Cross-referencing Claims in Large Corpora

Reza Akhtar, Andrew Costenoble, Kevin Fausett
CS 4120/6120
akhtar.r@husky.neu.edu
fausett.k@husky.neu.edu
costenoble.a@husky.neu.edu

1 Introduction

With the prevalence of deliberate misinformation, or "fake news," it has become increasingly difficult to accept articles at face value. One common and effective approach to gauging the veracity of a claim is cross-referencing the claim against other sources. However, this can be a laborious process. We propose a system which allows a user to test a claim against a set of articles and determine which, if any, assert an equivalent claim.

To do so, the plain text of an article is passed through a loosely-coupled three stage pipeline, compatible with multiple parsers and multiple comparators. It is syntactically parsed to receive a constituency tree. Next, we utilize a novel process to logically parse the tree to identify all of its constituent claims. The claims are then tested for semantic equality against the target claim: if any claim is equivalent, we assert that the article makes the claim. If none are equivalent, we assert that it does not.

2 Related Work

Identifying claims within an arbitrary context is a serious challenge, and one that many researchers are actively pursuing. There is no consensus on how to do this, as most claim detection research until recently has focused on specific domains and used semantic features such as tone. M. Lippi and P. Torroni introduced a method to identify a claim based on its rhetorical structure (P. Torroni, 2015). They achieved a high level of accuracy, but were still unable to match context-sensitive methods (M. Lippi, 2015). We hypothesized that our definition of a logical proposition, based solely on the syntactic structure of the sentence, would pull claims appropriately from different contexts and provide a solid framework on which to base our claim comparison.

The most important theoretical notion on which our definition of a logical proposition is founded is *phasehood*. A phase is defined as a syntactic object (SO) that is mapped to the sensory-motor and conceptual-intentional interfaces (N. Chomsky, 2005). This mapping process is most often referred to as Spell Out, and, when an SO is spelled out, only its edge is available for syntactic operations while the rest of the phase becomes opaque (no longer accessible for such operations).

This process of Spell Out is where the linearization of parse tree occurs, and, naturally, it has followed that phases are closely linked to units of logic. While phasehood is still hotly debated in contemporary literature on minimalist syntax, the most widely agreed upon properties of phase heads give rise to our precise definition of a logical proposition.

We sought, then, to leverage this definition of a logical proposition and its underlying rationale to isolate logic within raw text.

For the claim comparison, we compared word2vec sentence averages and the Universal Sentence encoder. Google research proposed that the Universal Sentence Encoder would be more effective than word2vec or alternative word-level models for tasks such as sentence similarity due to its ability to capture long range dependencies (D. Cer et al., 2018). Thus, we used it as a state of the art for comparison with our logical comparator and the word2vec baseline. Although it did perform best on sentence similarity, it did not transfer to claim equality as well as we hoped.

3 Dataset

In order to test our claim comparison, we came up with three claims in different topics: the Brazilian presidential election, climate change, and basketball. For each claim we assembled fifteen articles within that field and labeled them either true or false, according to whether that article contains a sentence asserting the overall claim. This set was used exclusively for testing.

There were two components which required training data. The first was our syntax parser which was trained on 200 Wall Street Journal articles from the Penn Treebank. These files contained annotated syntax trees from which our parser could induce a grammar. The second was the sentence similarity comparator. Our sentence comparator similarity predictions made use of the STS Benchmark set, which consists of pairs of sentences and a human-determined score of sentence similarity.

We compared (sentence average) word2vec and Universal Sentence Encoder's performance at the sentence similarity task using cosine similarity between sentence vectors. However, sentence similarity and claim equality aren't a perfect match; manually looking through the STS sentences, we found that at about a 4.6/5 similarity, their claims were usually equivalent. We determined the median cosine similarity via both methods of pairs in the training set with 4.6 or greater similarity and set that as a threshold for coercion. Values equal or above the threshold coerce to 1 for claim equality and below coerce to 0. This was then tested on the STS test set, having changed all 4.6+ scores to 1 and those below to 0.

4 Methodology

Our program takes two strings, a target claim and some text to evaluate, as its input. If the target claim is found to be in the text, it will output true; otherwise, it will output false, indicating the target claim was not agreed to by the text.

The program uses three discrete components to come to a conclusion about a text. First, the sentences of the text are syntactically parsed and a parse tree is obtained for every parsable sentence. Second, this parse tree is handed to the logic parser which extracts all logical propositions (or "claims") from the parse tree and outputs a linearized form of these tokens. Finally, these strings of claims are then taken by the

claim comparator for evaluation of the claims against the target claim.

4.1 Syntax Parser

During the first step, the model uses a premade open-source syntactic parser called *stat_parser* (Cer et al., 2018). This parser uses a probabilistic context-free grammar (PCFG) trained on the Wall Street Journal corpus from Penn Treebank. The parser uses this PCFG and the CKY parsing algorithm to parse the a given sentence string into a parse tree.

