Incorporating Task-specific Features into Deep Models to Classify Argument Components

Linting Xue
North Carolina State University
Raleigh, North Carolina, USA
lxue3@ncsu.edu

Collin F. Lynch
North Carolina State University
Raleigh, North Carolina, USA
cflynch@ncsu.edu

ABSTRACT

In order to effectively grade persuasive writing we must be able to reliably identify and extract extract argument structures. In order to do this we must classify arguments by their structural roles (e.g., major claim, claim, and premise). Current approaches to classification typically rely on statistical models with heavy feature-engineering or on deep neural-networks that do not consider prior knowledge or other secondary features. Little research has been carried out to investigate if we can incorporate features into deep models to address AM tasks. In this work, we propose to incorporate lightweight features into deep models to classify argument components. We experimented with two state of the art (SOTA) approaches: 1) linear-Long-Short-Term Memory (LSTM) models with concatenated feature vectors; or 2) Directed Acyclic Graph (DAG) structured LSTMs. In our models we incorporated the features of argument position (e.g., if the argument is in the first paragraph) and prior knowledge of discourse indicators (e.g., in conclusion, for example). We use two baselines in our work: 1) prior work using SVM models with heavy feature engineering; 2) traditional linear-Bi-LSTMs with no task-specific features.

Our results show that with a comparatively small number of lightweight features, both linear-Bi-LSTMs and DAG-Bi-LSTMs outperform SVM models that depend on more heavy feature engineering, and outperform linear-Bi-LSTMs with only general word embedding features. These results suggest that incorporating task-specific elements into deep models may potentially benefit argument mining tasks.

Keywords

Argument component classification, deep learning, feature engineering, DAG-LSTMs, LSTMs, argument structures, argumentation mining, automated essay grading

1. INTRODUCTION

Current automated essay grading systems are typically focused on the syntactic and semantic analysis of written arguments via Natural Language Processing (NLP) techniques (as in [7, 23, 3]). These systems are typically designed to evaluate arguments on the basis of: general readability (e.g., the number of prepositions and relative pronouns or the complexity of the sentence structure); shallow semantic analysis (e.g., lexical semantics or the analysis of the relationship among named entities); and syntax analysis (e.g., grammatical analysis). To the extent that argument structures considered in this work have been focused on the limited identification of individual components (e.g., hypothesis statements [4]), or on manual analysis by human experts [14], which is costly and time-consuming. Few existing systems perform any automatic analysis of the argumentative structures or seek to identify structural flaws due to the lack of an auto-extraction mechanism in the system.

In order to parse arguments it is necessary to extract the basic components. Extracting argument structures (EAS) is one of the essential tasks of argumentation mining (AM).EAS can be divided into three sub-tasks: 1) **argument component identification (ACI)** breaking down the text into argument units; 2) **argument component classification (ACC)** of classifying argument component (ACs) into types; 3) **argument relation identification (ARI)** of identifying the relationships between each pair of ACs. Prior researchers have focused on different subsets of these tasks (e.g., [27] addressed ACI, ACC, and ARI separately, [24] jointly modeled ACI and ACC) or built end-to-end models that address them sequentially (e.g. [18]). Our goal in this work, by contrast, is to investigate how to incorporate task-specific features into deep learning models and whether those features can improve our models' performance on the task of ACC. We carried out our work using an argumentation schema developed by Stab and Gurevych on a corpus of 402 persuasive essays (PE) [27]. As part of this work, we replicated their work on ACC and used it as a baseline model.

Most current approaches to ACC either rely on heavy feature engineering [27, 16, 22, 12] or use deep models that only consider pre-trained word embeddings with no other secondary features [19, 24, 11]. Little research to date has been focused on incorporating prior knowledge or lightweight features into deep models for AM tasks. To the best of our knowledge, Lugini and Litman carried out the only work that adds features into LSTM based models to address ACC on the argument dataset of classroom discussions [13]. In that work, they considered a set of features including semantic-density features (e.g. the number of pronouns), lexical features (e.g., unigram and bi-gram), and syntactic features from speech tags. They combined the feature matrix and LSTMs hidden output for classification. They showed that the features boosted the deep model performance.

