-NP
PREVIOUS_2_WORD=consensus   PREVIOUS_2_STEM=consensus   PREVIOUS_3_POS=DT
PREVIOUS_3_BIOTAG=B-NP  PREVIOUS_3_WORD=The PREVIOUS_3_STEM=the NEXT_POS=DT
NEXT_BIOTAG=B-NP    NEXT_WORD=a NEXT_STEM=a NEXT_2_POS=CD
NEXT_2_BIOTAG=I-NP  NEXT_2_WORD=0.4 NEXT_2_STEM=0.4 NEXT_3_POS=NN
NEXT_3_BIOTAG=I-NP  NEXT_3_WORD=%   NEXT_3_STEM=%   CAPITALIZED=False
```

Figure 1: Sample sentence features for maxent system

3.2 Plain Raw Input Pipeline

The simplest pipeline for finding ARG1 of partitive nouns in NomBank assumes that the input text is raw and unprocessed. The pipeline first tokenizes the text using the tokenizer provided by DistilBERT. Then, the model predicts whether each token is an ARG1 or not. The pipeline is illustrated in the following figure 2.

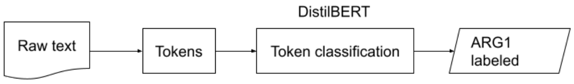


Figure 2: The simplest pipeline for finding ARG1 of partitive nouns in NomBank

3.3 Feature Enhanced Pipeline

NomBank provides a number of features for each token in the databank, including POS tags and NG-BIO tags. They could be helpful for the task of finding ARG1 of partitive nouns. The pipeline below incorporate these features to the raw text input in 3.2 to find ARG1 of partitive nouns. The pipeline is illustrated in the figure 3.

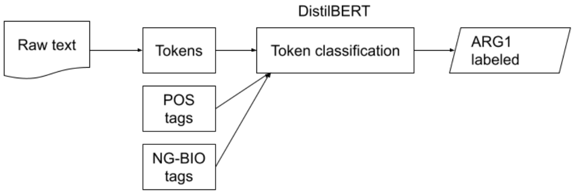


Figure 3: The feature-enhanced pipeline for finding ARG1 of partitive nouns in NomBank

Because the original pre-trained DistilBERT model does not take extra features including POS and NG-BIO tags as input, we have to modified the token classification model and alter the last linear layer, concatenating the original BERT hidden layer output with the extra features, then doing token classification through the linear layer.

3.4 Question-Answer Pipeline

Apart from token classification, we also explored question-answer approach which rephrase the original problem into a QA problem. Question-answering, just as the plain token classification approach, assumes tokenized input and no extra features. Additionally, question-answering tasks also requires a question prompt and answer location for each sample while training. The pipeline is illustrated in the figure 4.

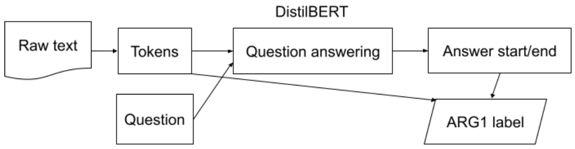


Figure 4: The question-answering pipeline for finding ARG1 of partitive nouns in NomBank

The output of the question-answering model is the location (start index and end index) of the predicted answer. By locating the token on the original input token list with the start index and end index, we can reconstruct the ARG1 label of the sentence. The answer relocation is a bit more complicated due to the tokenization discrepancies between the original input and the tokenized input. This is explained in detailed in section 4.2.

4 Experiments

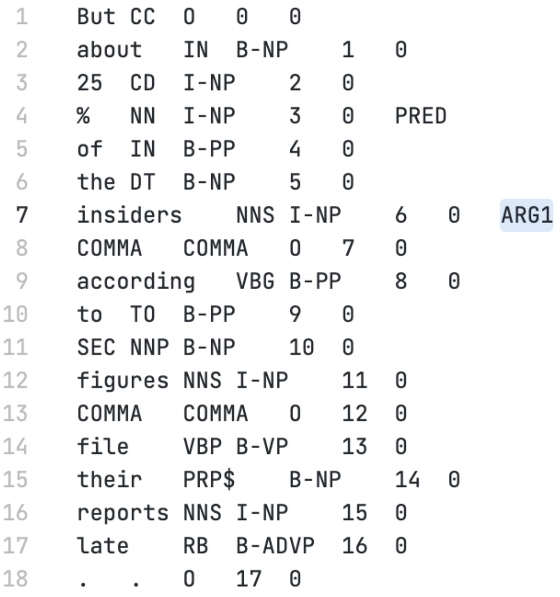
4.1 Dataset

The dataset we use in this task is part of the NomBank that specifically focus on partitive nouns. In the dataset is split into training, dev and test set. The training set has 2367 sentences or 66186 words, the dev set has 83 sentences or 2225 words, and the test set has 150 sentences or 4276 words. Each token in a sentence takes a line in the data file, the line containing the token itself, the POS tag, the NG-BIO tag, the index of the sentence and the index of the token in the sentence. The semantic role label is put at the end of each line, if the label is not one of "ARG1", "PRED" and "SUPPORT", then it's omitted. Figure 5 is one sample sentence from the training dataset.

In a sentence in the % dataset, the PRED labels % symbol, the ARG1 is the partitive noun that % symbol refers to, and the SUPPORT marks the word connecting the ARG1 and the PRED. It is guaranteed that there is one and only one ARG1 in each sentence of the dataset.

