End-to-End Argumentation Mining in Student Essays

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

Abstract

Understanding the argumentative structure of a persuasive essay involves addressing two challenging tasks: identifying the components of the essay's argument and identifying the relations that occur between them. We examine the under-investigated task of *end-to-end* argument mining in persuasive student essays, where we (1) present the first results on end-to-end argument mining in student essays using a *pipeline* approach; (2) address error propagation inherent in the pipeline approach by performing *joint inference* over the outputs of the tasks in an Integer Linear Programming (ILP) framework; and (3) propose a novel objective function that enables F-score to be maximized directly by an ILP solver. We evaluate our joint-inference approach with our novel objective function on a publicly-available corpus of 90 essays, where it yields an 18.5% relative error reduction in F-score over the pipeline system.

1 Introduction

There has been a surge of interest in argumentation mining in recent years. 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 a first step towards mining arguments in persuasive essays, Stab and Gurevych (S&G) annotated a corpus of 90 student essays with argument components and their relations (Stab and Gurevych, 2014a). To illustrate, consider the following excerpt from one essay:

> From this point of view, I firmly believe that (1) we should attach more importance to cooperation during primary education. First of all, (2) through cooperation, children can learn about interpersonal skills which are significant in the future life of all students. (3) What we acquired from team work is not only how to achieve the same goal with others but more importantly, how to get along with others.

In this example, premise (3) supports claim (2), which in turn supports major claim (1).

Using their annotated corpus, S&G presented initial results on *simplified* versions of the ACI and RI tasks (Stab and Gurevych, 2014b). Specifically, they applied their learned ACI classifier 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* essay.

Motivated by this weakness, we examine in this paper argument mining in persuasive student essays in a considerably more challenging setting than that of S&G: the *end-to-end* setting. In other words, we

Proceedings of NAACL-HLT 2016, pages 1384–1394,
San Diego, California, June 12-17, 2016.

perform argument mining on raw, unannotated essays. Our work makes three contributions. First, we present the first results on end-to-end argument mining in student essays using a *pipeline* approach, where the ACI task is performed prior to the RI task. Second, to avoid the error propagation problem inherent in the pipeline approach, we perform *joint inference* over the outputs of the ACI and RI classifiers in an Integer Linear Programming (ILP) framework (Roth and Yih, 2004), where we design constraints to enforce global consistency. Finally, we argue that the typical objective function used extensively in ILP programs for NLP tasks is not ideal for tasks whose primary evaluation metric is F-score, and subsequently propose a novel objective function that enables F-score to be maximized directly in an ILP framework. We believe that the impact of our work goes beyond argument mining, as our F-score optimizing objective function is general enough to be applied to any ILP-based joint inference tasks.

2 Related Work

Recall that identifying argumentative discourse structures consists of (1) identifying the locations and types of the argument components, and (2) identifying how they are related to each other. Below we divide related works into five broad categories based on which of these subtasks they addressed.

Argument location identification. Works in this category aimed to classify whether a sentence contains an argument or not (Florou et al., 2013; Moens et al., 2007; Song et al., 2014; Swanson et al., 2015). The usefulness of existing works is somewhat limited by the task's coarseness: it won't tell us which portion of a potentially long sentence contains the argument, for instance, but it can serve as a potentially useful first step in argument mining.

Argument component typing. Works in this category focused on determining the *type* of an argument. The vast majority of previous works perform argument component typing at the *sentence* level. For instance, Rooney et al. (2012) classified sentences into premises, conclusions, premise-conclusions, and non-argumentative components; Teufel (1999) classified each sentence into one of seven rhetorical classes (e.g., claim, result, purpose); Burstein et al. (2003), Ong et al. (2014), and Falakmasir et al. (2014) assigned argumentative labels (e.g., claim, thesis, conclusion) to an essay's sentences; Levy et al. (2014) detected sentences that support or attack an article's topic; Lippi and Torroni (2015; 2016) detected sentences containing claims; and Rinott et al. (2015) detected sentences containing evidence for a given claim. Sentence-level argument component typing has limitations, however. For example, it can identify sentences containing claims, but it cannot tell how many claims a sentence has or where in the sentence they are.