In our work, we investigated whether or not it is possible to incor-

Linting Xue and Collin Lynch "Incorporating Task-specific Features into Deep Models to Classify Argument Components" In: *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, Anna N. Rafferty, Jacob Whitehill, Violetta Cavalli-Sforza, and Cristobal Romero (eds.) 2020, pp. 531 - 537

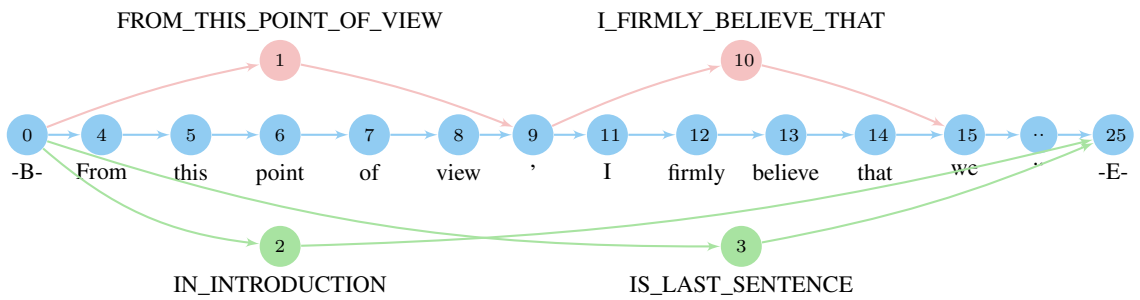

Figure 1: An example of DAG LSTM modeling an AC.

porate lightweight features into deep models to address ACC on PE dataset with different feature sets and deep models. In this work we considered prior research on the prior knowledge of discourse indicators and position features. The discourse indicators have been shown are potential features for identifying the argumentative section of online product reviews [30]. Researchers have also demonstrated that the structural features of AC position (e.g., if the AC shows in the introduction, if an AC is the first sentence of one paragraph) and token statistics (e.g., number of tokens of an AC) are the most effective features for AC classification [27]. For this study we only considered the position information for ACs. We chose to focus on these two features because these two are the most informative and also require the least amount of feature-engineering.

For our deep models, we experimented with Bi-LSTMs. We encoded the incorporated features by one-hot encoding, computed the element-wise summation of feature vectors, and then combined feature vectors with Bi-LSTMs output for prediction. This approach is the same as the work in [13]. We also considered bidirectional DAG-Structured Recurrent Neural Networks (RNNs) to incorporate features. **DAG-RNNs**, also known as Neural Lattice Language models, are an extension of linear-chained RNN models that can consume DAG-structured input [32]. If we treat the text as a linear path, the prior knowledge and secondary features can be added as new edges on the path to form a DAG structure. The discourse features are connected to the parent- and child- nodes of the related tokens, similar to the work of [32]. For position features, we simply connected them with the two special sequence delimiters which indicate the beginning and end of the sentences. Figure 1 shows an example of DAG input with discourse indicators of “FROM_THIS _POINT_OF _VIEW” and “I_FIRMLY_BELIEVE _THAT” in red and position features of “IN_INTRODUCTION” and “IS_LAST _SENTENCE” in green. The original input is in the blue nodes. Token “-B-” and “-E-” are special sequence delimiters. The nodes are indexed in topological order, as this is the order in which the one-directional DAG model consumes the input sequence. For bidirectional DAG models, we simply reverse the order. The intuition behind this approach is that it mimics how humans read and annotate essays as humans can incorporate linguistic intuition to determine the role of the ACs in written argumentation. For example, if a sentence appears to be the last sentence of the introduction in a five-paragraph essay, it most likely contains the author’s standpoint, i.e., claim.

DAG-RNNs have been used to incorporate linguistic knowledge (e.g., the non-compositional phrase in the form of n-gram) for sentiment classification [32]. They have also achieved SOTA results in many other NLP tasks such as neural machine translation [28], speech translation [25], and language modeling [2]. In this work, we utilized LSTMs, a special kind of RNN and building off Zhu et al.’s work [32], we implemented a DAG-LSTM in Tensorflow with a different hidden state bagging function (discussed in section 4).

