A Two-Phase Approach Towards Identifying Argument Structure in Natural Language

Arkanath Pathak
Deptt. Computer Sc & Engg.
IIT Kharagpur
pathak.arkanath@gmail.com

Pawan Goyal
Deptt. Computer Sc & Engg.
IIT Kharagpur
{pawang@cse,

Plaban Bhowmick
Centre for Educational Tech.
IIT Kharagpur
plaban@cet}.iitkgp.ernet.in

Abstract

We propose a new approach for extracting argument structure from natural language texts that contain an underlying argument. Our approach comprises of two phases: Score Assignment and Structure Prediction. The Score Assignment phase trains models to classify relations between argument units (Support, Attack or Neutral). To that end, different training strategies have been explored. We identify different linguistic and lexical features for training the classifiers. Through ablation study, we observe that our novel use of word-embedding features is most effective for this task. The Structure Prediction phase makes use of the scores from the Score Assignment phase to arrive at the optimal structure. We perform experiments on three argumentation datasets, namely, AraucariaDB, Debatepedia and Wikipedia. We also propose two baselines and observe that the proposed approach outperforms baseline systems for the final task of Structure Prediction.

1 Introduction

The problem of argumentation mining concerns the identification of argument structures in a text. The argument structure is typically represented as a directed graph with textual propositions as nodes and both Support and Attack relations as edges between the propositions. In their influential work, Mochales and Moens (2011) have discussed this problem in detail together with the relevant definitions, frameworks, and terminologies. They define the argumentation structure as consisting of various "arguments", forming a tree structure, where each argument consists of a single conclusion and one or more premise(s). Another widely used framework is the Freeman theory of argumentation structures (Freeman, 1991; Freeman, 2011), which treats an argument as a set of proponent or opponent propositions for a central claim. In the present study, we follow the framework used in (Mochales and Moens, 2011).

The full task of argumentation mining can be divided into four subtasks (Mochales and Moens, 2011):

1. **Segmentation**: Splitting the text into propositions.
2. **Detection**: Identifying the argumentative propositions.
3. **Classification**: Classifying the argumentative propositions into pre-defined classes (e.g. premise or conclusion in the Mochales and Moens' framework and proponent or opponent in the case of Freeman's framework).
4. **Structure Prediction**: Building the structure by identifying the relations (the edges in the argumentative graph structure) between the propositions.

Our work jointly tackles the Classification and Structure Prediction subtasks using a unified approach. Little work has been done as far as Structure Prediction is concerned. We are familiar with only two approaches that can be compared to our work. Lawrence et al. (2014) proposed to form bidirectional edges between propositions in their work on 19^{th} century philosophical texts. They used Euclidean

distance metric between topic measures derived from a generated topic model for the text to be studied. They achieved a raw accuracy of 33% for linking the edges. However, they do not form directed edges between the argumentation units, which is essential in case of arguments. Peldszus and Stede (2015) jointly predict different aspects of the argument structure. Recent works Persing and Ng (2016; Stab and Gurevych (2016) identified argument relations in the student essays. Discourse features (positional features) have been used in these studies However, absence of these features in argument graph dataset like AraucariaDB (used in our study) makes the relation classification task challenging.

This paper makes the following contributions. i). We propose a data-driven approach for identifying argument structure in natural language text. ii). We present a detailed study over various linguistic, structural and semantic features properties involved in the argument relation classification task. iii). Finally, we propose a metric for evaluating performance of the structure prediction task.

