Duluth at SemEval-2021 Task 11: Applying DeBERTa to Contributing Sentence Selection and Dependency Parsing for Entity Extraction

Anna Martin & Ted Pedersen
Department of Computer Science
University of Minnesota
Duluth, MN 55812, USA
{mart5877,tpederse}@d.umn.edu

Abstract

This paper describes the Duluth system that participated in SemEval-2021 Task 11, NLP Contribution Graph. It details the extraction of contribution sentences and scientific entities and their relations from scholarly articles in the domain of Natural Language Processing. Our solution uses deBERTa for multi-class sentence classification to extract the contributing sentences and their type, and dependency parsing to extract phrases from each sentence and format into subject-predicate-object triples. Our system ranked fifth of seven for Phase 1: end-to-end pipeline, sixth of eight for Phase 2 Part 1: phrases and triples, and fifth of eight for Phase 2 Part 2: triples extraction.

1 Introduction

The rapid rate at which scientific literature grows makes it difficult to keep up with new research even in one's own field. Automated solutions are challenging since text formatted and written for human consumption does not lend itself to machine processing.

(Jaradeh et al., 2019) have proposed a knowledge graph based system for collecting and structuring articles in a machine-readable format. This requires the annotation of many scholarly articles, a task that is time consuming to do by hand. The purpose of this SemEval task (D'Souza et al., 2021) is to enable the construction of a scholarly contributions graph over English language NLP articles by automating the task of annotating scientific papers.

This annotation processes consists of:

1. selecting sentences that describe the contribution of the article and outputting them to a sentences.txt file,
2. extracting scientific entities and relations from the selected sentences and outputting them to an entities.txt file,
3. and structuring the entities and relations into triples of the form subject-predicate-object. These triples are sorted into one of twelve information units, which are labels that describe the type of contribution being made by the sentence outlined in the triples file.

Following is an example of the annotation process using a sentence taken from the training data with the human–labeled entities in braces:

> We [apply] [dropout] of [0.4] [to layers], [0.3] [to RNN layers], [0.4] [to input embedding layers], [0.05] [to embedding layers], and [weight dropout] of [0.5] [to the RNN hidden-to-hidden matrix].

This task requires the 1) identification of this sentence as a contribution sentence belonging to the information unit HYPERPARAMETERS, 2) the extraction of the entities in braces, and 3) the formatting of those entities into triples. This example and various others will be used throughout this paper to help contextualize our system description.

SemEval-2021 Task 11 was organized into three phases, called Phase 1: end-to-end pipeline, Phase 2 Part 1: phrases and triples, and Phase 2 Part 2: triples extraction. Phase 1 tested the entire system, scoring for sentence extraction, phrases extraction, and triples extraction. The gold sentences.txt files were released for use in Phase 2 Part 1, in order to test phrases and triples extraction given perfect sentence selection. The gold entities.txt files were released for use in Phase 2 Part 2, in order to test triples extraction given perfect phrases selection.

There are four datasets mentioned in this paper. Gold data refers to the sentences.txt, entities.txt, and triples folders released by the organizers after each phase of the task. Test data refers to the data used to test the system during evaluation phases. Training data refers to the entire training dataset provided by the organizers. The trial dataset is the

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 490–501
Bangkok, Thailand (online), August 5–6, 2021.

segment of the training dataset that we used to test the Duluth system during validation, before the first evaluation phase began. For more information on how this dataset was created, see Appendix B.

Our approach[1] employs a variety of techniques to address each component of the task. The selection of contribution sentences was done by fine-tuning the base deBERTa model (He et al., 2021) on the training data, as fine-tuned BERT (Devlin et al., 2019) models have been found to perform well on multi-class text classification tasks (Liu and Wangperawong, 2019). The fine-tuned model is used to classify each sentence as either non-contributing, or one of the twelve information units.

The selected contribution sentences were then tagged to indicate likely scientific entities using a Maximum Entropy Markov Model (MEMM) (Bird et al., 2009) that was trained on the noun phrases selected as entities in the training data.

A dependency parse (Manning et al., 2014) was then found for each sentence. The dependency parse and entity tags were then leveraged to select contributing phrase spans and triples by using the entity tags to determine whether a subject or object noun phrase ought to be considered, and using the dependencies to extract Subject–Predicate–Object patterns from sentence.

2 Previous Work

Two previous SemEval tasks were also concerned with the extraction of relations and key phrases from scientific publications: SemEval 2017 Task 10: (ScienceIE - Extracting Keyphrases and Relations from Scientific Publications) (Augenstein et al., 2017), and SemEval 2018 Task 7: (Semantic Relation Extraction and Classification in Scientific Papers) (Gábor et al., 2018).