Argument location identification and typing. Some works focused on the more difficult task of *clause-level* argument component typing (Park and Cardie, 2014; Goudas et al., 2015; Sardianos et al., 2015), training a Conditional Random Field to jointly identify and type argument components.

Argument component typing and relation identification. Given the difficulty of clause-level argument component location identification, recent argument mining works that attempted argument component typing and relation identification are not end-to-end. Specifically, they simplified the task by assuming as input *gold* argument components (Stab and Gurevych, 2014b; Peldszus and Stede, 2015).

End-to-end argument mining. To our knowledge, only Palau and Moens (2009) addressed all the argument mining subtasks. They employed a hand-crafted context-free grammar (CFG) to generate (i.e., extract and type) argument components at the clause level and identify the relations between them. A CFG approach is less appealing in the essay domain because (1) constructing a CFG is a time- and labor-intensive task, (2) which would be more difficult in the less-rigidly structured essay domain, which contains fewer rhetorical markers indicating component types (e.g. words like "reject", or "dismiss" which indicate a legal document's conclusion); and (3) about 20% of essay arguments' structures are non-projective (i.e., when mapped to the ordered text, their argument trees have edges that cross), and thus cannot be captured by CFGs.

3 Corpus

Our corpus consists of 90 persuasive essays collected and annotated by S&G. Some relevant statistics are shown in Table 1. Each essay is an average

Essays: 90	Paragraphs: 417	Sentences: 1,673
Major Claims: 90	Claims: 429	Premises: 1,033
Support Relations: 1,312		Attack Relations: 161

Table 1: Corpus statistics.

of 4.6 paragraphs (18.6 sentences) in length and is written in response to a topic such as "should high school make music lessons compulsory?" or "competition or co-operation-which is better?".

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 **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 Pipeline-Based Argument Mining

Next, we describe our end-to-end *pipeline* argument mining system, which will serve as our baseline. In this system, ACI is performed prior to RI.

4.1 Argument Component Identification

We employ a two-step approach to the ACI task, where we first heuristically extract *argument component candidates* (ACCs) from an essay, and then classify each ACC as either a premise, claim, major claim, or non-argumentative, as described below.

4.1.1 Extracting ACCs

We extract ACCs by constructing a set of low precision, high recall heuristics for identifying the locations in each sentence where an argument component's boundaries might occur. The majority of these rules depend primarily on a syntactic parse tree we automatically generated for all sentences in the corpus using the Stanford CoreNLP (Manning et al., 2014) system. Since argument components are a clause-level annotation and therefore a large majority of annotated argument components are substrings of a simple declarative clause (an S node in the parse tree), we begin by identifying each S node in a sentence's tree.

(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.[1]
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.

Given an S clause, we collect a list of left and right boundaries where an argument component may begin or end. The rules we used to find these boundaries are summarized in Table 2. We then construct ACCs by combining each left boundary with each right boundary that occurs after it. As a result, we are able to find exact (boundaries exactly match) and approximate (over half of tokens shared) matches for 92.1% and 98.4% respectively of all ACs.

4.1.2 Training the ACI Classifier

We train a classifier for ACI using MALLET's (McCallum, 2002) implementation of maximum entropy classification. We create a training instance from each ACC extracted above. If the ACC's left and right endpoints exactly match an annotated argument component's, the corresponding training instance's class label is the same as that of the component. Otherwise, its class label is "non-

[1] An additional point that requires explanation is that the last two right boundary rules mention "possibly nested" nodes. In boundary rule 7, for example, this means that the S node might end in a PP node, which itself has a PP node as its last child, and so on. We generate a separate right boundary immediately before each of these PP nodes.

argumentative". Each training instance is represented using S&G's structural, lexical, syntactic, indicator, and contextual features for solving the same problem. Briefly, the structural features describe an ACC and its covering sentence's length, punctuations, and location in the essay. Lexical features describe the $1-3$ grams of the ACC and its covering sentence. Syntactic features are extracted from the ACC's covering sentence's parse tree and include things such as production rules. Indicator features describe any explicit connectives that immediately precede the ACC. Contextual features describe the contents of the sentences preceding and following the ACC primarily in ways similar to how the structural features describe the covering sentence.

