Unsupervised Argumentation Mining in Student Essays

Isaac Persing and Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688, USA
{persingq,vince}@hlt.utdallas.edu

Abstract

State-of-the-art systems for argumentation mining are supervised, thus relying on training data containing manually annotated argument components and the relationships between them. To eliminate the reliance on annotated data, we present a novel approach to unsupervised argument mining. The key idea is to bootstrap from a small set of argument components automatically identified using simple heuristics in combination with reliable contextual cues. Results on a Stab and Gurevych's corpus of 402 essays show that our unsupervised approach rivals two supervised baselines in performance and achieves 73.5−83.7% of the performance of a state-of-the-art neural approach.

Keywords: Opinion Mining / Sentiment Analysis, Discourse Annotation, Representation and Processing, Information Extraction, Information Retrieval

1. Introduction

Recent years have seen a surge of interest in argumentation mining (see Cabrio and Villata (2018) and Lawrence and Reed (2019) for comprehensive surveys of this area of research). Argumentation mining typically involves addressing two subtasks: (1) *argument component identification* (ACI), which consists of identifying the locations and types of the components that make up the arguments (i.e., major claims, claims, and premises), and (2) *relation identification* (RI), which involves identifying the type of relation that holds between two argument components (i.e., support, attack, none). As an example, consider the following text segment taken from a corpus of student essays that Stab and Gurevych (S&G) annotated with argument components and their relations (Stab and Gurevych, 2016):

> In my view point, (1) <u>I would agree to idea of taking a break before starting higher education</u>. (2) <u>Students who take break benefit by traveling around the places or by working</u>. (3) <u>They tend to contribute more due to their real world experiences which makes them more mature</u>.

In this example, premise (3) supports claim (2), and (1) is a major claim.

State-of-the-art argument mining systems are *supervised*, adopting a *feature-rich* approach that typically include large variety of structural, lexical, and syntactic features. The plethora of features seem to suggest that argument-annotated training data are indispensable for training these systems, as they provide a *supervised* learner guidance on which features are useful and how features should be combined. Moreover, world knowledge and domain-specific knowledge can potentially be learned from annotated data, as can be seen from the aforementioned example.

A natural question, then, is: in the absence of argument-annotated training data, is argument mining doomed to fail? In particular, are argument-annotated training data as indispensable as they seem in enabling state-of-the-art argument mining systems to achieve their current level of performance? By blindly applying supervised approaches with a rich feature set, previous work has largely failed to answer this question.

To shed light on this question, we propose a novel *unsupervised* approach to argument mining, with the goal of determining how well we can do *without* argument-annotated data. The key idea behind our approach is to bootstrap from a small set of argument components that are automatically identified and labeled using simple heuristics in combination with reliable contextual cues. We compare our approach against two supervised baselines and a state-of-the-art neural model in a challenging evaluation setting, the *end-to-end* setting, where we perform argument mining on raw, *unannotated* text.[1] In an evaluation on a corpus of 402 essays annotated by S&G, our unsupervised approach rivals its supervised counterparts in performance.

We believe that our findings have two important ramifications. First, our competitive results call for a reexamination of existing supervised approaches to argument mining. Specifically, future work should seek to understand what caused their performances to be rivaled by an unsupervised system. For instance, is it the *features* that fail to adequately capture the information provided by the annotated data or the *learner* that fails to effectively exploit the annotated data? Or is the current amount of annotated training data insufficient for effective learning? Second, our promising results could spark interest in the development of unsupervised and semi-supervised approaches to argument mining, which could reduce a system's reliance on

[1]Many existing argument mining systems were evaluated in a *non-end-to-end setting* (e.g., Stab and Gurevych (2014), Peldszus and Stede (2015), Wei et al. (2017)). For instance, Stab and Gurevych (2014) trained an ACI classifier and applied it to classify only *gold* argument components (i.e., text spans corresponding to a major claim, claim, or premise in the gold standard) or sentences that contain no gold argument components (as *non-argumentative*). Similarly, they applied their learned RI classifier to classify only the relation between two *gold* argument components. In other words, they simplified both tasks by avoiding the challenging task of identifying the locations of argument components. Consequently, their approach cannot be applied in a realistic setting where the input is an *unannotated* text.

the expensive-to-obtain argument-annotated training data.
The rest of the paper is organized as follows. In Section 2, we review related work on argument mining. Section 3 describes our evaluation corpus, the S&G corpus. Section 4 provides an overview of our supervised baselines. In Section 5, we present our unsupervised approach. Finally, we present evaluation results in Section 6 and our conclusions in Section 7.