Our results show that linear-Bi-LSTMs with no task-specific features performed worse than traditional models. However once we incorporated our two features, both the linear-Bi-LSTMs and DAG-Bi-LSTMs outperform general Bi-LSTMs with no features and they outperform other models that rely on heavy feature-engineering. DAG-Bi-LSTMs slightly outperform the linear-Bi-LSTMs when considering both features. The linear-Bi-LSTMs with only position features yield the best results.

The significance of this work is as follows. 1) Our work serves as the basis for automated essay grading systems, and can be applied to extract argument structures for detecting structural flaws. 2) We addressed a common issue in NLP that deep models tend to yield lower performance on small datasets. We showed that deep models can benefit from lightweight features and yield better performance. 3) We experimented with DAG-LSTMs to incorporate features on text classification tasks. We showed that it could be a promising architecture to incorporate features into sequence models. 4) We tested the same approach used in [13] to combine features with LSTMs on a different dataset. Our results are consistent with their work.

2. RELATED WORK

2.1 AC Classification

Most of the prior work on AC classification relies on traditional classification models and heavy feature engineering. In [27], researchers applied multiclass SVMs to classify ACs using structural, lexical, syntactic, discourse indicator, and contextual features. They obtained an F1 score of 0.794 on the PE corpus. In [26] and [18], authors performed classification task on a small portion of the PE dataset, again relying on extracted features. In [10], Namhee et al. analyzed online comments to identify and classify subjective claims using lexical and syntactic features. In [16], researchers worked to classify ACs on legal documents using extracted features while Niall et al. applied kernel methods for argument detection and classification on AraucariaDB dataset [22]. Different from the above, we only considered prior knowledge of discourse indicators and structural information of position features.

Many researchers have begun to explore the application of deep neural-network models to argument mining. In [11], authors experimented with CNN and RNN models to detect the claims [1] and evidences [21] on Wikipedia datasets. Potash et al. proposed a joint sequence-to-sequence model with attention to predict the links between ACs and classify ACs on the PE dataset, where they consid-

ered the sequential nature of ACs [19]. In our work, by contrast, we focused on incorporating prior knowledge to deep neural-networks to classify the ACs alone.

To the best of our knowledge, research from [13] is the only work that combines feature engineering and deep models to address ACC on classroom discussion. They showed that SOTA deep models with only pre-trained embeddings performed poorly on their dataset. However, by including secondary features they improved the performance substantially. The features included: semantic-density features (e.g., number of pro-nouns, descriptive word-level statistics, number of occurrences of words of different lengths), lexical features (e.g,, tf-idf feature for each unigram and bi-gram, descriptive argument move-level statistics), and syntactic features (e.g., unigrams, bigrams, and tri- grams of part of speech tags). In their work, they experimented with Convolutional neural network models and LSTMs. Their results showed that the model's performance was improved after adding the secondary features. In our work, we considered the same approach to incorporate features into linear-LSTMs and compare the model performance with DAG-LSTMs.

2.2 DAG-RNNs

DAG-RNNs, also known as Neural Lattice Language (NLL) models, are extensions of chain-structured RNNs [32, 28]. These models, first proposed by Zhu et al. in [32], leverage DAG structures to incorporate external semantics such as n-gram sentiment tags and expert annotations to improve performance on sentiment classification. Su et al. introduced NLL-based Gated Recurrent Units (GRUs) to encoder multiple word segmentation of Chinese text for translation [28]. Sperber et al. later used NLL-based GRU models to consume word lattices from the up-stream modes of the speech recognizer for speech translation [25]. These lattices were annotated with posterior probabilities on the alternative translation paths. And finally, in [2], researchers demonstrated that the NLL models outperformed the LSTM-based models at the task of language modeling when incorporating multi-word phrases (n-grams) and multiple-embeddings for polysemy. However, little research has been done to utilize DAG-RNNs to integrate features for AM tasks.

3. DATASET

The PE dataset was developed by [27]. It contains 402 essays from the online community *essayforum* [1]. The forum provides writing feedback for different kinds of text. Students can post practice essays for standardized tests in the community and obtain feedback about their writing skills. The dataset was randomly selected from the *writing feedback* section of the forum. The dataset comes with three argument components: *major claim* indicating the author's standpoint on the given controversial topic; *claim* of substandpoints that supports ("for") or attacks ("against") the major claim; *premise* that is the reason of the argument which supports or attacks the claim. Table 1 shows the class distribution of the PE corpus. The average number of tokens in *major claims*, *claims*, and *premises* are 19, 23 and 21, respectively.