4.2 Relation Identification

We consider RI between pairs of argument components to be 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. Below we describe how we train and apply our classifier for RI.

We learn our RI classifier using MALLET's implementation of maximum entropy classification. Each training instance, which we call a training *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.

We create training RCs as follows. We construct RCs corresponding to true relations out of all pairs of argument components in a training essay having a gold relation. As the number of potential RCs far exceeds the number of gold relations in an essay, we undersample the "no relation" class in the following way. Given a pair of argument components A and B between which there is a gold relation, we define A_p to be the closest previous ACC in the essay as generated in Section 4.1.1 such that A_p's text doesn't overlap with A. We also define A_s as the closest succeeding ACC after A such that A_s and A do not overlap. We define B_p and B_s similarly with respect to B. From these ACCs, we generate the the four instances (A_p, B), (A_s, B), (A, B_p), and (A, B_s), all of which have the "no relation" label, as long as the pairs' text sequences do not overlap. We believe the resulting "no relation" training instances are informative since each one is "close" to a gold relation. We represent each instance using S&G's structural, lexical, syntactic, and indicator features for solving the same problem. Briefly, RC structural features describe many of the same things about each ACC as did the ACC structural features, though they also describe the difference between the ACCs (e.g. the difference in punctuation counts). Lexical features consist primarily of the unigrams appearing in each ACC and word pairs, where each word from one ACC is paired with each word from the other. Syntactic and indicator features encode the same information about each ACC as the ACC syntactic and indicator features did.

We now apply the classifier to test essay RCs, which are created as follows. Given that this is a pipelined argument mining system, in order to ensure that the RI system's output is consistent with that of the ACI system, we generate test RCs from all possible pairs of ACCs that the ACI system predicted are real components (i.e. it labeled them something other than "non-argumentative").

5 Joint Inference for Argument Mining

5.1 Motivation

There are two major problems with the pipeline approach described in the previous section. First, many essay-level within-task constraints are not enforced. For instance, the ACI task has the constraint that each essay has exactly one major claim, and the RI task has the constraint that each claim has no more than one parent. This problem arises because our ACI and RI classifiers, like those of S&G, classify each ACI and RI test instance independently of other test instances. We propose to enforce such essay-level within-task constraints in an ILP framework, employing ILP to perform joint inference over the outputs of our ACI and RI classifiers so that the resulting classifications satisfy these constraints.[2]

[2]Note that while we partition *documents* into folds in our cross-validation experiments, S&G partition *instances* into folds. Hence, S&G's evaluation setting prevents them from enforcing essay-level constraints in addition to being unrealistic in practice.

The second problem with the pipeline approach is that errors made early on in the pipeline propagate. For instance, assume that a Support relation exists between two argument components in a test essay. If the pipeline system fails to (heuristically) extract one or both of these argument components, or if it successfully extracts them but misclassifies one or both of them as non-argumentative, then the pipeline system will not be able to identify the relationship between them because no test RCs will be created from them. The above problem arises because the RI classifier assumes the *most probable* output of the ACI classifier for each ACC as input. Hence, one possible solution to this problem is to make use of the *n-best* outputs of the ACI classifier for each argument component type, as this increases the robustness of the pipeline to errors made by the ACI classifier.

We obtain the n-best ACI outputs and employ them as follows. Recall that the ACI system uses a maximum entropy classifier, and therefore its output for each ACC is a list of probabilities indicating how likely it is that the ACC belongs to each of the four classes (premise, claim, major claim, or non-argumentative). This means that it is possible to rank all the ACCs in a text by these probabilities. We use this idea to identify (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.[3] Given these most likely ACC lists, we combine pairs of ACCs into test RCs for the RI classifier in the following way. As long as the ACCs do not overlap, we pair (1) each likely premise ACC with every other likely ACC of any type occurring in the same paragraph, and (2) each likely claim ACC with each likely major claim ACC. We then present these test RCs to the RI classifier normally, making no other changes to how the pipeline system works.

Employing n-best ACI outputs, however, introduces another problem: the output of the RI classifier may no longer be *consistent* with that of the ACI classifier because the RI classifier may posit a relationship between two ACCs that the ACI classifier labeled non-argumentative. To enforce this cross-task consistency constraint, we also propose to employ ILP. The rest of this section details our ILP-based joint inference approach for argument mining, which addresses both of the aforementioned problems with the pipeline approach.