Many of the approaches in these tasks use neural models to extract entities and their relations. The AI2 system (Ammar et al., 2017) at SemEval-2017 Task 10, which ranked first and second for task scenarios one and three respectively, approached this by building separate entity and relation models, each of which contain layers of LSTMs. The ETH-DS3Lab system (Rotsztejn et al., 2018) at SemEval-2018 Task 7, which ranked first in three of four subtasks, built an entity and relation classifier using a combination of RNNs and CNNs.

Other approaches used supervised machine learning algorithms while leveraging grammatical features. The LIPN system (Hernandez et al., 2017) approached SemEval-2017 Task 10 by first filtering possible keyphrases by labeling phrases with their POS sequence. Candidate keyphrases are filtered by comparing the POS tags of the phrase with POS sequences developed from the training data. They then trained a CRF model using the candidate phrases labeled with IOB tags. They were able to improve recall for keyphrase extraction by filtering candidate sentences before using a CRF.

3 Selection of Contribution Sentences

We approached the selection of a contribution sentence as a multi-class sentence classification problem with 13 classes, where each of the twelve information units is a class, and class 0 represents non-contributing sentences. Although the subtask of sentence extraction could be performed using a binary classifier to label sentences as either contributing or non-contributing, we decided to sort the sentences further into their information units. The alternative would require classifying phrases or triples further down the pipeline; the benefit to classifying the contributions during the sentence extraction step is that the whole context of each sentence is taken into account.

The main challenge with this approach was in the unbalanced nature of the data; 90.11% of the sentences used to train the classification model for Phase 1: end-to-end pipeline were non-contributing sentences, and the standard deviation of the frequencies of contributing classes was 8.59%. These frequencies can be seen in Appendix D.

Logistic regression and decision tree classifiers (Pedregosa et al., 2011) were not able to identify the underrepresented classes such as TASKS and DATASET, and were heavily skewed towards the dominant class of non-contributing sentences. During initial experiments using the trial dataset described in Appendix B, decision tree classifier earned a macro-F1 score of 0.1736 and the logistic regression classifier earned a macro-F1 score of 0.1738. The base deBERTa model performed better than both decision tree and logistic regression classifiers on the trial dataset, resulting in a macro-F1 score of 0.3079.

[1]Code is available at https://github.com/anmartin94/DuluthSemEval2021Task11.

3.1 Phase 1: Classifying Sentences using DeBERTa

For Phase 1: end-to-end pipeline testing, we fine-tuned the base deBERTa model on sentences from the training dataset to create a thirteen-class sentence classification model, using HuggingFace transformers' deBERTa for Sequence Classification model (Wolf et al., 2020). Hyperparameter settings can be found in Appendix E.

The provided training dataset includes sentences files that contain a list of the indexes of contributing sentences for each scholarly article. It also contains files for each information unit provided in json format; each of these files include the full sentences belonging to its specified information unit. We labeled the contributing sentences with their information units by looking up each sentence in the information unit json files. Sentences from the articles in the training dataset that were not included in the sentences files were labeled as non contributing.

3.2 Phase 2: Classifying Given Sentences

During evaluation Phase 2 the gold contribution sentences for the test data were given to all participants by the task organizers in sentences files containing the indexes of contributing sentences. Given this, we altered the sentence classification step by fine tuning deBERTa only on contribution sentences from the training data. This resulted in a twelve-class classifier that labeled the test data sentences according to their predicted information unit. The reason why a classification step was still required here is that the triples extraction task further down the pipeline require the sentences to be classified according to their information unit. Without the information unit json files, the information unit labels must be predicted.

Observing that the sentence indexes in the given sentences files appeared to be sorted by information unit, we adjusted the output from the sentence classifier so that chunks of consecutive sentence indexes would all receive the same label. This was performed by searching the classifier output for spans of consecutive sentences where the classifier vacillated between two commonly confused information units, such as EXPERIMENTAL SETUP and HYPERPARAMETERS. For each of these spans, the information unit that was more frequent within the span would be assigned to every sentence.

4 Entity and Relation Extraction

The Duluth system for Phase 1: end-to-end pipeline combined statistical and rule-based approaches for extracting scientific entities and their predicates from the contribution sentences. We used a dependency parser to extract noun phrases and their predicates, and trained an maximum-entropy Markov model (MEMM) on the training data entities files to predict whether each noun phrase contains a scientific entity.