2 Problem Formulation

We have used the AraucariaDB (Reed and Rowe, 2004) dataset[1] to discuss the problem at hand. This corpus consists of 661 argument structures, collected and analysed as a part of a project at the University of Dundee (UK). We found AraucariaDB to be one of the most suitable argumentation datasets, primarily because it is formed from natural language resources like newspapers and magazines. Fig. 1 shows an argument structure from AraucariaDB. The edges in the tree represent a Support relation. For instance, Node 271 and Node 272 are the children of Node 270. Therefore, $271 \rightarrow 270$ as well as $272 \rightarrow 270$ are Support relations. For a given Support relation $n_1 \rightarrow n_2$, we call the node n_1 as the *Text* node and n_2 as the *Hypothesis* node. Our goal can formally be defined as follows:

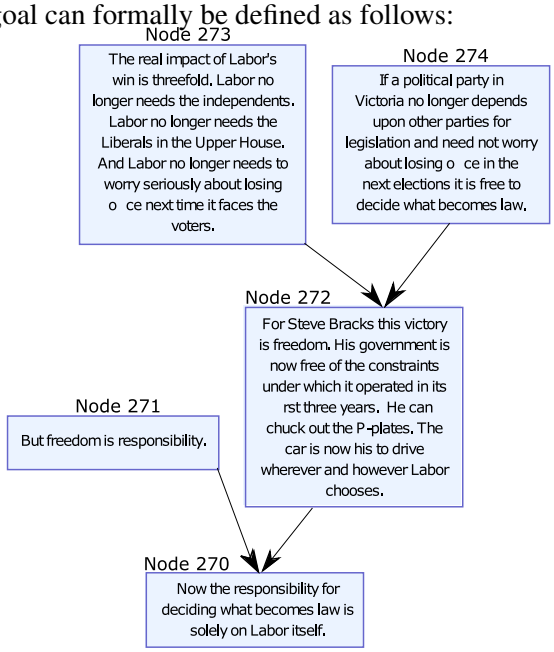


Figure 1: Sample argument (Argument No. 9) from the AraucariaDB.

For a given set of propositions, with an underlying argument structure connecting these propositions, identify the argument structure, that is structurally close to the corresponding gold standard argument.

A measure for structure similarity (which we call $SimScore$) is described in Section 5.2.

3 Proposed Approach

We model an argument structure as a graph, where each node represents a proposition. Given a set of nodes N for an argument, our approach can be divided into two subtasks:[2]

[1]The AraucariaDB dataset can be downloaded and visualized with the AIFdb (Lawrence et al., 2012) framework at http://www.arg.dundee.ac.uk/aif-corpora/araucaria.

[2]We assume only Support and Neutral relations between arguments, and the final structure to be a tree. In Section 6, we extend this model to include Attack relations as well as linear structures.

Score Assignment : Assign scores $s_{n_1,n_2} \in [0,1]$ for every pair of nodes $n_1, n_2 \in N$, $n_1 \neq n_2$,. These scores represent the degree of Support relation present in the hypothetical edge connecting n_1 and n_2.
Structure Prediction : Choose the tree T^* with the maximum sum of edge scores, i.e.,
$T^* = \underset{T}{\operatorname{argmax}} \sum_{(n_1,n_2)\in E(T)} s_{n_1,n_2}$

where T can be any tree with the set of nodes N and $E(T)$ denotes the set of edges in T. We use the confidence measures provided by machine learning classifiers as the edge scores in Score Assignment. Specifically, we use binary classification with the classes being *Support* and *Neutral*. The classifier takes as input an ordered pair (n_1,n_2), where n_1 and n_2 are the text nodes.

For Stucture Prediction, our implementation essentially iterates over all possible tree structures (for the given set of nodes) recursively to choose the best tree. A call to the recursive function will already have the tree structure formed upto the last level. The function iterates on all possible sets of nodes (subset of the set of remaining nodes) for the next level. The parent of each node in the next level can be identified (from the last level) as the node which gives rise to the best attachment score. The complexity of our implementation is exponential in the number of nodes in the argument. Therefore, we limit the experiments usually to arguments with 10-15 nodes.