2. Related Work

Recall that identifying argumentative structures consists of (1) identifying the locations and types of the argument components, and (2) identifying how they are related to each other.
Some researchers focused on *argument location identification*, classifying whether a sentence contains an argument (Florou et al., 2013; Moens et al., 2007; Song et al., 2014; Swanson et al., 2015). Others focused on *argument component typing*, determining the *type* of an argument component. While the vast majority of previous works perform argument component typing at the *sentence* level (Rooney et al., 2012; Teufel, 1999; Burstein et al., 2003; Ong et al., 2014; Falakmasir et al., 2014; Levy et al., 2014; Lippi and Torroni, 2015; Lippi and Torroni, 2016; Rinott et al., 2015), some recent work focused on the more difficult task of typing argument components at the clause level (Park and Cardie, 2014; Goudas et al., 2015; Sardianos et al., 2015).
Some researchers focused on *relation identification* instead. For instance, Nguyen and Litman (2016) showed how context can be exploited to identify relations between argument components. Stab and Gurevych (2014) and Peldszus and Stede (2015), on the other hand, addressed both argument component typing and relation identification, but simplified the task by assuming as input *gold* argument components.
Finally, some work addressed all argument mining subtasks (e.g., Persing and Ng (2016)). Unlike our system, however, virtually all of the aforementioned systems are *supervised*.

3. Corpus

Our corpus consists of 402 persuasive student essays collected and annotated by S&G. Some relevant statistics are shown in Table 1. Each essay is an average of 4.6 paragraphs (16.8 sentences) in length and is written in response to a topic such as "competition or co-operation-which is better?".
This corpus is ideal for argumentation mining because (1) student essays are more simply structured than professional writing, making them appropriate for early work on the task, (2) a major application for argumentation mining is the evaluation of student essays, and (3) the corpus has been previously annotated for argumentation mining.
The corpus annotations describe the essays' argument structure, including the locations and types of the components that make up the arguments, and the types of relations that hold between them. The three annotated argument component types include: **major claims**, which express the author's stance with respect to the essay's topic, **claims**, which are controversial statements that should not be accepted by readers without additional support, and

Essays: 402	Paragraphs: 1,833	Sentences: 6,741
Major claims: 751	Claims: 1,506	Premises: 3,838
Support relations: 3,613	Attack relations: 219	

Table 1: Corpus statistics.

(a) Potential left boundary locations

#	Rule
1	Exactly where the S node begins.
2	After an initial explicit connective, or if the connective is immediately followed by a comma, after the comma.
3	After nth comma that is an immediate child of the S node.
4	After nth comma.

(b) Potential right boundary locations

#	Rule
5	Exactly where the S node ends, or if S ends in a punctuation, immediately before the punctuation.
6	If the S node ends in a (possibly nested) SBAR node, immediately before the nth shallowest SBAR.
7	If the S node ends in a (possibly nested) PP node, immediately before the nth shallowest PP.

Table 2: Rules for extracting ACC boundary locations.

premises, which are reasons authors give to persuade readers about the truth of another argument component statement. The two relation types include: **support**, which indicates that one argument component supports another, and **attack**, which indicates that one argument component attacks another.

4. Supervised Baseline Systems

Next, we describe two state-of-the-art supervised baselines to end-to-end argument mining.

4.1. Pipeline (PIPE)

Our first baseline is a pipeline-based argument mining system previously developed by us (Persing and Ng, 2016) (henceforth P&N). It is a pipeline system because it solves the ACI and RI subtasks sequentially.

4.1.1. Argument Component Identification

PIPE's approach to the ACI subtask, in turn, also consists of two steps. In the first step, PIPE identifies a set of *argument component candidates* (ACCs), sequences of text that could potentially be argument components, from a paragraph. To do this, it first parses each of the paragraph's sentences using the Stanford CoreNLP toolkit (Manning et al., 2014). Then it applies a set of low precision, high recall heuristic rules developed by us based on the sentence's syntactic parse tree to extract the ACCs. These rules are shown in Table 2: they are able to identify an ACC with the same exact boundaries as 92% of all argument components.
Given this set of extracted ACCs, PIPE trains a classifier for ACI using MALLET's (McCallum, 2002) implementation of maximum entropy classification to label the ACCs with argument component labels. Each ACC serves as an

instance whose class label is the same as that of the argument component sharing its exact boundaries (major claim, claim, or premise). If there is no AC sharing the ACC's exact boundaries, it is labeled "non-argumentative". Each instance is represented using S&G's structural, lexical, syntactic, indicator, and contextual features for solving (a simplified version of) the same problem.[2]

4.1.2. Relation Identification

After the classifier in Section 4.1.1. predicts which ACCs in a paragraph represent real argument components as well as their labels, PIPE is ready to determine how the ACCs are related to each other. It considers RI between pairs of ACCs a five class classification problem. Given a pair of ACCs A_1 and A_2 where A_1 occurs before A_2 in the essay, either they are unrelated, A_1 supports A_2, A_2 supports A_1, A_1 attacks A_2, or A_2 attacks A_1.

To solve this problem, PIPE also learns an RI maximum entropy classifier. Each training instance, called a *relation candidate* (RC), consists of a pair of ACCs and one of the above five labels. By default, the instance's label is "no relation" unless each ACC has the exact boundaries of a gold standard argument component and one of the remaining four relations holds between the two gold argument components.