4.1 MEMM Entity Extraction

For Phase 1, in order to tag likely scientific entities in the test data, we trained a MEMM on the provided training data. The features we used include:

- current word type,
- current part-of-speech tag,
- current word shape,
- current IOB tag (I or B if present in the scientific entities list, O if not), and
- the above features for the previous word.

We generated the scientific phrases list from the training data entities files, by extracting the noun phrases from the phrase spans and inputting them into a file to be looked up by the MEMM entity extractor.

For Phase 2 Part 1: phrases and triples, the model was altered to only perform IO tagging. This change was made to address the fact that sometimes individual words appear in different positions in different phrases. For example, the noun "loss" appears in 41 different phrases in the scientific phrases list, sometimes in the beginning of a phrase as in "loss function", and sometimes in the end of a phrase as in "cross entropy loss".

We used these features to train NLTK's MaxentClassifier method (Bird et al., 2009) with maximum iterations set to 40. The model achieved a testing accuracy of 0.995. We used the Viterbi algorithm to derive the most likely IO tags for every word in each sentence.

The model was able to identify some scientific entities in the test data that aren't present in the scientific entities list derived from the training data. However, a complicating factor is that not all entities that must be found can be considered to be exclusively scientific entities. For example, terms like *ReLU*, and *SCIBERT* are clearly specific to

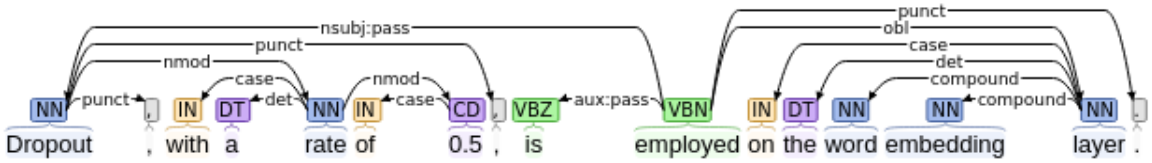

Figure 1: This is a dependency parse of example sentence from corenlp.run/, which shows the information provided to the phrase extraction system by the dependency parser. The dependencies are traced by the Duluth system in order to extract subject-predicate-object phrases from each sentence.

natural language processing and machine learning, but words such as *image*, *action*, *humans*, and other common nouns were also present in the entities training data. Since there is a wide range of specificity in the terms that must be extracted as entities, filtering the sentences through the MEMM entity tagger ultimately worsened the system's performance in Phase 2 Part 1: phrases and triples from a total F1 score of .4634 to .4299. This is because it filtered out sentences which were contribution sentences but whose subject phrases did not contain nouns tagged as scientific entities by the classifier.

4.2 Dependency Parsing

We used Stanford Core NLP's dependency parser (Manning et al., 2014; Chen and Manning, 2014) to generate a dependency parse for each contribution sentence. We used the dependency parse of each sentence to extract the root verb phrase from each sentence, its noun subject phrase, and dependent object phrases. In Phase 1, if neither the subject of the sentence nor the words dependent on it within its noun phrase were tagged as scientific entities, then the sentence would be ignored. Any object noun phrases not containing a scientific entity were also ignored. The intention was to create an outline of each contribution sentence that included only the relevant noun phrases and the relation between them.

Figure 1 illustrates the dependency parse of our running example. The system would extract the verb phrase "is employed" as the relation by finding the ROOT of the sentence ("employed"), finding the adjacent dependency ("is"), and concatenating the two into the phrase "is employed". Next, it would extract the dependent nsubj "Dropout" as the subject entity by searching for nsubj types dependent on "employed". Lastly, the system would extract the noun phrase which is dependent on the verb phrase "is employed" as the object entity. This would be accomplished by finding the noun dependent on "employed", which is "layer", and building the noun phrase "word embedding layer" from the words dependent on "layer". The phrases "Dropout" and "word embedding layer" would be labeled as scientific phrases, which means that this subject-predicate-object phrase would be kept.

In Phase 2, we altered our system so that it would not throw away any sentences, since it was provided with the gold contribution sentences made available by the task organizers. Rather, if the subject was a pronoun and the subject phrase did not contain a scientific entity, then the subject phrase would be removed. If there was a previously selected noun phrase from the same information unit, that phrase would replace the removed subject phrase. Otherwise, the name of the information unit would be used instead. The intention was to handle cases where a pronoun referring to an entity from the previous sentence was the subject of the verb phrase.