	Major Claim	Claim	Premise
Train & Dev	598	1202	3023
Test	153	304	809

Table 1: Number of instance in each class

[1] https://essayforum.com/

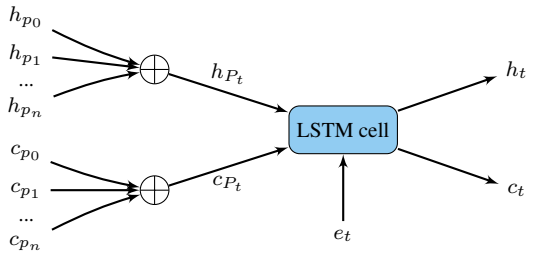

Figure 2: An unit of DAG-LSTM.

4. METHODS

4.1 Linear-Bi-LSTMs

In traditional Bi-LSTM models, ACs are encoded using Glove embeddings [17] that are obtained from training on large Wikipedia datasets. The encoded ACs are fed to Bi-LSTMs, and the last hidden states are passed to the softmax layer for prediction. To incorporate features, we used a one-hot vector to represent each feature and summed up the features vectors that are related to an AC. For example, if we have a total of three features in the feature space that are "if an AC is in the introduction", "if an AC is in the conclusion", and if an AC contains discourse indicator of "in conclusion". We use one-hot vectors to represent three features as [0, 0, 1], [0, 1, 0], and [1, 0, 0], respectively. When we have one example AC that contains a discourse indicator of "in conclusion," and this AC is in conclusion paragraph, we sum up two feature vectors by elements to get a vector of [0, 1, 1], which represents the feature for this AC. Then we concatenate the vectors on the hidden output of Bi-LSTMs for final prediction. The same approach has been used in [13]

4.2 DAG-Bi-LSTMs

We implemented the DAG-Bi-LSTMs using the TensorFlow platform. [1] The DAG-Bi-LSTMs in our work is similar to the models described in [32]. However, we applied a different hidden state merge function. While Zhu et al. used binarization, we elected to sum the parent hidden states as suggested in the TreeLSTM work [29]. Intuitively, by summing the previous states, we expect the DAG-models to learn both the summarized linear history and the incorporated knowledge. We used the same one-hot method to encode the incorporated features.

For sequential inputs the linear-LSTM models calculate hidden states h_t and cell states c_t based upon the proceeding hidden state h_{t-1}, cell states c_{t-1} and the input embedding e_t of token x_t as:

$$\mathbf{h}_t, \mathbf{c}_t = LSTM(\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{e}_t, \theta) \quad (1)$$

The primary difference between DAG- and linear- LSTMs is that the former can have multiple parent and child states, as shown in Figure 2, while the latter cannot. Given a DAG input, h_{p_i} indicates a set of parent states at t time step, where $i = 0, 1, ..., or\ n$ and $p_i \in P$. The DAG model first gathers its parent hidden states h_{p_i} and then sums over the parents' hidden states and over the parents' cell states as follows:

$$\mathbf{h}_{P_t} = \sum_{p_i \in P_t} \mathbf{h}_{p_i} \quad \mathbf{c}_{P_t} = \sum_{p_i \in P_t} \mathbf{c}_{p_i} \quad (2)$$

[1] We also experimented with DAG-Bi-Gated Recurrent Units, but DAG-Bi-LSTMs yielded better results.

The remainder of the DAG process is similar to that of linear-Bi-LSTMs in that h_{P_t}, c_{P_t}, e_t are fed to standard LSTM unit to generate new hidden, cell states of h_t, c_t, which are then copied to the child states. Finally, the last hidden states are fed to Multi-Layered Perceptrons (MLPs) for prediction.

4.3 Prior Knowledge and Features

In this work, we considered the prior knowledge of discourse indicators and the AC position features. In prior work [27], Stab and Gurevych collected a list of hand-crafted features for ACC tasks, including lexical, structural, discourse indicator, contextual, syntactic, etc. The detailed explanations of the features can be found in Section 5.3.1 of [27]. For discourse indicators, they include five categories: *forward indicators* (e.g., "therefore"); *backward indicators* (e.g., "because"); *thesis indicators* (e.g., "in my opinion"); *rebuttal indicators* (e.g., "although"); and *first-person indicators* (e.g., "I", "me"). For the position features, we annotated sentences if they were the first/last sentence and if they showed up in the last/first paragraph. The annotations are in the *special* n-gram form (e.g., "IS_LAST_SENTENCE", "_THEREFORE_") so that they are distinguished from original corpus. We also experimented with annotating the discourse indicators by category, such that the forward indicators were annotated as "FORWARD _INDICATORS".