To train the classifier, an RC corresponding to each of the gold relations in all paragraphs is created as an instance of one of the four argumentative labels mentioned above. From this gold relation, PIPE also creates four additional "no relation" RCs by replacing each of the original relationship's participating ACs with nearby ACCs. PIPE represents each RC using S&G's feature set for (a simpler version of) the same task. This feature set consists of structural, lexical, syntactic, and indicator features.[1]

Since information about gold ACs is not available in the test set, test RCs must be created differently. PIPE generates test RCs from all possible pairs of ACCs in the same paragraph that the ACI system predicted as something other than "non-argumentative".

4.2. Integer Linear Programming (ILP)

Two problems with the PIPE approach outlined above are that errors made by the ACI subsystem propagate to the RI task, thus reducing RI performance, and there are constraints on the annotations in S&G's corpus that aren't enforced by PIPE.

To solve both of these problems, we employ as our second baseline an ILP approach to argument mining previously proposed by us (Persing and Ng, 2016), which performs ILP-based joint inference over the outputs of the ACI classifier and the RI classifier used in PIPE. Unlike in PIPE, where ACI influences RI (but not vice versa), in a joint inference approach, both tasks can influence each other. For instance, if a paragraph contains an RC that is very likely to be a support relation (according to the RI classifier), but one of its participating ACCs' probability of being argumentative according to the ACI classifier is just barely too low for the PIPE system to label it as such, the ILP system can choose to label the ACC with one of the argumentative ACI labels in order to allow it to participate in the relationship. In other words, the final ACI decisions and RI decisions will be made jointly by the two components by considering the confidence values they individually assign to the extraction decisions.

In order to benefit from this kind of reasoning, the ILP system cannot use the same set of ACCs and RCs used by PIPE, as PIPE excludes from consideration any RCs that can be generated from ACCs that are most probably non-argumentative. To account for this, the ILP system selects (1) the 3 most likely premise ACCs from each sentence, (2) the 5 most likely claim ACCs from each paragraph, and (3) the 5 most likely major claim ACCs from each essay occurring in the first or last paragraph.[3] It then constructs RCs from all pairings of these ACCs occurring in the same paragraph as long as at least one was chosen because it might be a premise and neither one was chosen because it might be a major claim, since the corpus adheres to the constraint that all premises are in a relationship and all major claims are not. The test RCs generated in this way are presented to the RI maximum entropy classifier normally.

This introduces the problem that the RI classifier may posit a relationship between incompatible ACCs (e.g. one might be labeled non-argumentative). ILP addresses this kind of problem by enforcing consistency constraints on the outputs. The constraints that we employ are all constraints on the annotations in S&G's corpus. In particular, these constraints ensure that (1) each ACC is assigned exactly one type, (2) each RC is assigned exactly one type, (3) if there is a relationship between two ACCs, the ACCs must both be assigned argumentative types (i.e. they can't be labeled non-argumentative), (4) major claims can only occur in the first or last paragraphs, (5) major claims have no parents, (6) a premise must have a parent, (7) argumentatively labeled ACCs cannot overlap in the text, (8) a paragraph contains at least one claim or major claim, and (9) a sentence must not have more than two argumentatively labeled ACCs. ILP can then be used to perform joint inference subject to these constraints.

We claimed above that ILP can be used to find the best assignment of labels to our ACCs and RIs, but did not define how to measure an assignment's quality. Because performance on end-to-end argument mining is evaluated as the average of ACI and RI F-scores, the ILP system's objective function is constructed to maximize this number. Recall that the general F-score formula can be simplified to:

$$F = \frac{2TP}{2TP + FP + FN} \quad (1)$$

where TP, FP, and FN are counts of true positive, false positive, and false negative predictions respectively. An integer linear program cannot directly maximize this formula for any task because division cannot be encoded into an objective function and TP, FP, and FN can only be calculated if the system has access to gold standard test data. For

[2]Space limitations preclude a description of these features. See Stab and Gurevych (2014) for details.

[3]The ILP system selects more premise than claim ACCs (3 per sentence vs 5 per paragraph) because the corpus contains over twice as many premises as claims. The corpus constrains major claims so that they may only appear in an essay's first or last paragraph.

this reason, ILP instead encodes $\frac{G_{ACI}+G_{RI}}{2}$ as the objective function its integer linear program maximizes, where G_{ACI} and G_{RI} are the values of G below as calculated on the ACI and RI tasks respectively:

$$G = \alpha 2TP_e - (1-\alpha)(FP_e + FN_e) \quad (2)$$

where TP_e, FP_e, and FN_e, are *expected* values for TP, FP, and FN respectively, and α attempts to balance the importance of maximizing the numerator vs minimizing the denominator in the F-score formula.[4] These expected values for a complete labeling of all ACCs and RCs are calculated based on the probabilities assigned to the labeled instances as returned by the ACI and RI classifiers.

5. Unsupervised Argumentation Mining

Our unsupervised approach to argument mining first uses the same set of heuristics as the two baselines to identify a set of ACCs that can potentially be argument components from an essay, as described in Table 2. It then operates in four steps, as described below.

