Event Temporal State Identification

Vincent Chu, John Chiang
W266: Natural Language Processing & Deep Learning
UC MIDS

Abstract

An increasingly interconnected global society has amplified the potential impact societal events can have on outcomes of political elections, stock market performance, etc. Organizations or companies that can accurately identify the timing, or "temporal state" of these civil unrests can obtain valuable time to prepare adequate responses.

Our initial research focused on utilizing a bag of words technique to identify temporal states from English articles from the EventStatus Corpus. On top of our baseline approach, we explored additional data engineering features and utilize Convolutional Neural Networks (CNN) in an attempt to improve the prediction accuracy.

1. Introduction

Our goal for the project is to build and train models that identifies temporal states of civil unrests described in news articles. Planning in several types of organizations can greatly benefit from this work. Health organizations need to plan staff and supplies to respond to humanitarian aid needs. International organizations need plan product production in region where demand is greatly reduced by civil unrest.

The events that can create humanitarian needs or decrease in product demands are often discussed in the news. The discussion can be after the event, during the event, and often before the event. Organizations that can tap into this wealth of information can plan better and minimize disruption. Organizations that can identify temporal states of civil unrests organizations precious additional time to be proactive instead of reactive.

2. Background

Our project explores and extends around two key concepts: relation extraction and temporal status. The work from other researchers provided crucial understanding on both the progress and the challenges around those. Llorens' work on temporal expression extraction and annotation gave us an understanding macro-steps to approach temporal state classification **[1]**, Alfonso's work on temporal reasoning shed light on nuances around temporal expressions **[2]**, and Bejan on the EventCore Bank added perspective on the creation of annotated temporal status corpuses **[3]**.

Through the above research, we've identified a few key challenges to temporal state classification. First, compositional references to time are not purely related to just verb tense utilized in language as one might first surmise. Secondly, events of different interest level are described in a variety of ways, making classification difficult. Finally, classification of time can vary greatly in word and phrase length, making the capture of the optimal text length highly variable. With these challenges in mind, we focused our feature engineering efforts to explore context extraction in order to improve temporal state classification .

Another key challenge highlighted by almost all research on the topic is the highly manual effort to annotating texts for temporal state classification. As a result, we did not attempt to annotate our own corpus for training & testing. Instead, we focused on one specific corpus with quality & detailed annotations to build our model around. The select corpus is the EventStatus corpus **[4]** that was made with the specific focus to help train models to identify event temporal status. They utilized a variety of methodologies to annotate articles related to civil unrest events with 6 temporal state statuses. The researchers also utilized SVM and CNN models test classification accuracy. We incorporated other language features such as subwords to build upon their work. Instead of classifying all future events as a single class, our models aimed to distinguish between "Future Planned", "Future Alert" and "Future Possible".

3. Methods

3.1 Data:

As described above, our main data source is the EventStatus corpus **[4]** consisting of 4,500 articles that have partially annotated temporal states for events in each article. The corpus includes multiple languages, with ~3,000 articles in English and the remaining in Spanish. We have focused our efforts on the English annotated files.

The annotated temporal states are Past, Ongoing, and Future. The future state has an additional annotations to distinguish additional context in "Future Planned", "Future Alert" and "Future Possible". The corpus provided a rich baseline of articles with event mentions to build our models.

Link:
http://digitalassets.lib.berkeley.edu/UCBonly/ldc/2017T09/

3.2 Data Modeling:

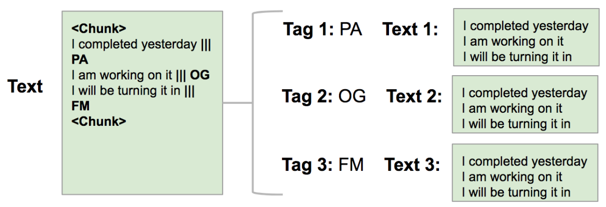
a. **Word model:** To prepare the corpus for training, we utilized the nltk packages to process the text. We tokenized the words in the annotated files and built customized functions to pick up chunks of words in the files that are annotated. (Although not used in the final data modeling code, we adopted the code base from Week 2 Materials to perform initial Exploratory Data Analysis on the corpus **[5]**.)

b. **Subword model:** To better capture morphological features such as tenses that exist within a word, we also created a sub-word model where words are broken down into sub-words at vowels and spaces. Our code logic also ensures that the subwords are of length two at the very least, except the cases when the word itself is of length one, e.g., "I", "a", etc. Although we are focusing on the English annotated files, this approach will work for both English and Spanish texts given the similar character compilation for both languages.

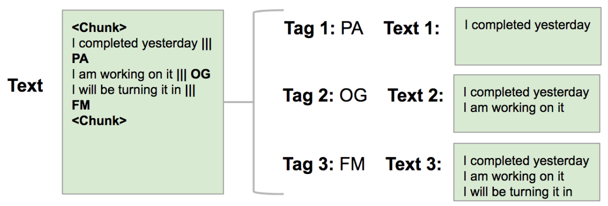
The outputted word and subword models served as our foundation to build features to capture temporal status context in a variety of ways as described below.