4 Classifier Features

In this section, we describe the set of features chosen for the Score Assignment subtask. The task of Score Assignment is similar to Recognizing Textual Entailment (RTE) or the detection of Natural Language Inference (NLI). However, the problem is still considerably different in the case of arguments since the types of arguments can be much more complex (Walton, 2007). To make this more evident, we have formed a baseline for our experiments (Section 5) where we use a state-of-the-art RTE tool instead for the Score Assignment subtask. We experimented with the features frequently used in NLI, RTE and similar tasks. MacCartney (2009) discusses the features used for the NLI task in detail. However, some of the frequently used features like POS n-grams, the length of propositions, POS of the main verb, etc., are not included in the set of features because they showed insignificant effect on the overall performance in our experiments. We suspect this is due to the fact that many attributes of these features are already captured in the features we have chosen.

Discourse Markers: These are the words that are indicative of argumentative discourse. Discourse markers have persistently been used for both RTE and NLI tasks. Eckle-Kohler et al. (2015) have also discussed the role of discourse markers in the context of argumentation mining. However, we observed that the presence of such words are rare in the AraucariaDB dataset. We have used i). counts of the following words in Text: *as*, *or*, *and*, *roughly*, *then*, *since*, and ii). counts of the following words in Hypothesis: *therefore*, *however*, *though*, *but*, *quite*. This feature set gives rise to 11 (6+5) features.

Modal Features: These are similar to discourse markers but they do not inherently belong to either one of Text or Hypothesis. Therefore, we take the counts of these as features for both Text as well as Hypothesis inputs. We have used the following as the modal words: *can*, *could*, *may*, *might*, *must*, *will*, *would*, *should*. Modal feature set, therefore, gives rise to 16 (8×2) features.

Longest Common Phrase: The number of words in the longest phrase present in both Text and Hypothesis is considered as a single feature.

Common Wikipedia Entities: In many cases, a specific argument usually revolves around some conceptual entities. For example, Argument 9 (Fig.1) involves the entities *Steve Bracks*, *Labor*, etc. We have used TAGME (Ferragina and Scaiella, 2010) to annotate a text with Wikipedia entities. After we have the annotations as a vector for both Text and Hypothesis, we take the inner product of the resulting vectors as a single feature.

Word N-grams: Word n-grams are used very frequently as features for NLP tasks. We have used the set of unigrams and bigrams filtered by relative likelihood of presence in Text or Hypothesis nodes in the training data. For instance, the n-grams with higher values of $\frac{p(ngram|Text)}{p(ngram|Hypothesis)}$ will be chosen from Text nodes. The filtering is performed using a constant threshold parameter of the likelihood. We have set the threshold parameter as 3 for all of our experiments. Since we have performed Cross Validation, the number of features for this category will be different for each fold. The mean count for unigrams was

115.4 and that for bigrams was 251.8. Hence, an average of 734.4 (115.4×2 + 251.8×2) n-gram features were used over the 5 folds.

Word Vector Embeddings: Word vectors capture a variety of helpful information in the context of natural language. We have directly used the 300-dimensional vectors trained on part of the Google News dataset (Mikolov et al., 2013) [3]. We have used the sum of word vectors over words present in Text node to form a feature vector. To generate another feature vector, a similar process is repeated for Hypothesis node. These vectors are concatenated to give rise to 600 features for an input pair. Using word vectors as features can help with various attributes inherent to Support edges. First, a simple similarity measure can be the difference of the two sum vectors, which can be well captured by using classifiers like linear SVM. Secondly, word vectors trained over an external dataset like Google News can provide the knowledge base for language not present in training set, which is very likely in case of argumentation mining since the arguments are expected to be unrelated. Lastly, since word vectors are based on contextual information, they can infer Support relation from similar contexts in training data.

5 Experiments

All of the experiments in this section are performed on the AraucariaDB arguments. The experiments are performed in a 5-fold cross-validation framework and the mean scores are reported. The folds are over the set of arguments rather than the pairs of text nodes in order to maintain contextual independence between the folds. A subset of AraucariaDB arguments have been used in the following experiments[4].