5.1. Step 1: Heuristic Labeling

Our approach begins by heuristically labeling a subset of these ACCs with argument component labels. These heuristics rely on three factors: (1) the number of the paragraph the ACC appears in (i.e., whether it is the first, last, or a middle paragraph), (2) the location of the sentence the ACC appears in within its paragraph (i.e., whether it is the first, last, or a middle sentence), and (3) the context n-grams surrounding the ACC. These context n-grams, as well as our heuristics, were designed based on the observations we made on 200 unannotated essays taken from another essay corpus, the International Corpus on Learners' English (Granger et al., 2009).

5.1.1. Heuristically Labeling Major Claims

The reason our heuristics depend on an ACC's paragraph's location within an essay is that the typical argumentative structure of, for example, the first paragraph, looks very different than the typical argumentative structure of a body paragraph (which occurs between the first and last paragraph). Indeed, there is a hard constraint on this corpus that major claims can only appear in the first and last paragraphs, which makes sense because essays are often introduced with with a thesis statement (a major claim), and often conclude with one. So for this reason, we search only the first and last paragraphs for ACCs to heuristically label as major claims.

To find major claims in the first paragraph, we first observe that each type of argument component is often introduced using some discourse marker like those shown in Table 3. In particular, major claims often follow the phrases in the first row of the major claim column. For this reason, we identify the last occurrence of one of these connectives within an essay's first paragraph (because introductory paragraphs are more likely to conclude with a major claim than they are to begin with one). We then find the ACC that appears most closely after the phrase which ends at the end of a simple, declarative clause (as determined by a syntactic

(a) Preceding AC

Major Claim	Claim	Premise
consequently	consequently	for instance ,
contend that	contend that	further ,
feel that	feel that	furthermore ,
i agree	finally ,	in addition ,
i believe	first ,	in fact ,
i feel	first of all ,	indeed ,
i think	firstly ,	moreover ,
in conclusion ,	i agree	since
	i believe	to illustrate ,
maintain that	i feel	
my opinion	i think	
my view	in conclusion ,	
so	in short ,	
suppose that	last ,	
therefore	lastly ,	
think that	maintain that	
thus ,	my opinion	
	my view	
	second ,	
	secondly ,	
	so	
	suppose that	
	therefore	
	think that	
	third ,	
	thirdly ,	
	thus ,	

(b) Succeeding AC

Major Claim	Claim	Premise
because	because	, so
that is ,	that is ,	, therefore

Table 3: Context n-grams used to heuristically label argument components of each type. Subtables (a) and (b) show the context n-grams preceding and succeeding an AC, respectively.

parse of the sentence), heuristically labeling this ACC as a major claim.

If no major claim can be identified in this way, we search the essay for the last occurrence of one of the n-grams in major claim's succeeding row in Table 3. The last ACC occurring before this n-gram that exactly covers a simple, declarative clause is heuristically labeled as a major claim, as n-grams in this cell frequently indicate that whatever preceded them is about to be further explained or reasoned about, which in turn indicates the importance of the preceding text to the argument.

If no major claim ACC can be identified in either of these ways, we simply do not heuristically label any major claims from this paragraph.

Essays' last paragraphs tend to be structurally very similar to their first paragraphs, containing major claims and little else. For this reason, we follow the same procedure outlined above for identifying major claims in an essay's last paragraph. However, since major claims are sometimes expressed without any nearby discourse markers like those shown in Table 3, it is important for us to identify a few major claims without them in order for our eventual learned

[4] We tune α, allowing it to take any value from 0.7, 0.8, or 0.9, as this range tended to perform well in early experiments.

system to generalize about major claim structure.
Since it is a little bit more common for last paragraphs to include major claims than it is for first paragraphs to include them, and since a paragraph's last sentence is more likely to include a major claim than its first, we employ the following additional rule for identifying major claims in the last paragraph. If an essay's last paragraph is very short, including two or fewer sentences, and we are not able to heuristically label any other major claims from this last paragraph using any of the methods described above, we heuristically label the longest simple, declarative clause in the last sentence as a major claim.

5.1.2. Heuristically Labeling Claims

While claims can occur in any paragraph, we only attempt to heuristically label claims occurring in body (middle) paragraphs. To understand the reason why we do this, it is helpful to compare the first two columns of Table 3. Notice that the columns' contents are identical except that the discourse markers preceding claims include ordinal markers (e.g., first, second, last). This is because, just as major claims express the main purpose of an essay, claims express the main purpose of a paragraph. Thus, they tend to be introduced using similar phrases. This makes it difficult to distinguish between claims and major claims occurring in an essay's first or last paragraphs.
Because of this similarity between claims and major claims, the way we heuristically identify claims in a body paragraph is identical to the way we identify major claims in the first paragraph except for the following two differences. First, we use discourse markers from the claim column of Table 3 rather than ones from the major claim column. Second, we exclude from consideration any ACCs occurring outside of the paragraph's first or last sentences. We do this because, while claims can occur in any sentence, body paragraphs tend to either begin or conclude by stating the claim the paragraph is about. The remaining sentences tend to be filled with premises that support the claim.