5 Triples Extraction

During evaluation Phase 1: end-to-end pipeline, almost all of the task of extracting Subject–Predicate–Object triples into information units files was already performed by previous steps. The entity extractor described in section 4 extracts phrase spans three at a time, following the subject-predicate-object format needed to organize phrase spans into triples. The sentence extractor described in section 3 classifies sentences into their information units, so the class label for the sentence that the triple is extracted from can be used to determine the information unit that the triple belongs to.

For Phase 1: end-to-end pipeline and Phase 2 Part 1: phrases and triples, the system formed triples based on the subject, predicate, and object phrases determined by the entity selection process described in section 4.

For example, if the sentence "Dropout, with a rate of 0.5, is employed on the word embedding layer" is classified by the sentence extractor as

belonging to the information unit EXPERIMENTAL SETUP, and the phrases "Dropout", "is employed", and "word embedding layer" were selected as a subject-predicate-object pattern, then triples would be formatted in the experimental-setup.txt triples file like so:

(Contribution||has||Experimental Setup)
(Experimental Setup||has||Dropout)
(Dropout||is employed ||word embedding layer)

5.1 Triples Formatting

While most of the information units' triples were formatted in the training data following the pattern in section 5, the information units Code and Research Problem were formatted following these patterns, respectively:

(Contribution||Code||url)
(Contribution||has research problem||phrase)

The triples files for information units Code and Research Problem were handled separately from the others in order to reflect these differences.

5.2 Phase 2 Part 2: Adapting to the Released Entities Files

The main problem with the method for building triples described in section 5 is that it does not address the fact that triples often build on each other and overlap. Triples build on each other in two ways: the first item of a triple may be the last item in the previous triple; and the first item of a triple may be the first item in a previous triple.

The Duluth system solution for Phase 2 Part 2: triples extraction attempts to imitate these patterns by following these rules: while items in the entities file alternate between noun phrases and predicates, then triples are formed where the middle item for each triple is a predicate phrase and the phrases on either side of it are the noun phrases in the entities file on either side of the predicate in the entities file. This creates the first pattern identified above. However, if two consecutive noun phrases are found in the entities file, then the second noun phrase is paired with the previous subject-predicate pair, rather than the previous noun phrase. If there is no previous subject-predicate pair, then the second noun phrase is paired with the name of the information unit and the predicate *has*. This creates the second pattern identified above.

Information Unit	F1
None	.9494
Ablation Analysis	.1516
Approach	.0000
Baselines	.2559
Code	.7857
Dataset	.0000
Experimental Setup	.2466
Experiments	.0000
Hyperparameters	.2358
Model	.1778
Research Problem	.4192
Results	.3165
Tasks	N/A

Table 1: F1 scores for each information unit based on evaluation on test data. Macro-F1 score = .2949. Weighted-F1 score = .8866.

6 Sentence Selection Results

The official competition F1 score on the gold standard test data for selection of contribution sentences in Phase 1: end-to-end pipeline was 0.38095. This score evaluates the results as a binary classification problem, where sentences are either contributing or non-contributing. Because our sentence selector performs multi-class classification, labeling each sentence as either non-contributing or belonging to one of twelve information units, we provide a confusion matrix in Table 2 that shows the detailed results by information unit, as well as a break down of individual F1 scores for each information unit in Table 1. The official competition F1 score for information units in Phase 1 was 0.6441.

6.1 Confusion Matrix Analysis

The high frequency of false positives and false negatives for the non-contribution class (None in Table 2) is likely due to its high frequency, as 90.11% of the sentences from the training dataset belong to this class. This shows that using a BERT model without any filtering or sampling techniques is not sufficient to accurately handle the unbalanced nature of this dataset. We will focus our discussion here on the most frequently confused information units.

30.19% (**109** of 361) of sentences describing

Predicted Class	Gold Class												
	N	AA	A	B	C	D	ES	E	H	M	RP	R	T
None	27,922	113	123	55	7	10	178	223	59	446	230	405	0
Ablation Analysis	72	21	0	0	0	0	0	8	0	0	0	**14**	0
Approach	0	0	0	0	0	0	0	0	0	0	0	0	0
Baselines	146	0	0	38	1	0	1	5	2	5	2	2	0
Code	10	0	0	0	33	0	0	0	0	0	0	0	0
Dataset	0	0	0	0	0	0	0	0	0	0	0	0	0
Experimental Setup	125	0	0	0	0	0	73	5	**27**	1	0	0	0
Experiments	0	0	0	0	0	0	0	0	0	0	0	0	0
Hyperparameters	136	0	0	0	0	0	**109**	4	52	0	0	0	0
Model	222	0	**7**	2	0	0	0	2	0	75	2	3	0
Research Problem	88	0	0	0	0	0	0	1	0	3	118	0	0
Results	327	**28**	0	0	0	0	0	**87**	0	1	1	201	0
Tasks	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2: Confusion matrix for thirteen-class sentence classification. The quantities in bold face correspond with the boldfaced quantities in section 6.1.