5. EXPERIMENTAL SETUP

We carried out a series of experiments using the same static training/testing split as in prior work [27]. Since the corpus does not have a designated development set, we used stratified sampling to select 15% of the training set to tune our hyper-parameters and reported our final results on the designated test set. We ran each experiment five times and reported the average Macro-F1 score of the test dataset.

We carried out four distinct experiments: **Base-SVMs**, which replicates the work in [27] with multiclass SVMs using polynomial kernels on a set of features; **Base-LSTMs**, which are baseline models of general Bi-LSTMs with no secondary features; **LSTMs** and **DAG-LSTMs** refers to Bi-LSTMs and DAG-Bi-LSTMs with task-specific features.

We used a grid search for hyper-parameter tuning, and we used the same set of parameters across all the models. We used 300-dimensional GloVe embeddings [17]. Tokens not present in the pre-trained embeddings or not features were randomly initialized with uniform samples from range $[-\sqrt{\frac{3}{dim}}, \sqrt{\frac{3}{dim}}]$ [15] where *dim* is the dimension of the embeddings 300. All of the tokens in the test and dev sets but not in the training set have one unique random embedding. The embeddings were fixed during training. We then used the Adam optimization algorithm [9] with a learning rate of 0.005, a batch size of 32, a layer LSTM with a hidden size of 64, and a drop out rate of 0.2, and a layer tanh-MLP with a hidden size of 64.

6. RESULTS & DISCUSSION

In this section, we will discuss the overall results. Later we will talk about how each feature impacts the linear-LSTMs and DAG-LSTMs by comparing them with traditional SVMs and linear-LSTMs with no task-specific features. In the end, we will compare the performance of linear-LSTMs and DAG-LSTMs.

6.1 Overall Results

Table 2 shows the results of each experiment and our baseline metrics. The standard deviations of the deep model results are all less than 0.009 over the runs. The first three columns show our benchmark, Stab and Gurevych's results with SVMs on: all features, structural features alone (including AC position in the document and token statistics), and contextual features alone (including discourse indicators and the number of noun and verb phrases in an AC). The next two columns show two of our baseline models: Base-SVMs and Base-LSTMs. Then the rest shows the linear-LSTMs and DAG-LSTMs with the two features together and separately. *Pos* includes the position features are considered, while *dis* refers to the discourse indicators were incorporated, and *Pos-dis* indicates that both features are used.

Overall, for linear-LSTMs with only position features return the best macro-F1 score of 0.805 across the board. DAG-LSTMs with both position and discourse features return a very close score of 0.802. Drilling won, linear-LSTMs with position features also yield the best F1 score for claim and premise components, especially for claim components, the F1 score is increased by 23% over the base-LSTMs and increased by 4.5% over base-SVMs. For major claims, the SMVs from prior work still have the best F1 score.

Thus for traditional models with heavy feature engineering, our Base-SVMs are close to Stab and Gurevych's result (SVMs) but with a lower F1 score on major claims. This may be due to minor differences in our feature extraction or the different experimental setting. Among three deep models, the base-LSTMs with only pre-trained word embeddings perform very poorly, and all the F1 scores are much lower than the SVM models, which is not very surprising because of the small amount of data. The trained models do not generalize well on test data. This may also be due to the fact that the pre-trained embeddings are obtained from training models on a large corpus of Wikipedia data [17], which can be thought of prior knowledge. However, Wikipedia is very different from the PEs, the writing is generally more formal; it is a product of collaborative work; and is heavily edited. PEs are most likely composed by non-native English writers. Thus the base-LSTMs with glove embeddings may not able to catch the semantic meaning in the PEs.