5.1.3. Heuristically Labeling Premises

Since premises are rare in first or last paragraphs, we also only heuristically label premises occurring in body paragraphs. We also exclude from premise labeling consideration any ACCs occurring in a paragraph's last sentence because, even if we cannot heuristically identify a claim in a last sentence, there is too strong a chance that the reason for this is because our claim labeling heuristics do not have a high enough recall and simply failed to recognize an existing claim's presence.
With those exceptions, our premise labeling heuristics are very similar to our claim and major claim labeling heuristics. We first find all occurrences in an essay of preceding premise discourse markers from Table 3. For each of these occurrences, we find the closest following ACC that ends at the end of a simple declarative clause and heuristically label it as a premise. We then find all occurrences in an essay of succeeding premise discourse markers from the table. For each of these occurrences, we heuristically label as a premise the nearest preceding ACC having the same boundaries as a simple declarative clause. The selected preceding or succeeding ACC can neither overlap with a previously heuristically labeled ACC nor occur in a different sentence than the discourse marker that triggered its labeling.
As can be seen in the description in the previous paragraph, our heuristics for labeling premises are not limited to labeling one premise per paragraph. The reason for this is that it is most common for a paragraph to contain one or fewer claims or major claims, but body paragraphs tend to contain very few non-argumentative sentences. Those body paragraph sentences that do not contain a claim usually contain a premise.
Despite being more permissive than the claim and major claim heuristics, this premise labeling method still fails to capture that most sentences contain a premise. To improve our recall on heuristically extracting premises, we search all body paragraph sentences except the last sentence for occurrences of any of the phrases from Table 3. If we do not find any of these phrases, we take this as a strong indication that the sentence itself is a premise, since the presence of one of these phrases might indicate that we just failed to identify an argument component that is actually present for reasons of precision. We therefore heuristically label the longest ACC occurring in such a sentence as a premise.
An exception to the last premise labeling heuristic is that, if the sentence begins with some introductory phrase measuring three tokens or less followed by a comma, we instead label the longest ACC occurring after this phrase as the premise. We do this because there are too many phrases that can be used to introduce a premise for us to enumerate, but as we can see from the premise column of the table, when they do exist, they are usually short and end with a comma.

5.2. Step 2: Classifier Training

A problem with the labels we applied in the previous subsection is that they are intended to be high precision but low recall. Ideally, an ACI system will strike an appropriate balance between precision and recall in order to maximize its ACI f-score. For this reason, our ACI system needs to be more general than the hand-constructed heuristics described above. So rather than treating our heuristic labeling as an ACI system's predicted labels, we instead employ the heuristically labeled data as our initial training data to train a classifier.
More specifically, we first extract all the ACCs from an unannotated training set. We label all instances that can be labeled heuristically with the appropriate ACI class label (major claim, claim, or premise), and label the remaining ACCs as probably *n*on-argumentative. Given this set of extracted ACCs, we train a four class maximum entropy classifier for ACI using MALLET. Each instance is represented using S&G's structural, lexical, syntactic, indicator, and contextual features for solving (a simplified version of) the same problem.[1]

5.3. Step 3: Self-Training

The result of the previous step is a maximum entropy classifier that can assign a probability distribution to any ACC describing how likely it is to be a major claim, claim, premise, or non-argumentative. A problem with this classifier is that, recall, it was trained on data wherein the non-argumentative training instances may have been labeled as such simply because we could not heuristically determine with enough confidence that they should have an argumentative label.

For this reason, we seek to improve the classifier via self-training. Specifically, we apply the classifier to the non-argumentatively labeled training ACCs. We select the 10 of these instances that the classifier is most confident are argumentative if it assigns at least an 80% probability to one of the argumentative classes. We then label these ACCs with the labels that the classifier indicates are appropriate and add them to the heuristically labeled dataset before training a new maximum entropy classifier on this labeled data. We repeat this process until there are no more training ACCs labeled with the non-argumentative label that can be relabeled in this way (i.e., the classifier assigns none of them an 80% probability of belonging to one of the argumentative classes).

5.4. Step 4: Building the Argument Tree

After applying the ACI classifier to the first or last paragraph in an essay, we use the resulting predictions to construct a candidate argument tree (CAT) for the paragraph in the following way. Because the only ACs appearing in most first or last paragraphs are major claims, and because it is uncommon for more than one major claim to appear in one of these paragraphs, we heuristically label the ACCs in such paragraphs as follows. First, we identify the ACC that is most likely to be a major claim, as indicated by the ACI maximum entropy classifier. If its probability of being a major claim is > 0.5, we label it as a major claim. Otherwise it is labeled as non-argumentative. All other ACCs in the paragraph are labeled non-argumentative.