EXPERIMENTAL SETUP were falsely labeled as belonging to the information unit HYPERPARAMETERS by our classifier, and 19.28% (**27** of 140) of sentences describing HYPERPARAMETERS were falsely labeled as EXPERIMENTAL SETUP. This makes sense, as sentences describing experimental setup may include discussion of hyperparameters.

Sentences belonging to the information unit ABLATION ANALYSIS were incorrectly labeled as RESULTS 17.28% (**28** of 162) of the time (the classification of RESULTS was more successful, only being incorrectly classified as ABLATION ANALYSIS 2.24% (**14** of 625) of the time). This confusion makes sense, as ablation analysis may often be discussed alongside analysis of results. Furthermore, the information unit RESULTS appears almost 3.65 times as often as the information unit ABLATION ANALYSIS, another example of how the uneven distribution of information units impacts the accuracy of our classifier.

Similarly, sentences describing EXPERIMENTS were incorrectly labeled as belonging to RESULTS 25.97% (**87** of 335) of the time, and RESULTS appears in the training data 3.16 times as often as EXPERIMENTS. This resulted in experiments sentences never being positively identified by our system in Phase 1. Lastly, our system was not able to positively identify sentences belonging to the information unit APPROACH, either classifying them as non-contributing, or as belonging to the information unit MODEL.

6.2 Implications for Future Work

Because the frequently confused sentences often have many features in common, in future work we will investigate a document-level classification approach, to explore whether there is information in the paper as a whole that can point towards one information unit over another. For example, the information units APPROACH and MODEL never appear together in the same article in the training data, because they describing the model used is equivalent to describing the approach taken. One might be able to determine whether a paper is likely to discuss a model rather than a general approach by looking at features of the whole document.

One type of feature that might be leveraged for better results is section headers. For example, sentence 5 in Table 3 is found in a section with the header "Our Approach", which indicates the gold label of APPROACH. Often, the contribution sentences describing results are under a section header with the word "Results" in it, as is the case for sentence 3 in Table 3. Similarly, sentence 2 in Table 3 is found in a section labeled "Experimental Setup". However, the correct information unit is not always consistent with the section header. Sentence 1 in Table 3 is labeled with the information unit HYPERPARAMETERS, though it is found in a subsection called "Set-Up" under the section "Experiment Results". Sentence 4 in Table 3 is found in a section called Experimental Results, but has the gold label ABLATION ANALYSIS.

	Predicted	True	Sentence
1	Experimental Setup	Hyperparameters	The initial learning rate is 0.0004 and the batch size is 32 .
2	Hyperparameters	Experimental Setup	Word2vec is used to produce the word embeddings .
3	Ablation Analysis	Results	Furthermore , using pseudo entity annotations boosted the accuracy by 0.3 % .
4	Results	Ablation Analysis	The best performance is achieved by Faceness , with a recall below 20 % .
5	Model	Approach	Crucially , the BiLSTM is trained with the rest of the parser in order to learn a good feature representation for the parsing problem .

Table 3: Examples of mislabeled sentences

Error	Common Words (frequency)	Top POS (probabilty)
B-miss	our (124), all (17), to (14), each (14)	PRP$ (.2255)
E-miss	model (20), to (17), using (13), method (8)	NN (.2411)
B-extra	best (3), use (2), set (2), the (2)	NN (.4235)
E-extra	. (24), on (8) in (7), than (7), by (6)	IN (.2784)

Table 4: For each error type, this table shows the most frequent words either missed by predicted phrase spans, or added (extra) by predicted phrase spans. Column 3 contains the most frequent POS tag for each error type (e.g. 22.55% of all words missed from the beginning of a span were possessive pronouns).

7 Entity and Relation Extraction Results

The gold test data had 13,028 total phrase spans, of which our system identified 4,277. 39.72% of these predicted phrase spans were exact matches with gold phrase spans. 29.97% of predicted phrase spans were complete false positives, not overlapping with any gold phrase spans. 30.30% of predicted phrase spans were not perfect matches, but did overlap with gold phrase spans. The majority of partial matches were missing a word, reflecting the fact that the average span length for the gold data was 12.37 characters but the average span length for the predicted data was 10.49 characters. 42.44% of partial matches were missing characters in the beginning of the phrase span, 52.11% were missing characters in the end of the phrase span, 6.56% had extra characters in the beginning of the phrase span, and 14.97% had extra characters in the end of the phrase span.