5.2 Basic ILP Approach

We perform joint inference over the outputs of the ACI and RI classifiers by designing and enforcing within-task and cross-task constraints in the ILP framework. Specifically, we create one ILP program for each test essay, as described below.

Let Xn_i, Xp_i, Xc_i, and Xm_i be binary indicator variables representing whether the ILP solver believes ACC i has type none, premise, claim, and major claim, respectively. Let Cn_i, Cp_i, Cc_i, and Cm_i be the probabilities that ACC i has type none, premise, claim, and major claim, respectively, as dictated by the ACI maximum entropy classifier's output.[4] Let a be the count of ACCs.

Let $Yn_{i,j}$, $Ys_{i,j}$, $Ya_{i,j}$, $Yrs_{i,j}$, and $Yra_{i,j}$ be binary indicator variables representing whether the ILP solver believes ACCs i and j have no relation, i is supported by j, i is attacked by j, j is supported by i, and j is attacked by i, respectively, where (i, j) appears in the set of RCs B that we presented to the RI system as modified in Section 5.1. We assume all other ACC pairs have no relation. Let $Dn_{i,j}$, $Ds_{i,j}$, $Da_{i,j}$, $Drs_{i,j}$, and $Dra_{i,j}$ be the probabilities that component candidates i and j have no relation, i is supported by j, i is attacked by j, j is supported by i, and j is attacked by i, respectively, as dictated by the modified RI classifier described in Section 5.1.[4]

Given these definitions and probabilities, our ILP program's default goal is to find an assignment of these variables X and Y in order to maximize $P(X) + P(Y)$, where:

$$P(X) = \frac{1}{a}\sum_{i=1}^{a} \log(Cn_i Xn_i + Cp_i Xp_i + Cc_i Xc_i + Cm_i Xm_i) \quad (1)$$

[3] We select more premise than claim ACCs (3 per sentence vs 5 per paragraph) because the corpus contains over twice as many premises as claims. We select major claim ACCs only from the first and last paragraph because major claims never occur in middle paragraphs.

[4] We additionally reserve .001 probability mass to distribute evenly among Cn_i, Cp_i, Cc_i, and Cm_i (or $Dn_{i,j}$, $Ds_{i,j}$, $Da_{i,j}$, $Drs_{i,j}$, and $Dra_{i,j}$) to prevent math errors involving taking the log of 0 which might otherwise occur in the formulas below.

$$P(Y) = \frac{1}{|B|} \sum_{(i,j)\in B} \log(Dn_{i,j}Yn_{i,j} + Ds_{i,j}Ys_{i,j} + Da_{i,j}Ya_{i,j} + Drs_{i,j}Yrs_{i,j} + Dra_{i,j}Yra_{i,j}) \quad (2)$$

subject to the *integrity* constraints that: (3) an ACC is either not an argument component or it has exactly one of the real argument component types, (4) a pair of component candidates (i, j) must have exactly one of the five relation types, and (5) if there is a relation between ACCs i and j, i and j must each be real components.[5]

$$Xn_i + Xp_i + Xc_i + Xm_i = 1 \quad (3)$$

$$Yn_{i,j} + Ys_{i,j} + Ya_{i,j} + Yrs_{i,j} + Yra_{i,j} = 1 \quad (4)$$

$$(Xp_i + Xc_i + Xm_i) + (Xp_j + Xc_j + Xm_j) - 2(Ys_{i,j} + Ya_{i,j} + Yrs_{i,j} + Yra_{i,j}) \geq 0 \quad (5)$$

In the objective function, a and $|B|$ serve to balance the contribution of the two tasks, preventing one from dominating the other.

5.3 Enforcing Consistency Constraints

So far we have described integrity constraints, but recall that our goal is to enforce consistency by imposing within-task and cross-task constraints, which force the ILP solutions to more closely resemble real essay argument structures. Our consistency constraints fall into four categories.