After applying the ACI classifier to a body paragraph, we use the resulting predictions to construct a CAT for the paragraph in the following way. As we discussed earlier, body paragraphs are structured differently. A typical body paragraph contains only one claim and many premises that support it. Major claims are not permitted in body paragraphs as per the annotation guidelines for this corpus. For this reason, to construct our body paragraph CAT, we first identify the ACC in the body paragraph with the highest probability of being a claim, as determined by the ACI maximum entropy classifier. We label this ACC as a claim. We then rank all remaining ACCs by their probability of being a premise, from most probable to least probable. Progressing through the ranked list sequentially, we label each ACC as a premise if it meets two conditions. First, it must not overlap with any previously-labeled ACC. Second, its probability of being a premise according to the ACI classifier must be greater than both its probability of being a claim and its probability of being a major claim. For each ACC we label as a premise, we add a support relationship between the premise and the ACC we labeled as the paragraph's claim.

6. Evaluation

In this section we discuss our evaluation of our unsupervised approach to argument mining.

6.1. Experimental Setup

Corpus. As mentioned before, we use as our corpus the 402 essays annotated with argumentative discourse structures by S&G. All of our experiments are conducted via five-fold cross-validation on this corpus. For the supervised baselines, we reserve 60% of the essays for training, 20% for development, and 20% for testing in each fold experiment. For our unsupervised approach, no annotated data is needed for training and development.

Evaluation metrics. To calculate f-score on each task, we need to explain what constitutes a true positive, false positive, or false negative on each task. Given that j is a true argument component and i is an ACC, the formulas for the ACI task are:

$$TP = \Big|\{j \mid \exists i \,|gl(j) = pl(i) \wedge i \doteq j\}\Big| \quad (3)$$

$$FP = \Big|\{i| \; pl(i) \neq n \wedge \nexists j \mid gl(j) = pl(i) \wedge i \doteq j\}\Big| \quad (4)$$

$$FN = \Big|\{j \mid \nexists i \,|gl(j) = pl(i) \wedge i \doteq j\}\Big| \quad (5)$$

where $gl(j)$ is the gold label of j, $pl(i)$ is the predicted label of i, n is the non-argumentative class, and $i \doteq j$ means i is a *match* for j. Two text spans i and j are considered an *exact match* if they have exactly the same boundaries, whereas they are considered an *approximate match* if they share over half their tokens.

We perform most of our analysis on approximate match results rather than exact match results as it can be difficult even for human annotators to identify exactly the same boundaries for an argument component.[5] We use the same formulas for calculating these numbers for RI except that j and i represent a true relation and an RC respectively, two relations approximately (exactly) match if both their source and target ACCs approximately (exactly) match, and n is the no-relation class.

6.2. Results and Discussion

Approximate and exact match results of the pipeline approach (PIPE), the joint inference approach (ILP), and our unsupervised approach (U) are shown in Table 4. In addition, we show the results taken verbatim from Eger et al. (2017).[6] Briefly, by employing a neural approach to argument mining that jointly performs ACI and RI using a sequence tagging model (LSTM-CRF), Eger et al. have achieved the best supervised results to date on this dataset. Note that Eger et al.'s partition of the available essays into a training set and a test set is different from ours; in particular, unlike us, they did not perform any cross-validation experiments. Hence, their results can be used as a reference point but are not directly comparable to ours.

As we can see, using approximate matching, our unsupervised system achieves better results than the two supervised baseline by a variety of measures. The most important of these results is shown in the last column, where our unsupervised approach, despite using no labeled training data,

[5] Approximate match has been used in evaluating opinion mining systems (e.g., Choi et al. (2006), Yang and Cardie (2013)), where researchers have also reported difficulties in having human annotators identify exactly the same boundaries for an opinion expression and its sources and targets.

[6] Only ACI and RI F-scores are reported in Eger et al.'s (2017) paper.

	System	ACI MC-F	C-F	P-F	P	R	F	RI S-F	A-F	P	R	F	Avg F
Approx	PIPE	44.0	36.1	69.8	75.2	49.3	59.6	20.1	00.0	22.3	17.4	19.5	39.5
	ILP	50.7	42.6	76.7	61.0	66.9	63.8	32.5	01.4	23.9	47.5	31.8	47.8
	U	24.9	**42.2**	**77.2**	**73.5**	57.6	**64.6**	**37.9**	**00.0**	**38.2**	35.5	**36.8**	**50.7**
	Eger et al. (2017)	–	–	–	–	–	77.2	–	–	–	–	50.1	63.7
Exact	PIPE	39.6	33.9	61.7	64.0	45.8	53.4	17.2	00.0	18.6	15.2	16.7	35.1
	ILP	41.7	34.9	63.5	50.6	55.1	52.7	23.0	00.0	16.9	33.4	22.4	37.6
	U	19.0	**36.4**	59.6	57.7	45.2	50.7	**26.0**	**00.0**	**26.2**	24.3	**25.2**	**38.0**
	Eger et al. (2017)	–	–	–	–	–	70.8	–	–	–	–	45.5	58.2

Table 4: Results of argument component identification (ACI) and relation identification (RI). Column abbreviations are major claim F-score (MC-F), claim F-score (C-F), premise F-score (P-F), Precision (P), Recall (R), F-score (F), support F-score (S-F), and attack F-score (A-F).