7.1 Partial Phrase Matches

In this section we first look at the characteristics of the partial matches to discover the cause of these errors. Because the Duluth system selected phrase spans using grammatical features, we look at the parts of speech of the words that were either missed by the Duluth system or erroneously added to phrase spans. Then, we look at the grammatical characteristics of the gold phrase spans that were entirely missed by the Duluth system.

The most common errors and their frequencies are shown in Table 4. The part-of-speech that was most likely to be missing from the end of a span (E-miss) in the Duluth system output was NN. Our system misses these noun phrases because the Duluth system outlines sentences starting with the verb labeled as ROOT by Stanford Core NLP's dependency parser, and identifying the nsubj of that root and its dependencies as an entity [2].

However, there are some cases where the word identified as the ROOT is not the verb that the subject of the sentence is directly dependent on. In these cases, the sentence subject goes undetected by our system. For sentence 1 in Table 5, if the verb *improves* were properly identified as the ROOT, the sentence might be outlined like so: Built on top of the model in but excluding ELMo, *base reinforced model* (ENTITY) *improves* (RELATION) the *average F 1 score* (ENTITY). However, *Built* is identified by the dependency parser as the ROOT, so the Duluth system fails to extract the noun phrase *our base reinforced model*, which is the nsubj phrase dependent on *improves*. Notice also in this example the omission of the possessive pronoun *our* in the system outline; this illustrates the most common part-of-speech missing from the beginning of

[2]ROOT refers to the root of the dependency parse tree. The nsubj of a parse is the nominal subject dependent on the root.

1	Built on top of the model in but excluding ELMo, our base reinforced model improves the average F1 score around 2 points [...]	
1	System: base, built on, top	Gold: our base reinforced model, improves, the average F2 score, around, 2 points
2	Finally, the baseline model, EDA, is largely outperformed by all other examined methods.	
2	System: baseline model, outperformed by, other examined methods	Gold: EDA, largely outperformed, by, all other examined methods
3	We also compare with Tagspace (Weston et al. , 2014) , which is a tag prediction model similar to ours [...]	
3	System: We, compare with, Tagspace	Gold: compare with, Tagspace (Weston et al., 2014), tag prediction model

Table 5: Example sentences with phrase spans improperly extracted by the Duluth system. The phrase spans are shown separated by commas.

a phrase span, as 24.11% of words missing from the beginning of phrase spans have POS tag PRP$.

27.84% of words that were erroneously appended to the end of phrase spans by the Duluth system were prepositions (IN). This is because the Duluth system includes prepositions in verb phrases, while the gold data contains some phrase spans where the preposition is separated from the verb, and others where the preposition is included in the same phrase span as the verb. Sentence 2 in Table 5 shows the Duluth system incorrectly including the preposition *by* with the verb *outperformed*, while sentence 3 in 5 shows our system correctly including the preposition *with* with the verb *compare*.

Phrase (Frequency)	of (502), with (295), on (274), for (260), in (190), to (129), from (87), using (85), by (80), as (66), than (60), at (52), is (46), between (44), results (39), over (36), achieves (36), based on (32), outperforms (31), training (26)

Table 6: Most frequently missed phrases.

Phrase spans that improperly capture additional words can have a ripple effect when the additional word is supposed to belong to a different phrase span, like the additional word *by* in the phrase span *outperformed by*. Because *by* is included in a phrase span already, it does not exist in its own phrase span as it does in the gold data, which means that the phrase span *by* is completely omitted by the Duluth system.

This error is quite common; 97% of the phrase spans in the gold data consisting of a single preposition were not identified by the Duluth system, due to the fact that these words tend to get absorbed by other phrases. This continues to have a detrimental effect further down the pipeline, as whether the prepositions are alone or chunked with other phrases affects the formation of triples.

7.2 Missed Phrases

To determine the possible causes of missing phrase spans, we looked at the error rates by part-of-speech. Since 77.31% of the total phrase spans in the gold standard data were missed by our system, we are only considering POS patterns that were missed over 77.31% of the time. We are also only looking at POS patterns that occurred at least 100 times in the gold data.