Measure	type-1 SVM	type-2 SVM	type-2 MLP
$confidence_S$	0.691	0.643	0.678
$confidence_N$	0.425	0.356	0.306
$recall_S$	0.759	0.677	0.677
$recall_N$	0.532	0.681	0.692
$precision_S$	0.193	0.68	0.688
$precision_N$	0.937	0.678	0.681
$accuracy$	0.561	0.679	0.684

Table 1: Classifier Performance: mean values are reported for Support (S) and Neutral (N) relations.

5.1 Classifier Performance (Score Assignment)

Support edges present in the input argument structures are directly taken as Support pair examples for the classifier. However, the generation of Neutral pairs is not so straightforward. To that end, we have experimented with two kinds of frameworks. Please note that the selection of training framework does not affect the ultimate goal of structure prediction. The framework used for training the classifiers is only limited to the Score Assignment phase.

The first framework, which we call the *type-1* framework, considers all the pairs (n_1,n_2) as Neutral such that n_1 and n_2 are text nodes belonging to the same argument and $n_1 \rightarrow n_2$ is not a Support relation. This, however, gives rise to a huge imbalance between the Support and Neutral examples. Many classifiers fail to perform well in such imbalance. Nonetheless, this issue can be resolved in classifiers like linear SVM, by assigning class weights inversely proportional to class frequencies (King and Zeng, 2001) in the input data. We have followed this framework for type-1 SVM. Another way to resolve the problem of imbalance is to down-sample the Neutral relations randomly. However, random sampling did not perform well in our experiments, and thus, we posit that a random subset might not be a good training sample.

To counter this, we devised the *type-2* framework which only considers those pairs (n_1,n_2) as Neutral for which $n_2 \rightarrow n_1$ is a Support relation. This gives a perfectly balanced input dataset with one Neutral example corresponding to each Support example. Thereby, making it suitable for machine learning classifiers. It is difficult to compare the information captured by the two frameworks. While type-1

[3]The pre-trained word vectors for Google News dataset are freely available at https://code.google.com/archive/p/word2vec/.

[4]Due to exponential order complexity of Structure Prediction algorithm, we have selected arguments of size 10 nodes or less. Furthermore, arguments involving relations other than Support are ignored.

might seem to capture more information than type-2, type-1 is also prone to more noise since the data is larger as compared to type-2.

A Multi-layer Perceptron (MLP) classifier using the type-2 framework performed better than type-1 and type-2 SVM implementations for arguments with 3 nodes in our experiments. The network is made up of 3 hidden layers with 200 neurons for each layer.

Table 1 summarizes the results for the three classifiers. We have shown 7 performance measures for each classifier. Specifically, we present the mean of the confidence values provided by the classifier for each class label, which is used directly in the Structure Prediction phase. We have scaled the confidence measure[5] linearly between 0 and 1 before using it as scores for Structure Prediction. A mean confidence of 1, therefore, will be the perfect score for Support pairs. Similarly, a mean confidence of 0 will be the perfect score for Neutral pairs. We can observe that each classifier outperforms the others for at least some metric. One can observe that type-2 classifiers perform better in predicting Neutral pairs. The confidence measure is lower than type-1 and the recall is higher as well. However, the precision for Neutral is better for type-1 SVM because of the data imbalance. Type-2 MLP gave the best accuracy in our experiments.

5.2 Structure Prediction Performance

Since the arguments are complex in nature, our approach (Section 3) often fails to predict the entire structure. To counter this, we formulate a measure to evaluate the similarity between the predicted tree and the input tree. The measure, $SimScore$, is defined as:

$$SimScore(T_1, T_2) = \frac{|E(T_1) \cap E(T_2)|}{|E(T_1)|}$$

where T_1 and T_2 have the same set of nodes and $E(T)$ is the set of edges for a tree T. This measure quite intuitively captures the fraction of edges common to both trees. Since the set of nodes are the same, this measure turns out to be directly related to measures like the graph edit distance (Sanfeliu and Fu, 1983).