	System	ACI MC-F	C-F	P-F	P	R	F	RI S-F	A-F	P	R	F	Avg F
Approx	H	49.7	30.7	74.8	76.2	53.1	62.6	00.0	00.0	00.0	00.0	00.0	31.3
	No ST	27.2	41.8	76.8	73.6	57.1	64.3	37.6	0.0	38.2	35.0	36.5	50.4
Exact	H	29.1	26.8	59.1	59.1	41.2	48.6	00.0	00.0	00.0	00.0	00.0	24.3
	No ST	19.3	36.3	59.4	57.7	44.8	50.5	26.3	0.0	26.7	24.4	25.5	38.0

Table 5: Results of two ablated versions of our unsupervised approach for ACI and RI. Column abbreviations are the same as those in Table 4.

achieves performance that is 2.9% higher than that of the better baseline (the ILP baseline). This column measures the average f-score between the two subtasks (ACI and RI).

Our unsupervised system also achieves results that are better than the better baseline on each of the subtasks, scoring 5.0% and 0.8% more in f-score than the ILP system on the RI task and the ACI task, respectively. The other measures on which our unsupervised system's approximate performance is similar to the better baseline are claim f-score, premise f-score, and attack f-score.

Due to the difficulty of heuristically detecting an AC's exact boundaries, our unsupervised system's exact matching performance is only marginally better than the baseline performances. By the most important measure, however, our unsupervised system scores only 0.4% more than the ILP baseline system in absolute average f-score.

While our unsupervised system performs as least as well as the two supervised baselines, it is outperformed by Eger et al.'s supervised system. Using approximate matching, our system achieves 83.7% of Eger et al.'s F-score on ACI and 73.5% of their F-score on RI. Using exact matching, our system achieves 71.6% of Eger et al.'s F-score on ACI and 55.3% of their F-score on RI. These results suggest that the performance gap between our system and Eger et al.'s system is bigger for approximate matching than for exact matching. This should perhaps not be surprising: as mentioned before, identifying argument component boundaries without labeled data is a very challenging task. Overall, we believe that our being able to achieve 83.7% (ACI) and 73.5% (RI) of Eger et al.'s F-scores (w.r.t. approximate matching) *without* using any labeled data is a result that would be of interest to argument mining researchers. In particular, our unsupervised approach can be fairly easily adapted to identify argumentative stuctures in essays written in languages for which argument-annotated data is not readily available.

6.3. Additional Experiments

Recall that our unsupervised approach is composed of four steps. To better understand the contribution of the various steps, we conduct two experiments. In the "No Self-Training" (No ST) experiment, we rerun our unsupervised system without step 3 (self-training), so it will allow us to assess the usefulness of self-training. In the "Heuristic" (H) experiment, we only run step 1 of our approach, so it will allow us to assess the contribution of the heuristics.

Results of these two experiments are shown in Table 5. A few points deserve mention. First, since there are no heuristic rules for identifying relationships without using an ACI classifier's output, the RI results of the H experiment are all 0.0. Second, note that the difference in the ACI results between the H experiment and the No ST experiment can be attributed entirely to step 2 (classifier training). As we can see, step 2 improved ACI f-score by $1.7-1.9$ points. Finally, comparing the No ST results with the U results in Table 4, we can see that leaving out self-training only causes ACI and RI f-scores to drop by no more than 0.3 points. Moreover, comparing the H results with the U results reveals that step 1 is largely responsible for the ACI results achieved by our approach. Overall, these results suggest that the data automatically annotated during self-training provided little knowledge beyond that already acquired in the first two steps of our approach.

6.4. Error Analysis and Future Work

It is obvious from Table 4 that our system has considerably more difficulty with the RI task than with the ACI task, regardless of whether evaluation occurs in an exact or approximate match setting. Indeed, this is also true of the baseline systems. The RI task is inherently more difficult than the ACI task because, in order to correctly identify a relationship, an argument mining system must also correctly identify the two participating argument components.

While this observation is correct, what it misses is that there do not currently exist high quality features for RI that perform better than heuristically constructing a support relationship between every premise and some claim in the same paragraph. Additional experiments revealed that removing all of the features designed by S&G specifically for the RI task from an argument mining system results in only a tiny drop in performance. The reason for this is that, unlike ACs, argumentative relations are often not triggered by discourse markers in the text. This is largely because argumentative relationships can occur between ACs in non-adjacent sentences. Thus a deeper semantic understanding of ACs is necessary in order to develop features or heuristics for detecting argumentative relationships between them.

