Relation Extraction with Automatic Ontonotes Annotation and ETC

by
Andrew Sauer

A THESIS

submitted to

Oregon State University

Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Computer Science and Mathematics
(Honors Scholar)

Presented June 9, 2023
Commencement June 2023

AN ABSTRACT OF THE THESIS OF

Andrew Sauer for the degree of Honors Baccalaureate of Science in Computer Science and Mathematics presented on June 9, 2023. Title: Relation Extraction with Automatic Ontonotes Annotation and ETC.

Abstract approved:_____

Prasad Tadepalli

Intuitively, it seems as though natural language processing tasks might benefit from explicit representations of the syntactic and semantic properties of text. Ontonotes is a dataset which attempts to annotate texts, to represent as much as possible of the meaning of the text explicitly within the annotation. Many tools exist for automatic annotation of various semantic features of raw text. This project uses these tools to annotate text from DocRED(Document Level Relation Extraction Dataset) with Ontonotes-style structured data, which is then used to train ETC, an extended transformer architecture for working with structured data, on the task of document-level relation extraction. We evaluate the hypothesis that ETC will perform better on this semantically-annotated dataset than the base DocRED dataset.

Keywords: NLP, Relation Extraction, Ontonotes, ETC, DocRED

Corresponding e-mail address: mrperson59@gmail.com

©Copyright by Andrew Sauer
June 9, 2023

Relation Extraction with Automatic Ontonotes Annotation and ETC

by
Andrew Sauer

A THESIS

submitted to

Oregon State University

Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Computer Science and Mathematics
(Honors Scholar)

Presented June 9, 2023
Commencement June 2023

Honors Baccalaureate of Science in Mathematics and Computer Science project of Andrew Sauer presented on June 9, 2023.

APPROVED:

_____

Prasad Tadepalli, Mentor, representing Department

_____

Stefan Lee, Committee Member, representing Department

_____

Xiaoli Fern, Committee Member, representing Department

_____

Toni Doolen, Dean, Oregon State University Honors College

I understand that my project will become part of the permanent collection of Oregon State University, Honors College. My signature below authorizes release of my project to any reader upon request.

_____

Andrew Sauer, Author

Introduction:
In Natural Language Processing, there have been two general approaches to building successful language models. First, there is the classical NLP pipeline approach, which aims to facilitate machine understanding of text by breaking the problem into a series of tasks which are created by humans based on our domain knowledge and understanding of the rules of language. Each of these tasks, in theory, makes some aspect of the meaning of the text explicit, and therefore easier for models to learn about. This approach is exemplified by the Stanford CoreNLP toolkit, which provides tools for several of these tasks [3], and by Ontonotes, which is a dataset which consists of many documents hand-annotated with these task features, for use in training these pipelines [6]. Since tools for each step of the pipeline already exist, we will simply use Ontonotes as a guide for the tasks to include in the pipeline.

In contrast to the classical NLP pipeline, there is the more recent, and often more effective, end-to-end training approach for large language models. These models are trained on simple tasks such as mask prediction, and then fine-tuned for use on more specific tasks, including those in the classical NLP pipeline as well as others. This approach has pushed forward the state of the art in NLP on many tasks with the advent of Transformer models [11] such as BERT [12] and RoBERTa [13]. More recently we have seen ChatGPT and GPT-4 [14], which have been extremely successful, achieving state-of-the-art performance in diverse areas of NLP [9][10].

The tasks in the classical NLP pipeline include part of speech tagging, parsing, entity typing, coreference resolution, semantic role labeling, and word sense disambiguation. Part-of-speech tagging is simply the process of classifying the part of speech of each token in the sentence. There are two types of parsing, constituency and dependency, but we will focus on constituency parsing because this is what is used in ontonotes-style annotation.