However, once we incorporated the position and discourse features, we obtained a very high Macro F1 score. We have F1 of 0.801 and 0.802 for linear and DAG models, a 16% improvement over Base-LSTMs with no features (0.633). These results imply that both models can utilize task-specific features to boost performance. When we compare deep models with heavy feature engineering based SVMs, we find that the deep models with only two features outperformed the SVMs with heavily features Engineering, which suggests that combining little features with deep models can improve the learning on AM tasks.

6.2 Position Features

When adding position features, both the linear models and DAG models outperformed the SVMs and base-LSTMs. The linear models return the best results. In fact, after incorporating position features, the major claims that were previously misclassified as premises by the Base-LSTMs were now either classified as claims or major claims in the linear and DAG models. However, while adding the position features improves the tasks on identifying the claims and major claims, they are not good at distinguishing the two. Looking deeper, the resulting models tend to be biased towards the major claim classification, especially for claims that show up in the introduction or conclusion. One possible explanation for this is that

	SVMs (Stab et al.)			Base-SVMs	Base-LSTMs	LSTMs			DAG-LSTMs		
Features	**All**	**Pos**	**Dis**	**All**	**None**	**Pos-Dis**	**Pos**	**Dis**	**Pos-Dis**	**Pos**	**Dis**
Major claim	**.891**	.803	.656	.844	.625	.832	.823	.650	.852*	.845	.645
Claim	.611	.551	.248	.640	.455	.670	**.685***	.428	.659	.668	.498
Premise	.879	.870	.836	.886	.819	.903	**.907***	.820	.896	.886	.797
Macro-F1	.794	.741	.580	.790	.633	.801*	**.805***	.633	.802	.799	.647

Table 2: Results on classifying ACs on PE corpus. "All" column refers to the eight type of features in [27]. "Pos" column indicates the models with position features. "Dis" columns shows the model results with discourse features. * means significant improvement over Linear-LSTMs with no features. Bold indicates the best result per row.

the position features dominate the feature space in the PE dataset; models pay too much attention to the position features and too little attention to the semantic context.

Our results are consistent with prior work which has suggested that position features play an important role in classifying ACs on the PE dataset. As shown in Table 2 with only position features, the traditional SVMs can reach a macro F1 score of 0.741. The utility of this feature is relatively intuitive. In five-paragraph essays, the major claims usually show up in the first or last paragraphs. And in our PE dataset, 70% of the major claims were either the last sentence of the introduction or in the first sentence of conclusions, while 67% of key subsidiary claims show up in the first or last sentence of the middle paragraphs.

Our results also suggested that we can incorporate non-semantic features into deep models to help the model learn, especially these non-semantic features can not be captured by the word embedding features.

6.3 Discourse Indicator features

Interestingly, the performance of linear models was not improved after adding the discourse features, and DAG model's performance was improved a little. When we examine the data more closely we see that one possible reason is the current discourse indicator list provided in [27] does not cover all the cases in the PE. We identified more discourse indicators that are not included in the list, such as thesis indicators of "it is believed that', "to summarise", "in short", "it is undeniable", and "I admit that" and forward indicators of "based on the above discussion". We will address this problem in our future work.

We also experimented with incorporating discourse features by category. We ran another set of experiments based on the same model parameters setting as above. Below Table 3 shows the results.

	LSTMs		DAG-LSTMs	
Features	**Pos-Dis**	**Dis**	**Pos-Dis**	**Dis**
Major Claim	.843*	.662*	.859*	.675*
Claim	.666	.475*	.659	.506*
Premise	.899	.817	.894	.794
Macro-F1	.803*	.651*	.804*	.659*

Table 3: Results for incorporating discourse indicators as annotation types for Linear-LSTMs and DAG-LSTMs.* means improvement over the results that show in Table 2.

After incorporating discourse indicator features by category, the model's performance was improved. In Table 3, * indicates the improvement over the results from n-gram discourse features. Both the linear and DAG models yield slightly better results on major claim and claim components, especially the major claim. Deeply looking at the results, some major claims were misclassified as claims before, which were correctly predicted here. The reason could be that when we consider the discourse indicators by category, we only have five features. For each feature, we have sufficient training examples for it. In fact, the thesis, first-person, forward, rebuttal, and backward indicators show up 701, 1535, 968, 719, and 1769 in the total data. Thus, it is easier for the models to capture the discourse features. However, when we considered them in the n-gram form, we have more than 100 of them. The number of them shows up in the ACs are much less than above. And some of them only show up once in the entire data, such as thesis indicators of "all things considered" and backward indicator of "is due to the fact that", which prohibits the models learning useful information.

