**Justificatory Claims in Open Source Mailing Lists**

Chalenge Masekera, Daniel Griffin, Pi-­Tan Hu

**Project Goals**

**Original Intent**

The goals of this project were to discover how authors of messages on open source mailing lists make justification claims by use of an unsupervised approach to recognize justificatory discourse relations. This approach builds off Rhetorical Structure Theory and follows the approach of Marcu and Echihabi’s 2002 paper: An Unsupervised Approach to Recognizing Discourse Relations.

**Progress**

We were able to extract text from emails by using the BigBang toolkit and extract justificatory claim text spans with a cue phrase extraction. We then attempted to follow Marcu and Echihabi but discovered we likely needed a much larger coded data set to evaluate a classifier. We shifted our approach from identifying the justificatory claims to identifying types of such claims first extracted through the cue phrase patterns. We hand coded six categories of justification claim and developed a Naïve Bayes classifier. With the small hand-coded data set (n = 403) we were able to reach a proof of concept with a ~33% accuracy with k-fold cross validation in differentiating among the six possible classifications. We also developed an SVM classifier that we were able to develop to a similar performance level.

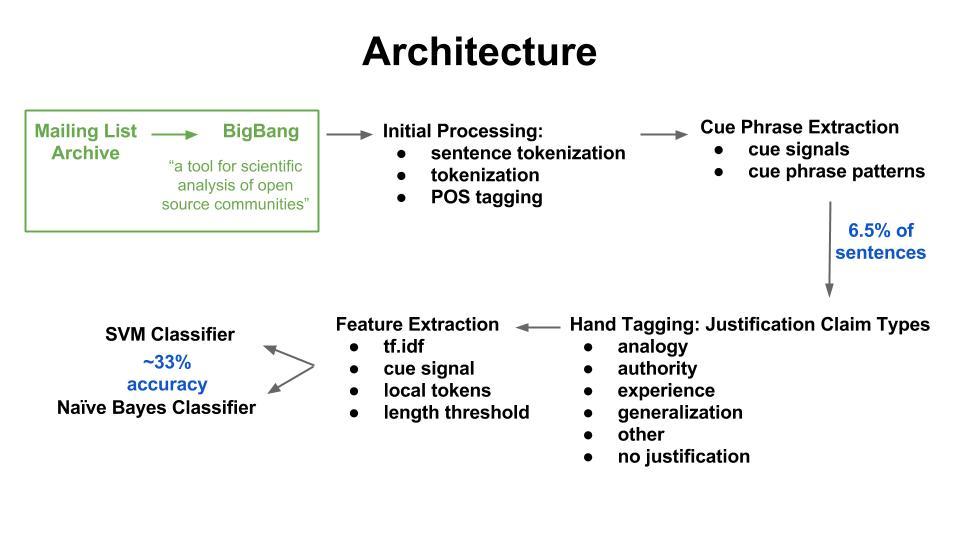
**Future Work**

There would be much benefit in being able to better clean these emails. The emails on the open source mailing we examined were generally very technical in nature, posing unique challenges. Improvement in isolating code in messages, both in code blocks and in-line, and in better sentence tokenization for this sort of data set would be useful. Future work would also require a much larger tagged data set to better train and evaluate the classifier. Testing strategies built off a tagged RST corpus such as the Penn Discourse Treebank would likely prove useful.

Additional work that might improve our results includes the application of more rigorous standards and distinctions between the types of justification claims that we hand tagged. More hand tagged samples would also likely have proved useful. Lastly, there may be some benefit of identifying and extracting features to support identification of specific claim types. A tighter specification for a human tagger would likely support efforts to extract features that may be useful in distinguishing one particular justification claim type.

With distinct justificatory claims types identified, it may be possible to classify members of open source communities and the communities themselves by the types of claims they make. The various types may be a useful metric in attempting to distinguish positions of power or influence within communities. Predicting the longevity or future volume of messages sent by a particular member may be possible. It may be possible to use preponderance of particular types of justification as a proxy for discourse quality, perhaps predicting the health of a community. Could a tool successfully identifying claim types help improve the quality of discourse in a community?

**Architecture**



**Data**

We used messages sent in the 'IPython-dev', 'IPython-user', 'SciPy-dev', 'SciPy-user’, and 'Wiki-research-l' open source mailing lists. We ended up using the last for our final classifier due to the cleaner text within emails. An IPython Notebook from BigBang was adapted to download and extract the messages. BigBang, maintained by Sebastian Benthall, is an open source toolkit being developed at the UC Berkeley D-Lab to support scientific analysis of open source communities. Quoted text from replied-to messages was removed from each message. Messages were then sentence tokenized with the pre-trained Punkt Sentence Tokenizer from NLTK. Then all sentences were processed through a function that tokenized individual words with NLTK’s regexp\_tokenize, based off a custom regular expression pattern, and tagged part of speech with a backoff tagger. The messages were then processed through our cue phrase pattern extraction, described below and the text spans extracted therein were then processed to be hand-coded. The hand-coding tagged each text span with one of six justification claim types.

**Algorithms**

**BigBang Mailing List Message Extraction**

This was a modification of an IPython notebook available in BigBang. The key aspect here was the extraction of the messages to a CSV file to facilitate later NLP processing.

**Cue Phrase Pattern Extraction**

Following Marcu and Echihabi, we identified patterns that justificatory claims seemed to follow and then used those patterns to extract justificatory text spans from the messages.