Phrase spans made up of a single preposition were frequently not identified by the Duluth system. This is apparent in Table 6; the top 5 phrases most commonly ignored by the Duluth system are all prepositions. Table 7 shows the phrase types (phrases described by the POS tag for each word) that are least likely to be captured in their entirety by the Duluth System. In addition to prepositions, infinitive verbs (TO VB) are also frequently omitted by the Duluth system. This is because the Duluth system bases the outline of the sentence off of the ROOT, which is almost always a finite verb. For this reason, infinitives are frequently ignored.

Our grammar-based approach might be improved on by changing the rule that outlines the sentences based on the ROOT verb of the sentence, in order to focus more on noun phrases that are more likely to be scientific entities. This would

POS Pattern	Missed/Total	Example
IN	.9735	for
TO	.9416	to
CD	.9364	99.60
TO VB	.9098	to compute
RB	.9040	jointly
VBG	.8837	using
NN NNS	.8750	feature maps

Table 7: These are the POS patterns that are most likely to be ignored by the Duluth system, and the conditional probability that they will be omitted.

require developing a method for tagging likely scientific entities that improves on our MEMM entity tagger. However, entities extraction might be better addressed with a neural approach since our grammatical approach (without semantics) does not capture the nuances in the training data.

8 Triples Extraction Results

The official F1 score on the gold standard test data for triples extraction from Phase 2 Part 2 was 0.2762, and the F1 score for information units was 0.7556. The Duluth system scored the lowest at extracting triples for the information units DATASET, TASKS, EXPERIMENTS, receiving F1 scores of 0.0, 0.0, and 0.0597 respectively. The extracted triples are input into files named for the information unit they belong to.

Triples extracted from a sentence classified as RESEARCH PROBLEM are output into a file named research-problem.txt. The gold data does not contain any tasks.txt files, so all tasks.txt files generated by our system were false positives. The gold data contained two dataset.txt files, both of which were missed by our system. These scores are consistent with the initial results from sentence classification, as seen in Table 1. Because our system deals with information unit classification during the sentences extraction step rather than the triples extraction step, we focus our discussion here on errors related to our triples extraction methodology.

Our triples extraction system relies heavily on part-of-speech tagging to determine whether an entity belongs to the edge of a triple or the middle (for example, noun-phrases are more likely to be subjects or objects, and verb-phrases are more likely to be predicates). In order to determine the efficacy of this approach, we look at the words that were improperly positioned in our system triples.

One pattern that emerged is that many words were wrongly positioned in the Duluth system data. Some phrases and their POS tags that only exist in the middle position in the gold data that are found in the edges of triples in the system data include “achieves” (VBZ), “propose” (VB), “performs” (VBZ), and “uses” (VBZ). Other phrases that only exist in the edges of triples in the gold data that are found in the middle position in the system data include “outperformed” (VBG), “worse” (JJR), “outperforming” (VBG), and “randomly initialized” (RB VBN).

This observation is consistent with the POS distributions found for the gold triples not identified by the system and the system triples that were wrongly identified; 70 of the 792 phrases belonging to the edge of the gold triples missed by the Duluth system have the POS pattern RB VBN, while 46 of the 730 phrases that were wrongly positioned in the middle of triples by the Duluth system also have the pattern RB VBN. Similarly, 324 of the 1,656 phrases belonging to the middle of gold triples missed by the Duluth system have the POS tag VBZ, while 712 of the 4,242 phrases that were wrongly positioned at the edge of triples by the Duluth system have the same POS tag of VBZ. This shows that the Duluth system sometimes shifts the phrases to the left or right of where they ought to be in the triple pattern.

9 Future Work

One weakness of this system is that the selection of contributing sentences only uses sentence-level information; the classifier misses useful contextual information such as headers and the predicted class of preceding and following sentences. Future work may be able to address this problem by fine-tuning BERT to classify sequences of sentences rather than isolated sentences (Cohan et al., 2019). Generally, a document-level approach could be beneficial in terms of capturing important context.

Another issue is that the end of the system does not have the ability to provide feedback to earlier parts of the pipeline; the only agency it has in terms of contributing sentence selection is the ability to discard a sentence provided by the sentence classifier. Future work could incorporate a neural entity classification model into the entity and triple extraction subsystem, which could be used to validate or invalidate the classification made at the sentence level (Rotsztejn et al., 2018).