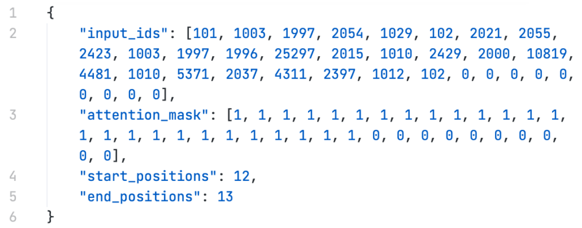
4.2 Data Preprocessing

Even though for deep neural network input wouldn't need much feature engineering, we still need to tokenize the text and convert the text into numerical representation so that it can be served as the input of the model. For DistilBERT-based pipelines, the tokenizer provided by DistilBERT is used to tokenize the text. However, the dataset is already tokenized and the way DistilBERT tokenizes a sentence could be different from how the sentences are already tokenized in the dataset. More specifically, the tokenizer from DistilBERT could tokenize the sentence into even smaller tokens than the tokenizer used in the dataset. Therefore, this could cause the output label of the model fail to align with the input given. Fortunately, the results of the DistilBERT tokenizer include the align information for the tokenization. With this information, we can reconstruct the tokenized sentence from the DistilBERT tokenizer output and match the model output with the original input.



```
But CC  0   0   0
about   IN  B-NP    1   0
25  CD  I-NP    2   0
%   NN  I-NP    3   0   PRED
of  IN  B-PP    4   0
the DT  B-NP    5   0
insiders    NNS I-NP    6   0   ARG1
COMMA   COMMA   0   7   0
according   VBG B-PP    8   0
to  TO  B-PP    9   0
SEC NNP B-NP    10  0
figures NNS I-NP    11  0
COMMA   COMMA   0   12  0
file    VBP B-VP    13  0
their   PRP$    B-NP    14  0
reports NNS I-NP    15  0
late    RB  B-ADVP  16  0
.   .   0   17  0
```

Figure 5: One sample sentence in the training dataset



```
{
    "input_ids": [101, 1003, 1997, 2054, 1029, 102, 2021, 2055,
    2423, 1003, 1997, 1996, 25297, 2015, 1010, 2429, 2000, 10819,
    4481, 1010, 5371, 2037, 4311, 2397, 1012, 102, 0, 0, 0, 0, 0,
    0, 0, 0, 0],
    "attention_mask": [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0],
    "start_positions": 12,
    "end_positions": 13
}
```

Figure 6: One sample sentence for QA task

A special treatment for the question-answering pipeline is that the ARG1 label has to be transformed into the answer location (the start and end indexes). And the question prompt for every sample is the same "What is the ARG1 of the sentence?". The exact same question prompt is used for validation and testing. Figure 6 is one processed sample sentence for the QA task, inputs are padded in batch and stacked as matrix to be able to be processed by the model in batches. `input_ids` are the numerized tokens of both the question prompt and the input text concatenated by `102`. `attention_mask` is the attention the model should be given to the tokens, `0` means the token should not be considered by the model. `start_position` and `end_position` are the start and end indexes of the answer in the input text.

4.3 Experiment Settings

All deep models used in this paper are implemented in PyTorch and with HuggingFace transformers libraries. The pre-trained models are fine-tuned on a single NVIDIA P100 (16GB) GPU on the Google Colab platform. The deep models are fine-tuned for 10 epochs with the learning rate of `2e-5` and weight decay of `0.01`. For token classification tasks, the batch size is 64. For question-answering tasks, the input would be bigger with the question prompt, so the batch size is set to 32 to be able to fit into the GPU's RAM.

4.4 Evaluation Metrics

To evaluate the performance of the model, we used precision, recall and F1-score as evaluation metrics. The precision and recall are calculated by the number of correct predictions divided by the total number of predictions. The F1-score is calculated by the harmonic mean of precision and recall. Below are the formulas for calculating the precision, recall and F1-score, in which TP is the number of true positives, FP is the number of false positives, FP is the number of false positives and FN is the number of false negatives.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3}$$

4.5 Partitive Dataset

Apart from the original % task, we also experimented the token classification of DistilBERT on the combined dataset of % and partitive datasets of NomBank. Different from the % dataset, there's not only ARG1 but also ARG0, ARG2... and more role labels in the partitive dataset. In order to adopt the same pipelines, we preprocessed the partitive dataset into the same format as the % dataset and combine them together into a larger dataset. The larger dataset is also split into training, dev and test set. The training set has 12991 sentences or 380247 words, the dev set has 426 sentences or 12671 words, and the test set has 746 sentences or 21536 words.

Acknowledgements

This document has been adapted by Steven Bethard, Ryan Cotterell and Rui Yan from the instructions for earlier ACL and NAACL proceedings, including those for ACL 2019 by Douwe Kiela and Ivan Vulić, NAACL 2019 by Stephanie Lukin and Alla Roskovskaya, ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Mitchell and Stephanie Lukin, BibTEX suggestions for (NA)ACL 2017/2018 from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, ACL 2010 by Jing-Shin Chang and Philipp Koehn, ACL 2008 by Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, ACL 2005 by Hwee Tou Ng and Kemal Oflazer, ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

A Example Appendix

This is an appendix.