Our constraints on *major claims* are that: (6) there is exactly one major claim in each essay, (7) major claims always occur in the first or last paragraph, and (8) major claims have no parents.

$$\sum_{i=1}^{a} Xm_i = 1 \quad (6)$$

$$Xm_i = 0 \mid i \notin first\ or\ last\ paragraph \quad (7)$$

$$Ys_{i,j} + Xm_j \leq 1\ ,\ Ya_{i,j} + Xm_j \leq 1 \quad Yrs_{i,j} + Xm_i \leq 1\ ,\ Yra_{i,j} + Xm_i \leq 1 \quad (8)$$

Our constraints on *premises* are that: (9) a premise has at least one parent, and (10) a premise is related only to components in the same paragraph.

$$\sum_{\{i|(i,j)\in B\}} (Ys_{i,j} + Ya_{i,j}) + \sum_{\{k|(j,k)\in B\}} (Yrs_{j,k} + Yra_{j,k}) - Xp_j \geq 0 \quad (9)$$

$$for\ i < j,\ i \notin Par(j) : Xp_j - Yn_{i,j} \leq 0 \quad for\ k > j,\ k \notin Par(j) : Xp_j - Yn_{j,k} \leq 0 \quad (10)$$

where $i < j$ and $j < k$ mean ACC i appears before j, and j appears before k, and $Par(j)$ is the set of ACCs in j's covering paragraph.

Our constraints on *claims* state that: (11) a claim has no more than one parent[6], and (12) if a claim has a parent, that parent must be a major claim.

$$\Big(\Big|\{i|(i,j)\in B\}\Big| + \Big|\{k|(j,k)\in B\}\Big|\Big)Xc_j + \sum_{\{i|(i,j)\in B\}} (Ys_{i,j} + Ya_{i,j}) + \sum_{\{k|(j,k)\in B\}} (Yrs_{j,k} + Yra_{j,k}) \leq \Big|\{i|(i,j)\in B\}\Big| + \Big|\{k|(j,k)\in B\}\Big| + 1 \quad (11)$$

$$for\ j < k,\ Xm_k - Xc_j - Yrs_{j,k} - Yra_{j,k} \geq -1 \quad for\ j > i,\ Xm_i - Xc_j - Ys_{i,j} - Ya_{i,j} \geq -1 \quad (12)$$

The last category, which comprises constraints that do not fit well into any other category, are: (13) the boundaries of actual components never overlap, (14) each paragraph must have at least one claim or major claim, and (15) each sentence may have at most two argument components.[7]

$$for\ i\ overlaps\ j\ ,\ Xn_i + Xn_j \geq 1 \quad (13)$$

$$\forall\ paragraphs\ P\ :\ \sum_{i\in P} Xc_i + Xm_i \geq 1 \quad (14)$$

$$\forall\ sentences\ S\ :\ \sum_{i\in S} Xp_i + Xc_i + Xm_i \leq 2 \quad (15)$$

We solve each ILP program using Gurobi.[8]

[5] Note that because of previous integrity constraints, the first term in constraint 5 is 1 only if we predict that i is a real component, the second term is 1 only if we predict that j is a real component, and the third term is -2 only if we predict that there is a relationship between them. Otherwise, each term is 0. Thus term 3 prevents us from predicting a relationship between i and j unless the first and second terms cancel it out through predicting that i and j are real components.

[6] The coefficient of Xc_j is equal to the number of terms in the two summations on the left hand side of the equation. The intention is that component j's claim status should have equal weight with the relations it might potentially participate in, so it is possible for j to participate as a child in a relation with all other available components unless it is a claim, in which case it can participate in at most one relation as a child.

[7] Unlike the other constraints, the last two constraints are only mostly true: 5% of paragraphs have no claims or major claims, and 1.2% of sentences have 3 or more components.

[8] `http://www.gurobi.com`

5.4 F-score Maximizing Objective Function

The objective function we employ in the previous subsection attempts to maximize the average probability of correct assignment of variables over the ACI and RI problems. This kind of objective function, which aims to maximize classification accuracy, was originally introduced by Roth and Yih (2004) in their seminal ILP paper, and has since then been extensively applied to NLP tasks. However, it is arguably not an ideal objective function for our task, where F-score rather than classification accuracy is used as the evaluation metric.