References

Waleed Ammar, Matthew Peters, Chandra Bhagavatula, and Russell Power. 2017. The AI2 system at SemEval-2017 task 10 (ScienceIE): semi-supervised end-to-end entity and relation extraction. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 592–596, Vancouver, Canada. Association for Computational Linguistics.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Arman Cohan, Iz Beltagy, Daniel King, Bhavana Dalvi, and Dan Weld. 2019. Pretrained language models for sequential sentence classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3693–3699, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jennifer D'Souza, Sören Auer, and Ted Pedersen. 2021. SemEval-2021 task 11: NLPContributionGraph - structuring scholarly NLP contributions for a research knowledge graph. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). Association for Computational Linguistics.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, and Thierry Charnois. 2018. SemEval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 679–688, New Orleans, Louisiana. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Wei Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *2021 International Conference on Learning Representations*.

Simon David Hernandez, Davide Buscaldi, and Thierry Charnois. 2017. LIPN at SemEval-2017 task 10: Filtering candidate keyphrases from scientific publications with part-of-speech tag sequences to train a sequence labeling model. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 995–999, Vancouver, Canada. Association for Computational Linguistics.

Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D'Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. 2019. Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture*, K-CAP '19, page 243–246, New York, NY, USA. Association for Computing Machinery.

Xinyi Liu and Artit Wangperawong. 2019. Transfer learning robustness in multi-class categorization by fine-tuning pre-trained contextualized language models. *arXiv preprint arXiv:1909.03564*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Prismatic Inc, Steven J. Bethard, and David Mcclosky. 2014. The stanford corenlp natural language processing toolkit. In *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jonathan Rotsztejn, Nora Hollenstein, and Ce Zhang. 2018. ETH-DS3Lab at SemEval-2018 task 7: Effectively combining recurrent and convolutional neural networks for relation classification and extraction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 689–696, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on*

Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

A Ethical Considerations

The use of knowledge graphs to process, organize, and display information comes with some ethical implications, due to the fact that such a system acts as an intermediary between human readers and academic research. Overall such a system, if integrated into academia, may benefit authors of scholarly work by making their articles easier to find. It may also benefit readers, by making the process of literature exploration more efficient. However, unintentional harm may arise if there is bias in the training used to annotate the articles. Since the Open Research Knowledge Graph is structured so that contributions are interconnected in the graph across papers, there may be a risk that different kinds of contributions are more easily found than others.

Another concern is whether such an infrastructure could affect how people design and report their research, and if that effect is harmful. If researchers know that their work will be read by a machine and then integrated into the knowledge graph according to its most likely contributions, they could be consciously or unconsciously motivated to favor some methods and terms over others, in an attempt to optimize the likelihood that their work is seen. There is also the potential for bias to become embedded in the machine reader, that could influence what kinds of researchers have work that is easily discoverable in the knowledge graph.

B Task Data

Prior to Phase 1, most of the data used to train the models was extracted from the original training dataset, and most of the provided trial dataset was used for evaluation. Six of the eight articles in the trial data that contained the information unit TASKS were moved to the training dataset so that it would contain enough examples to learn the patterns associated with this information unit. These folders were folder 7 from machine-translation, folders 3, 4, and 8 from named-entity-recognition, folder 8 from question-answering, and folder 2 from text-classification. For all evaluation phases, the models were retrained on the combined training and trial datasets.

C System Architecture

The system architecture is organized into three sections: preprocessing, training, and testing. The preprocessing section is responsible for :

1. extracting noun and noun phrases from the training data entities.txt files,
2. extracting each sentence from the training data and labeling them with their information unit (or 0 for non-contributing sentences), and
3. extracting sentences from the evaluation phase data and labeling each one with its file path and sentence index.

The training pipeline is responsible for fine-tuning the deBERTa model using the sentences extracted from the training data, and training the MEMM using the extracted sentences and the list of nouns extracted from the training entities files.

The testing pipeline is responsible for taking the sentences extracted from the evaluation phase data and labeling each sentence with their predicted information unit, entity tags, and dependency parse. The data is passed between each step of the pipeline in a dictionary, which is added to at each step.

D Information Unit Class Distributions

Phase 1 Class Distributions	
None	.9011
Ablation Analysis	.0062
Approach	.0030
Baselines	.0083
Code	.0010
Dataset	.0010
Experimental Setup	.0091
Experiments	.0072
Hyperparameters	.0102
Model	.0166
Research Problem	.0123
Results	.0228
Tasks	.0012

Table 8: Distribution of classes in the 59,755 training sentences provided by the training data.

E DeBERTa Hyperparameters

optimizer	AdamW
learning rate	5e-5
max sequence length	128
epochs	8
batch size	32

Table 9: Hyperparameters used to fine-tune deBERTa