While our unsupervised approach outperforms the two baselines on both subtasks w.r.t. the approximate metric, it underperforms them on the ACI task w.r.t. the exact metric. This is largely due to the fact that it is difficult to develop heuristics for finding the exact boundaries of an AC. There are simply too many ways for an author to introduce and trigger the termination of an argument component. Indeed, this is why we needed to develop the complicated boundary detection rules illustrated in Table 2. This is a subtask on which some annotated training data may be useful, so a semi-supervised approach that focuses on annotating essays having a wide variety of sentence structures may help improve our exact matching performances most.

7. Conclusion

We developed a novel unsupervised approach to the challenging task of end-to-end argument mining in persuasive student essays. In an evaluation on version 2 of Stab and Gurevych's corpus of 402 argument-annotated essays, our unsupervised approach rivaled two supervised baselines in performance and achieved 73.5–83.7% of the performance of a state-of-the-art neural approach. We believe that these results would be of interest to argument mining researchers, as our approach can be fairly easily adapted to identify argumentative structures in essays written in languages for which argument-annotated data is not readily available.

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

Cabrio, E. and Villata, S. (2018). Five years of argument mining: A data-driven analysis. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 5427–5433.

Choi, Y., Breck, E., and Cardie, C. (2006). Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 431–439, Sydney, Australia, July. Association for Computational Linguistics.

Eger, S., Daxenberger, J., and Gurevych, I. (2017). Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22.

Falakmasir, M. H., Ashley, K. D., Schunn, C. D., and Litman, D. J. (2014). Identifying thesis and conclusion statements in student essays to scaffold peer review. In *Intelligent Tutoring Systems*, pages 254–259. Springer International Publishing.

Florou, E., Konstantopoulos, S., Koukourikos, A., and Karampiperis, P. (2013). Argument extraction for supporting public policy formulation. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 49–54, Sofia, Bulgaria, August. Association for Computational Linguistics.

Goudas, T., Louizos, C., Petasis, G., and Karkaletsis, V. (2015). Argument extraction from news, blogs, and the social web. *International Journal on Artificial Intelligence Tools*, 24(5).

Granger, S., Dagneaux, E., Meunier, F., and Paquot, M. (2009). *International Corpus of Learner English (Version 2)*. Presses universitaires de Louvain.

Lawrence, J. and Reed, C. (2019). Argument mining: A survey. *Computational Linguistics*, 45(4):765–818.

Levy, R., Bilu, Y., Hershcovich, D., Aharoni, E., and Slonim, N. (2014). Context dependent claim detection. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 1489–1500.

Lippi, M. and Torroni, P. (2015). Context-independent claim detection for argument mining. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 185–191.

Lippi, M. and Torroni, P. (2016). Argument mining from speech: Detecting claims in political debates. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

McCallum, A. K. (2002). MALLET: A Machine Learning for Language Toolkit. `http://mallet.cs.umass.edu`.

Moens, M.-F., Boiy, E., Palau, R., and Reed, C. (2007). Automatic detection of arguments in legal texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*, ICAIL '07, pages 225–230, New York, NY, USA. ACM.

Nguyen, H. and Litman, D. (2016). Context-aware argumentative relation mining. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1127–1137, Berlin, Germany, August. Association for Computational Linguistics.

Ong, N., Litman, D., and Brusilovsky, A. (2014).

Ontology-based argument mining and automatic essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 24–28, Baltimore, Maryland, June. Association for Computational Linguistics.

Park, J. and Cardie, C. (2014). Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland, June. Association for Computational Linguistics.

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

Persing, I. and Ng, V. (2016). 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, San Diego, California, June. Association for Computational Linguistics.

Rinott, R., Dankin, L., Alzate Perez, C., Khapra, M. M., Aharoni, E., and Slonim, N. (2015). 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.

Rooney, N., Wang, H., and Browne, F. (2012). Applying kernel methods to argumentation mining.

Sardianos, C., Katakis, I. M., Petasis, G., and Karkaletsis, V. (2015). Argument extraction from news. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 56–66.

Song, Y., Heilman, M., Beigman Klebanov, B., and Deane, P. (2014). Applying argumentation schemes for essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 69–78, Baltimore, Maryland, June. Association for Computational Linguistics.

Stab, C. and Gurevych, I. (2014). Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar, October. Association for Computational Linguistics.

Stab, C. and Gurevych, I. (2016). Parsing argumentation structures in persuasive essays. Technical report, arXiv preprint arXiv:1604.07370.

Swanson, R., Ecker, B., and Walker, M. (2015). Argument mining: Extracting arguments from online dialogue. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 217–226, Prague, Czech Republic, September. Association for Computational Linguistics.

Teufel, S. (1999). *Argumentative Zoning: Information Extraction from Scientific Text*. Ph.D. thesis, University of Edinburgh,.

Wei, Z., Li, C., and Liu, Y. (2017). A joint framework for argumentative text analysis incorporating domain knowledge. Technical report, arXiv preprint arXiv:1701.05343.

Yang, B. and Cardie, C. (2013). Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1640–1649, Sofia, Bulgaria, August. Association for Computational Linguistics.