Since our problem formulation is new, it is difficult to compare the results with existing literature for Structure Prediction in argumentation mining. However, we compare our performance to two baselines. The first baseline, **Random**, is a baseline which randomly chooses any tree structure over the given set of nodes. It can be shown that the expected value of the $SimScore(T_i, T)$ for a given tree T is equal to $1/n$ where n is the number of nodes in T. For the second baseline, **EDITS**, we use the state-of-the-art RTE software package EDITS (Kouylekov and Negri, 2010), instead of the classifiers we proposed, for scoring the edges. However, the metric for scoring structures remains the same as the sum of edge scores. In this case, the entailment relation corresponds to Support relation. EDITS has also been used previously by Cabrio and Villata (2012) in the context of argumentation mining.

Nodes	**Arguments**	$SimScore$				
		type-1 SVM	**type-2 SVM**	**type-2 MLP**	**EDITS**	**Random**
2	10	**0.9**	0.8	0.8	0.7	0.5
3	187	0.564	0.566	**0.625**	0.363	0.333
4	85	0.529	**0.552**	0.482	0.250	0.250
5	62	**0.446**	0.399	0.435	0.231	0.2
6	72	**0.363**	0.341	0.322	0.263	0.166
7	58	**0.369**	0.323	0.309	0.231	0.142
8	41	**0.230**	0.19	0.199	0.205	0.125
9	19	**0.351**	0.28	0.222	0.265	0.111
10	23	**0.217**	0.188	0.115	0.212	0.1
Any	557	**0.459**	0.442	0.447	0.289	0.234

Table 2: Structure Prediction Performance: Mean of $SimScore$ for the arguments grouped by the number of nodes in the argument. EDITS and Random are baselines whereas type-1 SVM, type-2 SVM and type-2 MLP are proposed approaches.

[5]We have used the SVM and Multi-layer Perceptron classifier implementations provided by the open source library scikit-learn (Pedregosa et al., 2011). For the confidence measure, we have used the decision function in the case of SVM and the predicted probability in the case of MLP. Class imbalance was handled using 'balanced' weighting of classes.

In Table 2, we compare the mean value of $SimScore$ for each classifier. We have further categorized the results based on the number of nodes present in the argument. As evident by the Random baseline, it is expected that the performance will degrade as the number of nodes increase. We can observe that all the three classifiers outperform the EDITS and Random baselines by a considerable factor. For the arguments with 3 nodes, type-2 MLP outperforms type-1 SVM and type-2 SVM. However, for arguments with higher number of nodes, type-1 SVM performs the best. The results reported are statistically significant with a p-value of 0.00198 after performing a two-tailed paired t-test between the type-1 SVM and the EDITS baseline.

5.3 Ablation Study

To test the efficacy of each individual feature/feature group, we have performed a leave-one-out ablation test. In the second column of Table 3, we report the % decrease in the mean SimScore (for any number of nodes) when the type-1 SVM classifier is used. Following observations are made from this study.

- Discourse markers and modal features are observed to be the least effective feature groups.
- Word vectors trained on an external knowledge base are highly effective.

Feature Set	**% decrease in $SimScore$**	
	With Word Vectors	**Without Word Vectors**
Discourse Markers	0.09%	0.21%
Modal Features	0.27%	0.37%
Wikipedia Similarity	0.59%	0.91%
Word N-grams	1.56%	**21.14%**
Word Vectors	**11.4%**	-
Longest Common Phrase	1.02%	1.22%