3.3 Features:

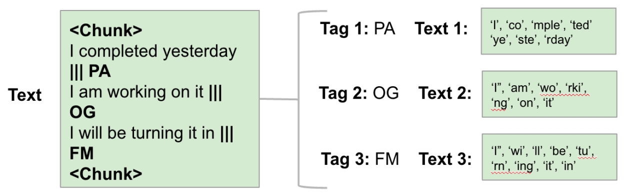
a. **Chunk to multiple annotations**: Our initial approach is to capture the maximum amount of text that could potentially contribute to the given temporal state of the article. To do this, we associate an entire text chunk (a section of text with multiple sentences and potentially multiple temporal state annotations) to each of the temporal state that are annotated within the text chunk per the example below



b. Accumulated phrase to single annotation: Our second approach is to accumulate text to a specific annotation within the text chunk, with the idea that the probability of the text immediately following an annotation tag would not be as applicable as the text preceding it.



c. Phrase to single annotation: Our third approach paired the annotation with only the phrase directly preceding it. Our goal is test to see whether the direct phrase without the non-helpful context in additional phrases will outweigh the helpful context in the additional phrases.



Text
<Chunk>
I completed yesterday ||| PA
I am working on it ||| OG
I will be turning it in ||| FM
<Chunk>
Tag 1: PA
Text 1:
'I', 'co', 'mple', 'ted' 'ye', 'ste', 'rday'
Tag 2: OG
Text 2:
'I", 'am', 'wo', 'rki', 'ng', 'on', 'it'
Tag 3: FM
Text 3:
'I", 'wi', 'll', 'be', 'tu', 'rn', 'ing', 'it', 'in'

Code: https://github.com/vslchumids/Event_Temporal_Classification/tree/master/code

3.4 Models:

Bag of Words + Random Forest: We built the baseline model using the bag of words technique. Given that bag of words and random forest and good models to use for classification tasks with smaller corpuses, we utilize this as our first model. Adopting ideas and code from a Kaggle movie classification competition **[6]**, we generated a set of Bag of Words features using the CountVectorizer class of the sklearn package. We then trained a Random Forest classifier to determine the temporal state annotation to the accompanying text tokens from either the word or the subword model. Finally, we trained the model utilizing the various features described above to classify the temporal state annotation.

Convolutional Neural Network (CNN): CNN is a modeling approach utilized in a few of the research on temporal state classification we referenced. We wanted to combine this model with features we have built to explore their subsequent impacts. We implemented a CNN for text classification model using TensorFlow adopting the approach and corresponding code snippets from a WILDML article by Danny Britz **[7]** and influenced by Kim Yoon's work on CNN for sentence classification **[8]**.[1]

To utilize the CNN model, we reprocessed our corpus data to fit CNN as opposed to bag of words. We also removed stopwords and numbers replaced with <NUM>. We also padded phrases to make all lengths equal, and built a vocabulary index that mapped words to integers. We opted not to use pre-trained embeddings. Instead, we built the model to learn from scratch.

In addition, the CNN model underwent a number of iterations to tweak the hyperparameters. First, we tweaked the number of filters from 50, to 100, and finally to 128. These tweaks improved the classification accuracy. We also tweaked the embedding dimensions from 50 to 100 to 150; however, negligible impact was made on the testing

[1] Our NLPCNN class was modified from the TextCNN class in the text_cnn.py file. Logic and code snippets from the data_helpers.py and train.py files were adopted to our code in societal_events_cnn.ipynb to construct, train and execute the CNN network to identify temporal states of the annotated files from the EventStatus corpus [9].

results. We also noticed that the metrics improved when no dropout is adopted (i.e., changing dropout keep probability from 0.5 to 1), however this might have caused overfitting as there would be no regularization on the convolutional neural networks without any dropout..

Code:
https://github.com/vslchumids/Event_Temporal_Classification/tree/master/code

3.5 Implementation:

We implemented our training and testing across the various techniques described above: 2 word models (word and subword) x 3 features (chunk to multiple annotation, accumulated text to annotation, and phrase to single annotation) x 2 learning modelings (bag of words + random forest and CNN).

Results

The results for the scenarios stemming from the above combination of techniques are listed below:

Bag of Words with Random Forest

Ver	Word Model	Feature	Prec	Recall	F1 Score
Base -line	Word	Chunk to multiple annotation	0.66	0.72	0.66
V1 - best	**Word**	**Accum phrase to single annotation**	**0.70**	**0.73**	**0.70**
V2	Word	Phrase to single annotation	0.69	0.72	0.70
V3	Subword	Chunk to multiple annotation	0.59	0.67	0.62
V4	Subword	Accum phrase to single annotation	0.67	0.69	0.65
V5	Subword	Phrase to single annotation	0.65	0.68	0.65

CNN
(Hyperparameters: Filter sizes = 3, 4, 5;
Number of filters = 128;
Embedding dimension = 50;
No dropout)