In this section, we introduce a novel method for constructing an ILP objective function that directly maximizes the average F-score over the two problems. Recall that F-score can be simplified to:

$$F = \frac{2TP}{2TP + FP + FN} \tag{16}$$

where TP, FP, and FN are the counts of true positives, false positives, and false negatives respectively. Unfortunately, we cannot use this equation for F-score in an ILP objective function for two reasons. First, this equation involves division, which cannot be handled using ILP since ILP can only handle linear combinations of variables. Second, TP, FP, and FN need to be computed using gold annotations, which we don't have in a test document. We propose to instead maximize F by maximizing the following:

$$G = \alpha 2TP_e - (1-\alpha)(FP_e + FN_e) \tag{17}$$

where TP_e, FP_e, and FN_e, are *estimated* values for TP, FP, and FN respectively, and α attempts to balance the importance of maximizing the numerator vs minimizing the denominator.[9] We ignore the $2TP$ term in the denominator because minimizing it would directly reduce the numerator.

To maximize average F-score, we can therefore attempt to maximize the function $\frac{G_c+G_r}{2}$, where G_c and G_r are the values of G in equation 17 as calculated using the estimated values from the ACI and RI problem respectively.

The question that still remains is, how can we estimate values for TP, FP, and FN mentioned in Equation 17? Our key idea is inspired by the E-step of the Expectation-Maximization algorithm (Dempster et al., 1977): while we cannot compute the actual TP, FP, and FN due to the lack of gold annotations, we can compute their *expected* values using the probabilities returned by the ACI and RI classifiers. Using the notation introduced in Section 5.2, the expected TP, FP, and FN values for the ACI task can be computed as follows:

$$TP_e = \sum_{i,g} Cg_i Xg_i \tag{18}$$

$$FP_e = \sum_{i,g} (1 - Cg_i) Xg_i \tag{19}$$

$$FN_e = \sum_{i,g} \left(Xg_i \sum_{h \neq g} Ch_i \right) + \sum_{i} Xn_i (1 - Cn_i) \tag{20}$$

where g and h can be any argumentative class from the ACI problem (i.e. premise (p), claim (c), or major claim (m)). The formulas we use to calculate TP_e, FP_e, and FN_e for the RI problem are identical except C is replaced with D, X is replaced with Y, and g and h can be any class from the RI problem other than no-relation.

6 Evaluation

6.1 Experimental Setup

Corpus. As mentioned before, we use as our corpus the 90 essays annotated with argumentative discourse structures by S&G. All of our experiments are conducted via five-fold cross-validation on this corpus. In each fold experiment, we reserve 60% of the essays for training, 20% for development (selecting features and tuning α), and 20% for testing.

Evaluation metrics. To calculate F-score on each task using Equation 16, 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:

$$TP = \left| \{ j \mid \exists i \, | gl(j) = pl(i) \wedge i \doteq j \} \right| \tag{21}$$

$$FP = \left| \{ i | \, pl(i) \neq n \wedge \nexists j \mid gl(j) = pl(i) \wedge i \doteq j \} \right| \tag{22}$$

$$FN = \left| \{ j \mid \nexists i \, | gl(j) = pl(i) \wedge i \doteq j \} \right| \tag{23}$$

where $gl(j)$ is the gold standard 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. i and j are considered an *exact match* if they have exactly

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

		ACI						RI					Avg
	System	MC-F	C-F	P-F	P	R	F	S-F	A-F	P	R	F	F
Approx	BASE	11.1	26.9	51.9	64.0	33.6	44.0	6.1	0.8	5.7	6.2	5.8	24.9
	OUR	22.2	**42.6**	**66.0**	56.6	**57.9**	**57.2**	**21.3**	1.1	**16.8**	**28.0**	**20.4**	**38.8**
Exact	BASE	7.4	24.2	43.2	50.4	29.6	37.3	4.4	0.8	4.1	4.7	4.3	20.8
	OUR	16.9	**37.4**	**53.4**	47.5	**46.7**	**47.1**	**13.6**	0.0	12.7	15.4	**12.9**	**30.0**

Table 3: Five-fold cross-validation average percentages for argument component identification (ACI) and relation identification (RI) for OUR system and the pipeline-based BASEline system. 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).