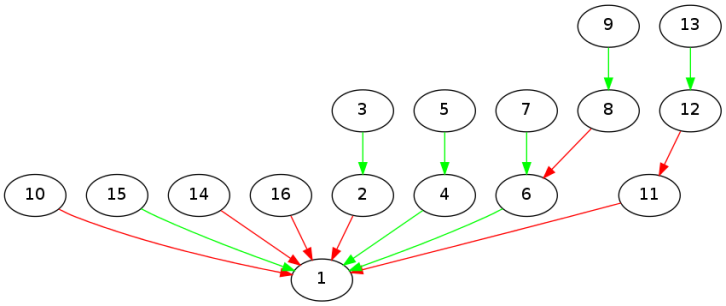
Table 3: Ablation Study: The decrease in Structure Prediction performance due to the removal of each kind of feature. The second column corresponds to the experiment with word vectors feature set. The third column corresponds to the experiment in the absence of word vectors feature set.

We conjecture that word vectors encode much more information than n-grams and other linguistic features for Score Assignment. To support this conjecture, we perform another ablation test to judge the effectiveness of other features in absence of word vector feature group. The results are reported in the third column of Table 3. We observe that word n-grams are now influential with a decrease of 21.14%, which was not the case when word vectors were present. Therefore, we can deduce that word vectors were able to capture word n-grams to a great extent. However, discourse markers and modal features are still not very influential. Discourse markers assume 3.21% in the set of terms in AraucariaDB while the modal features are present with 1.89%. We hypothesize the ineffectiveness of discourse markers and modal features to their rarity in AraucariaDB.

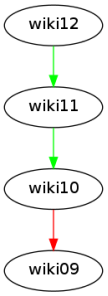
6 Arguments with Attack relations

Till now, we have only considered arguments which solely include Support relations. A natural extension of this approach is to support the arguments which include both Support and Attack relations. In an Attack relation, $A \rightarrow B$, statement in node A is used to contradict the statement of node B. We could not find enough argumentation datasets which include attack relations in a significant proportion. We have used two datasets, namely, *Debatepedia* and *Wikipedia* from NoDE (Cabrio and Villata, 2014)[6], a benchmark of natural argument. Although these datasets are pretty small for a machine learning approach, our approach still performs reasonably well for these datasets. The first dataset, *Debatepedia*, consists of data extracted from online debate platforms (debatepedia.org and procon.org). This dataset consists of 260 (140 Support, 120 Attack) relations. Each debate is formed by the responses for a given topic. Fig. 2a shows an example debate from the dataset for the topic of "Violent Games". We will treat such a structure for a given topic as an argument. There are 22 such topics in the dataset. We ignore one topic: "Ground Zero Mosque", because it does not follow a tree structure. The second dataset is built

[6]These datasets are described in detail, and are freely available for download, at http://www-sop.inria.fr/NoDE/NoDE-xml.html.



(a) An example debate structure from Debatepedia for the topic of Violent Games.



(b) An example argument from the Wikipedia dataset.

Figure 2: Argument structures with Attack relations. Green edges indicate Support relation whereas red edges indicate Attack relation.

on the Wikipedia revision history over a four-year period, focusing on the top five most revised articles. The Wikipedia dataset consists of 452 pairs (215 Support, 237 Attack). We consider the structure formed by the revision history to be an argument. Therefore, each argument will be a linear graph, as shown in Fig. 2b. To extend our approach for decoding the best possible tree, we need to modify the algorithm to accommodate the fact that the classifiers are no longer binary.

There are now three possible relations for an ordered pair of nodes (a,b): **Support**, **Attack** and **Neutral**. In the earlier case (AraucariaDB), when we assumed the sole presence of Support relations, distinction between Support and Neutral also included the effect of features that are not related to directional inference, e.g., common Wikipedia entities. However, those features are now common to both Support and Attack relations. To account for these complications, we consider two approaches:

Two-Step Approach: In the first step, a classifier marks the pairs which have either Support or Attack edge. We call it the **Detection** classifier. In a similar fashion, this classifier can also be type-1 or type-2. In the second step, an independent classifier resolves those edges into Support or Attack. We call the second classifier the **Resolver**. In the Structure Prediction phase, the Detection classifier will decide the best tree structure and the Resolver classifier will then resolve the edges into either Support or Attack.