Patterns:

**[BOS justification EOS] [BOS So, claim EOS]**

**[BOS claim], [per justification EOS]**

**[BOS justification EOS] [BOS Because of this, claim EOS]**

**[BOS Because justification, claim EOS]**

**[BOS claim, because justification EOS]**

**[BOS claim because justification EOS]**

**[BOS Since justification, claim EOS]**

**[BOS justification EOS] [BOS Therefore, claim EOS]**

**[BOS justification EOS] [BOS Consequently, claim EOS]**

Text spans identified by these patterns, and any including the cue signals themselves (so, per, because, since, therefore, and consequently), were then extracted and hand-coded.

**Hand-coding**

We developed a web platform to efficiently tag the data that we extracted from the cue phrase patterns - adapting code from a prior project. The purpose for the hand-coding was to generate a training set which we would use for the supervised learning. Each sentence extracted through the cue phrase extraction was tagged with one of the six possible types of justification claim which are (based on an article by Gray (2010):

1. Appeal to Authority
2. Argument from Analogy
3. Generalization
4. Personal Experience
5. Other / Combination
6. No Justification

The lack of clear boundaries between the types of justification made tagging particularly hard and delicate. In retrospect, less types and definitive types of claims could have been useful and would have helped in making the classifier more accurate. After tagging the sentences were saved to a file and was used to train the classifier.

**Feature Extraction**

tf.idf pos and tuple : tuple is (word, pos) pair - This performs tf.idf on all parts of speech in the text spans, and separately on the word and part of speech tuple pairs.

cue signal (first, second set, last, index) - This was a group of four features that identified the first cue signal appearing in the two text spans, the set of cue signals appearing in the second text span, the last cue signal appearing in the text spans, and the index of the first cue signal in the second text span.

leng - This created a binary feature, whether or not the length of the second text span was longer than 90 tokens.

local tokens - This created features from the two tokens immediately following the cue signal within the text span.

The features below were tested, but were not shown to improve accuracy:

* Cartesian product of two text-spans - This created every possible pair of tokens between the two text spans, following the example of Marcu and Echihabi.
* Number of words of two text-spans
* Number of unique words of two text-spans
* TF-IDF (word) - The tf.idf of the words in the text spans did not perform as well as the tuple token and part of speech pair tf.idf.
* Bigram TF-IDF (word, pos, tuple)
* Has CD pos tag
* POS cue signal
* First POS cue signal
* Pronouns
* Verbs
* Most common POS

**Results & Evaluation**

We evaluated the performance of our Naïve Bayes classifier and our SVM classifier with k-fold Cross-Validation. We also examined both the most informative features for the classifier and looked at wrong guesses to see the actual text spans to help identify other features that may be useful. For the latter we did adapt code from the NLTK book. Between the six categories we were able to achieve an average accuracy of ~33%. After transforming our features and training an SVM classifier, we achieved similar performance.

Some of the successful results from our demo during the showcase are below:

|  |
| --- |
| Test sentence: Since this text is dirty, NLP is hard.  ----------------------  Output  ----------------------  Found ANALOGY justification  Test sentence: This should work because I said it should.  ----------------------  Output  ----------------------  Found AUTHORITY justification  Test sentence: I shouldn't need an IRB because this is a harmless study.  ----------------------  Output  ----------------------  Found GENERALIZATION justification |

**Contributions of Each Team Member**

Extraction: Daniel

Initial Processing: Daniel

Tokenizing and POS-tagging: Chalenge

Cue Phrase Pattern Extraction: Initially Chalenge, significant adjustment by all

Reaggregating in preparation for classifier: Initially Pi-Tan

Initial cartesian product featurization: Daniel (following Marcu and Echihabi, turned out not to be useful)

Code adapted to hand-tag sentences: Chalenge

Transforming hand-tag JSON to Python format for classifier: Daniel

tf.idf feature extraction: Pi-Tan and Daniel

Primary responsibility for merging different feature extraction functions: Pi-Tan

SVM classifier: Pi-Tan

**Supporting Materials**

**Code**

Justificatory Claims - 1 - BigBang Extraction

Justificatory Claims - 2 - Tokenize and POS

Justificatory Claims - 3 - Training Extraction

Justificatory Claims - 4 - Classifier

Hand-coding code

**Data**

wikimedia\_data: This contains the mailing list data extracted through BigBang and all the output from the first notebook, and the saved POS tagger.

**Presentation**

* demo - IPython notebook allowing users to test a sample claim.
  + demo\_data - input and output from the demo.
* slides - Diagram showing the architecture of the project from BigBang extraction through cue phrase extraction, hand-coding, and classifiers, a listing of cue phrase patterns used to identify text spans to be hand-coded, and the types of justification claims hand-coded..

**Works Cited**

Benthall, Sebastian. "BigBang." <http://sbenthall.github.io/bigbang/>, 2014.

Gray, J. S. "Four Types of Justification." *Ethical Realism*. Ethical Realism, 25 Feb. 2010. Web. 18 Dec. 2014.

Marcu, Daniel, and Abdessamad Echihabi. "An unsupervised approach to recognizing discourse relations." *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2002.

Pitler, Emily, and Ani Nenkova. "Using syntax to disambiguate explicit discourse connectives in text." *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers. Association for Computational Linguistics*, 2009.