Overall, the discourse indicator features did not improve the model's performance massively, which indicates that deep models with pretrained embeddings already capture the semantic information. Thus, adding discourse indicator features does not help as much as position features on PE dataset.

6.4 Linear-LSTMs vs DAG-LSTMs

When adding both classes of features, the linear and DAG models yielded similar results for their macro-F1 score. However, the linear models outperformed the DAG models with the position features alone, and the DAG models utilized the discourse indicator features better in both the Table 2 and Table 3. One possible explanation is that when we concatenated the feature vectors on the last hidden output of linear models, the models are not able to learn the interactions between the discourse indicators and surrounding words. But the DAG models can learn those interactions by merging the hidden annotation states with current hidden states, and the current states contain the semantic information from all previous tokens. For example, for the DAG-input shown in Figure 1, we use the index of the DAG input to refer the time step that the LSTM unit processes the hidden state. At time step 15 of the forward training, we first merge the hidden output of state 10 and state 14, and then pass the merged hidden state as the hidden input of state 15. In this way, the DAG models consider both semantic meaning of all previous tokens and the discourse feature of "I firmly believe that", and pass that information to the next hidden state. These results imply that DAG-models tend to utilize the semantic features better as they can learn the interaction between the features and tokens. However, they might not perform very well when we consider non-semantic features, such as the position features used here. One possible ap-

proach to address this problem is to combine two proposed methods to incorporate features. We can use DAG models to incorporate discourse indicator features and then concatenate the one-hot position feature vectors on the final hidden states for prediction.

7. CONCLUSIONS

In this work, we experimented with two approaches to combine feature engineering with deep models: linear-Bi-LSTMs with feature vectors concatenated on the hidden output and DAG-Bi-LSTMs. Our results show that both deep models could benefit from task-specific features, as both of them outperformed traditional models with heavy feature engineering and deep models with no task-specific features. We also show that the linear models handle the position feature better, and that the DAG models utilize the semantic features better since the linear models can not learn the interaction between discourse features and tokens. And finally we show that the deep models benefit more from position features than discourse indicator features on the PE dataset. Our results imply that when we apply the deep learning models to classify ACs, we could consider utilizing some task-specific features to guide the model learning.

This work can serve as a basis for the development of structurally-aware support platforms for reading and writing. This can include automated essay grading systems that detect and evaluate structural deficiencies as well as writing tutors that scaffold the construction of coherent essays or identify structural issues. As discussed in Section 1, current automated grading systems suffer from the lack of reliable auto-extraction mechanisms with most still relying on traditional ML models that use heavy feature engineering to function. Such work is costly and time consuming to develop and may not always generalize to other essay types. Our work addresses this problem by showing that lightweight features and off the shelf methods can outperform those methods. At the same time our work also showed that while traditional machine learning models are costly and deep learning models are sensitive to small datasets, as discussed by [8, 31], this limitation too can be addressed through the use of lightweight feature work to guide the deep models. By addressing these two problems we have shown a path for developing robust argument detection mechanisms for automated educational platforms using novel deep learning approaches a path that can lead to substantive improvements for students and educators.

8. FUTURE WORK

These preliminary results serve as a basis for our ongoing research, in which we are building an end-to-end model with feature engineering to address all three sub-tasks for argument structure extraction. For that work we will frame this task as sequence tagging problems. We propose to use linear-LSTM and DAG-LSTM based models with task-specific features to address EAS. We estimate that incorporating the task-specific features into end-to-end models can improve the model's performance compared to the deep models based on general word embedding [6].

In future work, we will also consider experimenting these two approaches on different argumentation datasets, and compare the results with fine-tuning SOTA language models (e.g. BERT [5], T5 [20]).

9. REFERENCES

[1] E. Aharoni, A. Polnarov, T. Lavee, D. Hershcovich, R. Levy, R. Rinott, D. Gutfreund, and N. Slonim. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the first workshop on argumentation mining*, pages 64–68, 2014.

[2] J. Buckman and G. Neubig. Neural lattice language models. *TACL*, 6:529–541, 2018.