Single-Step Approach: A multi-class classifier resolves all the three relations: Support, Attack or Neutral. Neutral edges are generated using the type-1 framework: any possible edge formed by nodes within the argument which is not an existing Support or Attack edge. In the Structure Prediction phase, the tree structure with the maximum sum of edge scores is chosen. However, the edge score will now be $confidence_S - confidence_N$ instead of $confidence_S$ for each Support edge, where $confidence_S$ and $confidence_N$ are the confidence scores provided by the Single-Step classifier for the Support and Neutral labels, respectively. Similarly, an edge score of $confidence_A - confidence_N$ will be assigned to each Attack edge, where $confidence_A$ is the confidence score assigned by the Single-Step classifier for the Attack label. The $confidence_N$ is deducted due to the missing Neutral edge if a Support edge is chosen. It is evident that assigning $confidence_S - confidence_N$ resolves to assigning $confidence_S$ in Structure Prediction when there are no Attack labels.

For experiments on these approaches, two types of features were included in addition to the features described in Section 3:

1. Negation Discourse Markers: These markers try to capture contrast or negation sentiments in a sentence. Examples of such markers include: *can't*, *never*, etc. This feature set improved the Single-Step Classifier accuracy by 1.3%.

2. Negation/Contrast Relation Indicators: Features in this category intend to capture negation or contrast relations present in an ordered pair of sentences. We have followed the approaches proposed in (Harabagiu et al., 2006). This feature set improved the Single-Step Classifier accuracy by 6.4%.

In this section, we follow a leave-one-out Cross Validation framework due to the small size of the datasets. Table 4 reports the mean classification accuracies for each classifier for the two datasets. We can see that type-2 framework performs better than type-1 for the Detection classifier. The Two-Step classifier combines the Detection (type-2) and the Resolver classification labels. These results imply that

Classifier	Debatepedia	Wikipedia
Detection (type-1)	0.804	0.535
Detection (type-2)	0.906	0.553
Resolver	0.665	0.719
Two-Step	0.560	0.493
Single-Step	0.761	0.453

Table 4: Classifier Performance for datasets with Attack relations.

Nodes	Arguments	T-S-1	T-S	S-S
7	2	0.83	0.666	0
8	1	0.57 1	0.428	0
9	1	0.875	0.5	0.25
10	2	0.721	0.385	0.055
11	4	0.325	0.225	0.05
12	2	0.545	0.409	0
13	2	0.541	0.333	0
Any	14	0.573	0.387	0.04

Table 5: Mean SimScore for Debatepedia. *T-S-1*: Step 1 of the Two-Step Approach *T-S*: Two-Step Approach *S-S*: Single-Step Approach

a Single-Step classification approach performs better than Two-Step for the Debatepedia dataset[7]. However, we shall see in Table 5 that the Single-Step approach performs poorly in the Structure Prediction phase. Table 5 reports the mean SimScore after the Structure Prediction for the Debatepedia dataset. Similar to Table 2, these results are additionally filtered by the number of nodes in the argument. The third column **T-S-1**, reports performance of the Two-Step approach before the edges are resolved into Support or Attack, i.e. there is no distinction between Support and Attack edges. This is similar to the experiments we performed in Section 5.2. The fourth column **T-S**, reports the overall performance of the Two-Step approach. The fifth column **S-S**, reports the performance of the Single-Step approach. We can see that Single-Step performs poorly as compared to Two-Step approach by a large margin. Table

Nodes	Arg.	T-S-1	T-S	T-S-WL	S-S
2	142	0.507	0.366	0.366	0.274
3	103	0.441	0.305	0.262	0.203
4	34	0.254	0.156	0.176	0.098
Any	279	0.452	0.318	0.304	0.227

Table 6: Mean SimScore for Wikipedia. *T-S-WL*: Two-Step Approach without any restriction for linear structures. Rest of the abbreviations as per Table 5.