The CKY algorithm is a dynamic programming algorithm which calculates the highest probability subtree for every sequences of words in the sentence. Therefore, after the algorithm has calculated the probability of the most probable structure of each subtree, and the the structure and its probability can be deduced from these stored results.

The *stat_parser* parser uses a normalization process to convert the PCFG into a more concise set of production rules with predefined labels not seen in the training data. This had a great impact on the running time because the PCFG naturally induced from the training data had ~7600 production rules associated with it.

After observing some salient mistakes made by the *stat_parser* when parsing our test corpus, we attempted to produce an alternative parser which would utilize the Viterbi parsing algorithm and the same same PCFG.

The Viterbi algorithm works similarly to the CKY algorithm. However, we used the full set of production rules as the grammar, and this had multiple effects. The outputs had more accurate structure and labelling, but the set of rules was not parsimonious.

Therefore, although we implemented the algorithm, we were unable to optimize the PCFG such that the production rules were concise enough for the parser to run in a reasonable amount of time.

Thus, we used the CKY algorithm implemented in the *stat_parser* to obtain all of the parse trees of our training data.

4.2 Logic Parser

After syntactic parsing, the logic parser takes the parse tree from the syntactic parser as its input and conducts claim identification. During this process the logic parser delimits claims by inserting brackets into an array of leaves of the parse tree. The specification for a claim conforms to a novel syntactic definition of a logical proposition.

The definition is as follows: a parse tree represents a logical proposition (a claim) if and only if (a) its label is the start symbol of its grammar, (b) the phrase has tense (i.e. no infinitives), (c) the parse tree is not a sibling to a restrictive head. In the grammar given by stat_parser, the only start symbol is *S*, and heads of *SBAR* are assessed for restrictiveness. Restrictive heads denote the phrase's logical dependency on a superordinate node.

Logic parsing is an important step in this process because the extraction of claims from text resolves any convolution that may arise from logic embedding. For example, the target claim and input text in (1) agree; however, any sentence comparison method on its own would resolve that the two have a greater level of dissimilarity than a reasonable threshold.

(1) Target claim: *Michael Jordan is the greatest basketball player of all time*.
Input text: *Michael Jordan is the greatest basketball player of all time because he won all six of the championships he played in.*

The perceived dissimilarity occurs because the input text is more descriptive, adding an additional claim (that Jordan won all six of the championships he played in). The logic parsing mechanism resolves this by isolating the logic propositions so that the comparator can evaluate similarity piecewise.

For example, the input text in (1) would be parsed into three claims: the full sentence and the two subordinate sentences, "Michael Jordan is the greatest basketball player of all time" and "he won all six of the championships he played in." Thus, any good comparator should recognize that the target claim is present.

4.3 Claim Comparator

We used word2vec as a baseline due to its speed and well-rounded performance. We compared to it a cutting-edge solution in applying Google's Universal Sentence Encoder and a novel solution in our own logic based comparator.

4.4 word2vec-based comparison

We make use of the Google News pre-trained word2vec model, which contains embeddings for approximately 3,000,000 words. We word-tokenize every claim as well as the target claim. Tokens not contained in the model are changed to "UNK". We then convert every word in a sentence into its word2vec representation (a 300-dimensional vector) and take the average for each claim. Cosine similarity is calculated between each claim and the target claim; if any passes above a set threshold, we determine that article likely asserts an equivalent claim. Details on how that threshold is determined are described in section 5.3.

4.5 Universal Sentence Encoder-based comparison

Sentence vectors are compared in the same manner, although the threshold is set at a different point. We used Google's Universal Sentence Encoder (D. Cer et al.). It relies on transformation, using encoding similar to the first half of sequence to sequence learning. All words are converted to vectors, and a 512-dimensional sentence vector is generated by using an attention mechanism to weight each word relative to its importance to specific other words. It offers similarly large vocabulary to the Google News word2vec with the added benefit of greater sensitivity to long-range dependencies and word-ordering by virtue of the attention mechanism.We coerced from similarity to equality at a particular threshold separate from word2vec's threshold, as described in section 5.3.

4.6 Logical Comparison

Logical propositions often share similar syntactic structures; specifically, they are usually composed of a single noun phrase and a single verb phrase, each possibly with its own subordinate phrases. (Lippi, 2015) We sought to exploit this structure to provide a more robust way to compare claims. Intuitively, if two claims share a verb and each refer to the same entity in their subject and object, they likely represent

equivalent claims. The subject of a sentence was defined as the first noun phrase that is a child of a S label and sibling to a verb phrase. The object was defined as the first noun phrase that is a child of a verb phrase. The main verb was defined as the first verb in a verb phrase. (Bird, 2016) This is perhaps an overly simplistic model, but we hoped it would provide a baseline for further improvements. Unfortunately, for reasons explained further on, this model never produced viable results.

5 Experiments and Metrics

5.1 Claim Comparison

We used average word2vec, a fast and common solution, as a baseline by which to judge the performance of the more advanced Universal Sentence Encoder (USE) and our novel Logical Comparator. First, we had to evaluate how well word2vec and USE performed in sentence similarity by testing the Pearson correlation of their cosine similarity against the STS Test Set (1,379 sentences scored from 0 to 5 in similarity).