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.[10] We use the same formulas for calculating these numbers for the RI problem 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 (BASE) and the joint approach (OUR) are shown in Table 3. As we can see, using approximate matching, OUR system achieves highly significant[11] improvements over the pipelined baseline system by a variety of measures.[12] The most important of these improvements is shown in the last column, where our system outperforms the baseline by 13.9% absolute F-score (a relative error reduction of 18.5%). This is the most important result because it most directly measures our performance in pursuit of our ultimate goal, to maximize the average F-score over both the ACI and RI problems. The highly significant improvements in other measures, particularly the improvements of 13.2% and 14.6% in ACI and RI F-score respectively, follow as a consequence of this maximization. Using exact matching, the differences in scores between OUR system and BASE's are smaller and highly significant with respect to a smaller number of measures. In particular, under Exact matching OUR system's performances on the RI-P and RI-R metrics are significant ($p < 0.02$), while under Approx matching, they are highly significant ($p < 0.002$).

6.3 Ablation Results

To analyze the performance gains yielded by each improvement to our system, we show ablation results in Table 4. Each row of the table shows the results of one ablation experiment on the test set. That is, we obtain them by removing exactly one feature set or improvement type from our system.

The **B**aseline feature sets we remove include those for the A**C**I task (C_b) from Section 4.1.2 and those for the **RI** task (R_b) from Section 4.2.[13] The **I**LP improvement sets we remove are the **D**efault ILP (I_d) system[14] from Section 5.2, the **M**ajor claim (I_m), **P**remise (I_p), **C**laim (I_c), and **O**ther (I_o) constraints from Section 5.3 Equations 6−8, 9−10,

[10] 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. They have adopted an even more relaxed notion of approximate match: they consider two text spans an approximate match if they share at least one overlapping token.

[11] Boldfaced results in Table 3 are highly significant ($p < 0.002$, paired t-test) compared to the baseline.

[12] All the results in Tables 3 and 4 are averaged across five folds, so it is not true that $F_{avg} = \frac{2P_{avg}R_{avg}}{P_{avg}+R_{avg}}$. Our F-score averaging method is preferable to calculating F-scores using the above formula because the formula can be exploited to give artificially inflated F-scores by alternating between high precision, low recall, and low precision, high recall labelings on different folds.

[13] When we remove a baseline feature set, we represent each instance to the corresponding classifier using no features. As a result, the classifier's predictions are based solely on the frequency of the classes seen during training.

[14] Note that removing the default ILP system (I_d) necessitates simultaneously removing all other ILP-related improvements. Thus, a system without it is equivalent to BASE, but with RCs generated as described in Section 5.1.

Mod	ACI			RI			Avg
	P	R	F	P	R	F	F
ALL	54.9	58.8	56.7	16.7	26.4	20.4	38.6
C_b	**44.0**	**38.5**	**40.9**	14.2	**14.8**	**14.4**	**27.7**
R_b	57.9	58.7	58.2	16.6	24.0	18.2	38.2
I_d	64.0	**33.6**	**44.0**	**6.6**	**21.8**	**10.1**	**27.0**
I_m	**44.6**	**46.7**	**45.6**	**14.7**	27.8	19.2	**32.4**
I_p	51.5	58.4	54.7	**13.2**	26.1	**17.3**	36.0
I_c	**49.0**	**51.5**	**50.2**	**14.2**	27.9	18.8	**34.5**
I_o	**40.5**	65.9	**50.1**	**7.0**	29.2	**11.2**	**30.7**
I_f	61.1	**40.4**	**48.5**	22.3	**15.7**	**18.2**	**33.4**

Table 4: Ablation results. How OUR system performs on one development set as measured by percent Precision, Recall, and F-score if each improvement or feature set is removed.

11−12, and 13−15 respectively, and the **F**-score maximizing objective function from Section 5.4.

Broadly, we see from the last column that all of our improvement sets are beneficial (usually significantly[15]) to the system, as performance drops with their removal. Notice also that whenever removing an ILP improvement set harms average F-score, it also simultaneously harms ACI and RI F-scores, usually significantly. This holds true even when the improvement set deals primarily only with one task (e.g. I_o for the ACI task), suggesting that our system is benefiting from joint inference over both tasks.