[3] J. Burstein, C. Leacock, and R. Swartz. Automated evaluation of essays and short answers. 2001.

[4] J. Burstein, D. Marcu, and K. Knight. Finding the write stuff: Automatic identification of discourse structure in student essays. *IEEE Intelligent Systems*, 18(1):32–39, 2003.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] S. Eger, J. Daxenberger, and I. Gurevych. Neural end-to-end learning for computational argumentation mining. In R. Barzilay and M. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 11–22. Association for Computational Linguistics, 2017.

[7] M. A. Hearst. The debate on automated essay grading. *IEEE Intelligent Systems and their Applications*, 15(5):22–37, 2000.

[8] N. A. Khayi and V. Rus. Bi-gru capsule networks for student answers assessment. In *2019 KDD Workshop on Deep Learning for Education (DL4Ed)*, 2019.

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] N. Kwon, L. Zhou, E. Hovy, and S. W. Shulman. Identifying and classifying subjective claims. In *Proceedings of the 8th annual international conference on Digital government research: bridging disciplines & domains*, pages 76–81. Digital Government Society of North America, 2007.

[11] A. Laha and V. Raykar. An empirical evaluation of various deep learning architectures for bi-sequence classification tasks. *arXiv preprint arXiv:1607.04853*, 2016.

[12] M. Lippi and P. Torroni. Context-independent claim detection for argument mining. In *IJCAI*, volume 15, pages 185–191, 2015.

[13] L. Lugini and D. J. Litman. Argument component classification for classroom discussions. In N. Slonim and R. Aharonov, editors, *Proceedings of the 5th Workshop on Argument Mining, ArgMining@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 57–67. Association for Computational Linguistics, 2018.

[14] C. F. Lynch, K. D. Ashley, and M. Chi. Can diagrams predict essay grades? In S. Trausan-Matu, K. E. Boyer, M. E. Crosby, and K. Panourgia, editors, *ITS*, Lecture Notes, pages 260–265. Springer, 2014.

[15] X. Ma and E. H. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[16] R. Mochales and M.-F. Moens. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22, 2011.

[17] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[18] I. Persing and V. Ng. End-to-end argumentation mining in student essays. In *Proceedings of the 2016 Conference of the*

North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1384–1394, 2016.

[19] P. Potash, A. Romanov, and A. Rumshisky. Here's my point: Joint pointer architecture for argument mining. *arXiv preprint arXiv:1612.08994*, 2016.

[20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[21] R. Rinott, L. Dankin, C. Alzate, M. M. Khapra, E. Aharoni, and N. Slonim. Show me your evidence-an automatic method for context dependent evidence detection. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 440–450, 2015.

[22] N. Rooney, H. Wang, and F. Browne. Applying kernel methods to argumentation mining. In *FLAIRS Conference*, 2012.

[23] L. M. Rudner and T. Liang. Automated essay scoring using bayes' theorem. *The Journal of Technology, Learning and Assessment*, 1(2), 2002.

[24] C. Schulz, S. Eger, J. Daxenberger, T. Kahse, and I. Gurevych. Multi-task learning for argumentation mining in low-resource settings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 35–41, 2018.

[25] M. Sperber, G. Neubig, J. Niehues, and A. Waibel. Neural lattice-to-sequence models for uncertain inputs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1380–1389, 2017.

[26] C. Stab and I. Gurevych. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, 2014.

[27] C. Stab and I. Gurevych. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659, 2017.

[28] J. Su, Z. Tan, D. Xiong, R. Ji, X. Shi, and Y. Liu. Lattice-based recurrent neural network encoders for neural machine translation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3302–3308, 2017.

[29] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[30] A. Z. Wyner, J. Schneider, K. Atkinson, and T. J. Bench-Capon. Semi-automated argumentative analysis of online product reviews. *COMMA*, 245:43–50, 2012.

[31] Y. Xu and C. F. Lynch. What do you want? applying deep learning models to detect question topics in mooc forum posts? In *2019 KDD Workshop on Deep Learning for Education (DL4Ed)]*, 2019.

[32] X. Zhu, P. Sobhani, and H. Guo. Dag-structured long short-term memory for semantic compositionality. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,San Diego California, USA, June 12-17, 2016*, pages 917–926, 2016.