Method	Pearson Correlation
W2V	0.40298655
USE	0.67478752

The Universal Sentence Encoder has a stronger correlation with human judgment compared to the baseline; however, it takes time on the order of $O(n^2)$ compared to word2vec's O(n). Next, we needed to coerce from a similarity score in the continuous range from 0 to 1 to a binary score predicting equality. Manually examining the STS training set, we judged that at a score of about 4.6 is the point when most sentence pairs' meanings are equivalent. We collected all training sentences of similarity 4.6 or higher and determined the median cosine similarity of this subset with both methods.

Method	Threshold
W2V	0.8713053166866302
USE	0.9147900938987732

At cosine similarities equal or greater than these thresholds, the respective methods would coerce to 1, and for those below, 0.

To test the accuracy of the word2vec and USE, we took the STS Test set and coerced all pairs of 4.6+ similarity to 1 and the rest to 0. To process the data for the logical comparator, we parsed each sentence into its constituency tree and compared the pairs. We then evaluated the performance of our two tested methods and the baseline on identifying likely equivalent claims in this set.

Method	Precision	Recall	Balanced F Measure
W2V	0.5693	0.4096	0.4765
USE	0.6463	0.2453	0.3556
Logical	0.1667	0.0487	0.0707

The baseline proved to be the most effective overall. Universal Sentence Encoder's greater granularity in sentence encoder helped increase its precision ("The sandwich bit the dog" vs "The dog bit the sandwich" are equivalent in word2vec and different in Universal Sentence Encoder.) However, that same granularity likely reduced recall, leading to false negatives on equivalent claims with different structures. The logical comparator performed very poorly, however we believe most of its error was due to the inaccurate syntactic parser used to preprocess the data. In hand testing, it was found that changing even a single word to something that should be equivalent could produce a constituency tree that was significantly different, and often wrong. In many sentences it was found that the parse tree would not contain a subject or object at all, according to our definitions of such. Because the logical comparator in its current form is very sensitive to the structure of the sentence, it was unable to effectively compare many of the pairs of sentences. We had hoped that with a stronger syntactic parser the comparator would produce better results, but we were unable to implement one in the allotted time frame

5.2 End to End

We tested end to end on a gold-standard corpus of real news articles on three topics we collected.

Method	Recall	Precision
W2V	.05	1.0
USE	.1	1.0
Logical	0.0	0.0
Comparator		

This was a challenging test corpus as the articles in the same domain as a target claim were on the same subject (i.e., sports articles were all on the subject of who is the greatest basketball player of all time for the target claim of "Michael Jordan is the greatest basketball player of all time." We expected this to hurt precision, as there would be very close verbiage even in different claims on the same subject. In reality, the definitions of equality were stringent enough with each method that we received no false positives. However, recall was universally low. This is likely due to the propagation of a few sources of error. W2E and USE's performance suffered due to the necessity of coercion from cosine similarity to binary equality. The logical comparator had difficulty due to the unreliable parse trees it received. The former group would benefit from the existence of a more specific training set: one for semantic equality rather than similarity. The latter would improve performance with a more accurate syntax parser.

6 Conclusion

It is often difficult to identify what information on the internet is reliable. Readers may have to cross-reference several different sources, which is time-consuming and may require significant effort. Our goal was to make this process easier and more accurate, by creating a tool that could identify a claim within a corpus of documents. First we needed a way to identify claims, for which we used a novel definition of logical propositions. This could

determine how many claims a sentence represents, if any, and where in the sentence each claim is made. Our next task was to compare claims for equivalency. We used three methods for this: a baseline of word2vec sentence similarity, comparison using Universal Sentence Encoder (USE), and a logical comparator that exploits the syntactic structure of logical propositions. Neither of our methods outperformed the baseline overall. The USE comparison was highly specific, which caused it to have higher precision than the baseline, but as a result its recall and F measure suffered. To improve this, changes to the encoder could be made to make it more sensitive to sentence structure. The logical comparator performed significantly worse across the board, but we believe much of that was due to the syntactic parser we used. Future experiments would hopefully implement a more accurate syntax parser and could produce more viable results from the logical comparator. In addition, its definition of what makes two propositions logically equivalent could be made more robust against changes in sentence structure.

References

[1] D. Cer et al. *Universal Sentence Encoder,* (2018) https://arxiv.org/abs/1803.11175

[2] E. Monti, *pyStatParser* (2017), Github repository, https://github.com/emilmont/pyStatParser

[3] M. Lippi and P. Torroni, "*Context-Independent Claim Detection for Argument Mining*," IJCAI, 2015.

[4] S. Bird, *Natural language processing with python*. O'Reilly Media, 2016.

[5] Chomsky, Noam, *On Phases*. In *Foundational Issues in Linguistic Theory* (MIT), ed. by Robert Freidin, Carlos P. Otero, and Maria Luisa Zubizarreta, 133-166. 2005.