Ver	Word Model	Feature	Prec	Recall	F1 Score
V6	Word	Chunk to multiple annotations	0.65	0.68	0.65
V7	Word	Accum phrase to single annotation	0.63	0.65	0.63
V8	Word	Phrase to single annotation	0.59	0.64	0.61
V9	Subword	Chunk to multiple annotations	0.63	0.68	0.65
V10	Subword	Accum phrase to single annotation	0.67	0.71	0.68
V11	Subword	Phrase to single annotation	0.66	0.69	0.67

The Bag-of-words + Random Forest scenario that utilized whole word tokenizing and accumulated phrase to single annotation rose to the top with the best results, with F1 Score of 0.70, Precision of 0.70, and Recall of 0.73. The Bag-of-words + Random Forest scenario that utilized whole word tokenizing and non-accumulated phrase to single annotation came to an extremely close second, with F1 Score of 0.70, Precision of 0.69, and Recall of 0.72. The other word level Bag-of-words + Random Forest scenario obviously did not perform as well, probably because of confusion caused by multiple annotations being paired with the exact same chunk of words.

We were interested to see if the subword model would improve upon the word tokenization model by capturing tense indicators that will improve the classification score as described in Mikolov's Subword language modeling with Neural networks **[10]**. However, the subword model returned lower F1 scores and recall than our word models for the top scoring scenarios. Given that Mikolov's research focused on tasks other than temporal status classification, the subword model may be less proven in this application. Additionally, the small corpus may have inhibited the desired outcome of capturing tense related context from being successful.

Utilizing phrase to single annotation also produced better results than compared to the chunk to multi-annotation and the accumulated text to annotation features. We believe this may be due to the confusion brought on by the additional text in the latter two techniques. While both the chunk to multi-annotation and the accumulated text to annotations captured additional context that may point towards a temporal state, these benefits were outweighed by overlaps that captured key texts that point to other temporal states, thus increasing classification errors.

Finally, our implementation of the CNN model did not outperform our baseline model choice. Given CNN's many parameters and our small corpus **[11]**, CNN classification can be prone to perform poorly due to overfitting. Having that said, the CNN models that utilized accumulated and non-accumulated phrase to single annotation pairing and operated on subwords resulted in the third and fourth highest F1 Score of 0.68 and 0.67, just 0.02 and 0.03 points away from the F1 Score of the top two performing Bag-of-words + Random Forest models.

Code:
https://github.com/vslchumids/Event_Temporal_Classification/tree/master/code

Conclusion

The project results reinforced the difficulty in capturing the large variation in temporal state context within text. The results showed small variations to our baseline performance despite application of different features. Secondly, the the limited availability of well-annotated corpuses and the time consuming process to self-annotate really impacted the effectiveness for models such as convolutional neural networks. For future tasks on temporal status classification, training on annotated corpuses focusing on different types of events that of the EventStatus corpus can provide alternative ways to train on different temporal states contexts.

Another challenge is for the academic community to find a way to scale the annotation process for temporal statuses. The process laid out by the EventStatus team depended on a hybrid of algorithm based annotation combined with manual review. Having a proven and consistent methodology to apply annotation on more publications would drive great improvements in the task of temporal state classification. In terms of modeling, using word embeddings trained with other corpuses relevant to news articles could enhance accuracy and robustness of the CNN model.

Citations

1. Marc Verhagen, Robert Gaizauskas, Mark Hepple, Frank Schilder, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In Proceedings of the 4th International Workshop on Semantic Evaluations, pages 75– 80, Prague. <https://arxiv.org/pdf/1206.5333.pdf>
2. Alfonso Gerevini, Lenhart Schubert, and Stephanie Schaeffer. 1993. Temporal reasoning in Timegraph I–II. SIGART Bulletin, 4(3):21–25. <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=2C8095DFE8A7EF2A576B297912F65839?doi=10.1.1.98.8116&rep=rep1&type=pdf>
3. C. Bejan and S. Harabagiu. 2008. A Linguistic Resource for Discovering Event Structures and Resolving Event Coreference. In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC). <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.176.1107&rep=rep1&type=pdf>

4. Huang, R. Cases, I. Jurafsky, D. Condoravdi, C. Riloff E. Distinguishing Past, On-going, and Future Events: The EventStatus Corpus. <https://nlp.stanford.edu/pubs/huang2016events.pdf>
5. Language Modeling Notebook, W266 Week 2 Material. <w266/materials/week2/lm1.ipynb>
6. Bag of Words meets Bag of Popcorn, kaggle.com, 2014. <https://www.kaggle.com/c/word2vec-nlp-tutorial>
7. Danny Britz. Implementing a CNN for text classification in Tensorflow. <http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/>
8. Kim Yoon. Convolutional Neural Network for Sentence Classification. <https://arxiv.org/abs/1408.5882>
9. Danny Britz. cnn-text-classification-tf, Github. <https://github.com/dennybritz/cnn-text-classification-tf>
10. Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, Jan Cernocky. Subword language modeling with neural networks. <http://www.fit.vutbr.cz/~imikolov/rnnlm/char.pdf>
11. Pouya Ahmadvand, Reza Ebrahimpour, Payam Ahmadvand. How Popular CNNs Perform in Real Applications of Facial Recognition. <https://www.researchgate.net/publication/312463511_How_Popular_CNNs_Perform_in_Real_Applications_of_Face_Recognition>