6.4 Error Analysis and Future Work

Table 3 shows that OUR system has more trouble with the RI task than the ACI task. A closer inspection of OUR system's RI predictions reveals that its low precision is mostly due to predicted relationships wherein one of the participating ACCs is not a true argument component. Since false positives in the ACI task have an outsized impact on RI precision, it may be worthwhile to investigate ILP objective functions that more harshly penalize false positive ACCs.

The RI task's poor recall has two primary causes. The first is false negatives in the ACI task. It is impossible for an RI system to correctly identify a relationship between two ACs if the ACI system fails to identify either one of them as an AC. We believe ACI recall, and by extension, RI recall, can be improved by exploiting the following observations. First, we noticed that many argument components OUR system fails to identify, regardless of their type, contain words that are semantically similar to words in the essay's topic (e.g., if the topic mentions "school", argument components might mention "students"). Hence, one way to improve ACI recall, and by extension, RI recall, would be to create ACI features using a semantic similarity measure such as the Wikipedia Link-based similarity measure (Milne and Witten, 2008). Second, major claims are involved in 32% of all relationships, but OUR system did an especially poor job at identifying them due to their scarcity. Since we noticed that major claims tend to include strong stancetaking language (e.g., words like "should", "must", and "believe"), it may be possible to improve major claim identification by constructing an arguing language lexicon as in Somasundaran and Wiebe (2010), then encoding the presence of any of these arguing words as ACC features.

The second major cause of OUR system's poor RI recall is its failure to identify relationships between two correctly extracted ACs. We noticed many of the missed relationships involve ACs that mention some of the same entities. Thus, a coreference resolver could help us build features that describe whether two ACCs are talking about the same entities.

7 Conclusion

We presented the *first* results on *end-to-end* argument mining in persuasive student essays using a pipeline approach, improved this baseline approach by designing and employing global consistency constraints to perform joint inference over the outputs of the tasks in an ILP framework and proposed a novel objective function that enables F-score to be maximized directly by an ILP solver. In an evaluation on Stab and Gurevych's corpus of 90 essays, our approach yields an 18.5% relative error reduction in F-score over the pipeline system.

Acknowledgments

We thank the three anonymous reviewers for their detailed comments. This work was supported in part by NSF Grants IIS-1147644 and IIS-1219142.

[15]Boldfaced results are significantly lower than ALL, the system with all improvements left intact, with $p < 0.05$.

References

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

Yejin Choi, Eric Breck, and Claire Cardie. 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.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38.

Mohammad Hassan Falakmasir, Kevin D. Ashley, Christian D. Schunn, and Diane J. Litman. 2014. Identifying thesis and conclusion statements in student essays to scaffold peer review. In *Intelligent Tutoring Systems*, pages 254–259. Springer International Publishing.

Eirini Florou, Stasinos Konstantopoulos, Antonis Koukourikos, and Pythagoras Karampiperis. 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.

Theodosios Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. 2015. Argument extraction from news, blogs, and the social web. *International Journal on Artificial Intelligence Tools*, 24(5).

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 1489–1500.

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

Marco Lippi and Paolo Torroni. 2016. Argument mining from speech: Detecting claims in political debates. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 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.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. `http://mallet.cs.umass.edu`.

David Milne and Ian Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, pages 25–30.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic detection of arguments in legal texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*, pages 225–230.

Nathan Ong, Diane Litman, and Alexandra Brusilovsky. 2014. Ontology-based argument mining and automatic essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 24–28.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38.

Andreas Peldszus and Manfred Stede. 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.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 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.

Niall Rooney, Hui Wang, and Fiona Browne. 2012. Applying kernel methods to argumentation mining. In *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference*.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, pages 1–8.

Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. 2015. Argument extraction from news. In *Proceedings of the Second Workshop on Argumentation Mining*, pages 56–66.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124.

Yi Song, Michael Heilman, Beata Beigman Klebanov, and Paul Deane. 2014. Applying argumentation

schemes for essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 69–78.

Christian Stab and Iryna Gurevych. 2014a. Annotating argument components and relations in persuasive essays. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 1501–1510.

Christian Stab and Iryna Gurevych. 2014b. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 46–56.

Reid Swanson, Brian Ecker, and Marilyn Walker. 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.

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

Bishan Yang and Claire Cardie. 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.