6 reports the mean SimScore for the Wikipedia dataset. Here we imposed an additional restriction for the structures to be linear. However, in the fifth column **T-S-WL**, we report the SimScore following the Two-Step approach without any restriction. We observe that the Single-Step approach performs relatively better for the Wikipedia dataset as compared to the Debatepedia dataset. We think this is due to the bigger arguments in Debatepedia.

7 Conclusion

In this paper, we introduced a two-phase approach towards identification of argument structure in natural language text. The first phase involves building models for classifying text-hypothesis pairs into argument relations. The second phase makes use of the classifier confidence scores to construct the argument structure. We have proposed different training models to train the argument relation classifier. With the help of ablation study, we showed that our novel use of word vectors trained on an external corpus can be a crucial feature for such tasks, contributing as much as 11.4% towards the performance. For the final goal of Structure Prediction, our approach predicted almost twice as many correct edges as with the random baseline. We showed that the proposed approach can be extended to arguments containing Attack relations as well, where our experiments predicted an average of 38% edges correctly for Debatepedia dataset.

[7] Arguments having size 13 or less are used in this experiment.

References

Elena Cabrio and Serena Villata. 2012. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 208–212. Association for Computational Linguistics.

Elena Cabrio and Serena Villata. 2014. Node: A benchmark of natural language arguments. *COMMA*, 266:449–450.

Judith Eckle-Kohler, Roland Kluge, and Iryna Gurevych. 2015. On the role of discourse markers for discriminating claims and premises in argumentative discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, Lisbon, Portugal, to appear*. Citeseer.

Paolo Ferragina and Ugo Scaiella. 2010. Fast and accurate annotation of short texts with wikipedia pages. *arXiv preprint arXiv:1006.3498*.

James B Freeman. 1991. *Dialectics and the macrostructure of arguments: A theory of argument structure*, volume 10. Walter de Gruyter.

James B Freeman. 2011. *Argument Structure:: Representation and Theory*, volume 18. Springer Science & Business Media.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, contrast and contradiction in text processing. In *AAAI*, volume 6, pages 755–762.

Gary King and Langche Zeng. 2001. Logistic regression in rare events data. *Political analysis*, 9(2):137–163.

Milen Kouylekov and Matteo Negri. 2010. An open-source package for recognizing textual entailment. In *Proceedings of the ACL 2010 System Demonstrations*, pages 42–47. Association for Computational Linguistics.

John Lawrence, Floris Bex, Chris Reed, and Mark Snaith. 2012. Aifdb: Infrastructure for the argument web. In *COMMA*, pages 515–516.

John Lawrence, Chris Reed, Colin Allen, Simon McAlister, Andrew Ravenscroft, and David Bourget. 2014. Mining arguments from 19th century philosophical texts using topic based modelling. In *Proceedings of the First Workshop on Argumentation Mining*, pages 79–87. Citeseer.

Bill MacCartney. 2009. *Natural language inference*. Ph.D. thesis, Citeseer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Raquel Mochales and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 938–948.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *Proceedings of NAACL-HLT*, pages 1384–1394.

Chris Reed and Glenn Rowe. 2004. Araucaria: Software for argument analysis, diagramming and representation. *International Journal on Artificial Intelligence Tools*, 13(04):961–979.

Alberto Sanfeliu and King-Sun Fu. 1983. A distance measure between attributed relational graphs for pattern recognition. *Systems, Man and Cybernetics, IEEE Transactions on*, (3):353–362.

Christian Stab and Iryna Gurevych. 2016. Parsing argumentation structures in persuasive essays. *arXiv preprint arXiv:1604.07370*.

Douglas Walton. 2007. Visualization tools, argumentation schemes and expert opinion evidence in law. *Law, Probability and Risk*, 6(1-4):119–140.