Constituency parsing annotates the structure of a sentence by creating a tree structure of token spans, each span representing a grammatical construct, for example a noun phrase, an adjective phrase, or a predicate. Each span of more than one token is split into smaller spans for its child nodes, which represent the smaller grammatical constructs that constitute it, thus the term constituency parsing. Penn Treebank is the most influential dataset for this task, and Ontonotes uses the Penn Treebank format for its parsing features [6]. As an example of the format from Wikipedia, “John loves Mary” becomes:

```
(S (NP (NNP John))
  (VP (VPZ loves)
     (NP (NNP Mary)))
  (. .))
```

Entity typing is the process of identifying proper nouns, or entities, within a text, and assigning a broad type to them(such as Person, Time, Place, etc). In the DocRED dataset [1], this information is already included in the features, and so no automatic annotation is needed.

Coreference resolution is the process of identifying grammatical constructs, mainly noun phrases and pronouns, within a document which refer to the same thing, thus making explicit connections between semantically related phrases. For example, in the sentence “The caterpillar ate through an apple, but he was still hungry,” the phrases “The caterpillar” and “he” refer to the same thing.

Semantic role labeling is the process of determining the subject and object of a given verb within a sentence, alongside other information related to the verb, for example the time or place the event occurred at. Propbank is the most influential dataset for this task, and Ontonotes uses the semantic frames provided by Propbank [6]. For example, in the sentence “In 2015, Ted became a doctor,” the verb “became” fits the Propbank frame “become.01”. ARG1, the entity changing, would be Ted, and ARG2, the new state, would be “a doctor”, while there would also be an auxiliary argument, ARGM-TMP, for the time the frame takes place at, in this case “2015”.

Word sense disambiguation is the process of determining which of several possible meanings a given token has in context. For example, the term “bank” means completely different things in the phrase “rob a bank” and the phrase “swim to the bank”, even though in both cases it is the same part of speech. The most influential set of word senses used for this task is WordNet. Ontonotes uses its own method of clustering WordNet senses into more coarse-grained senses [6], but for this project we will just predict WordNet senses, as this is what is used by most WSD models, including CoNSeC [3] which we use.

These tasks were created and specified by humans, using linguistic domain knowledge, with the purpose of training models to make explicit the implicit semantic and syntactic connections within natural language texts that inform their meaning. However, this pipeline-based approach has fallen out of favor recently. The previously mentioned newer approach of fine-tuning on large pre-trained Transformer models has tended to provide better results, including on pipeline tasks such as word sense disambiguation and semantic role labeling. The WSD annotator we use, CoNSeC, is built on top of the transformer model DEBERTA [3], and the SRL annotator we use, MRC-SRL, is built on top of RoBERTa [4].

This raises the question of whether the classical pipeline-based approach is still useful. Can annotations generated by these pipeline tools be used to improve the performance of Transformer models on some natural language task? Put another way, can Transformer models benefit from a pipeline architecture, where they are trained explicitly to extract relevant textual information of the types described above, with architecture designed specifically for good performance on each one separately, as an intermediate step to a final natural language task? Our project seeks to investigate this question.

Relation extraction is the task of inferring semantic relationships between entities within a text. For example, in the sentence “Fred’s son David lives in Australia”, a relation extractor would find the relations (Fred, father, David), (David, son, Fred), (David, country, Australia).
Document-level relation extraction, the task that we’ll be focusing on, requires inference of such relations between entities which may not be mentioned in the same sentence (for example, from

“Bob’s mother is Jane. She lives in England.” infer (Jane, country, England)). This is the task the DocRED dataset was created for [1].

We select this task for testing because it is not one of the tasks in the pipeline, but the information being inferred is similar, in particular to the task of semantic role labeling. For example, it does not seem too far a leap from the semantic frame (V: is, ARG0: Bob’s mother, ARG1: Jane) to the relation (Jane, mother, Bob). Similarly, if we have the coreference information that “She” refers to “Jane”, and the semantic frame (V: lives, ARG0: She, ARG1: England), it doesn’t seem to be too far a leap to the relation (Jane,country,England).

For our model architecture, we select ETC, or Extended Transformer Construction. ETC is ideal for our project because in addition to having achieved state-of-the-art results on several language processing tasks, it is specifically designed to work with non-sequential structured input [2]. With ETC, arbitrary relations on the input features can be encoded as attention modifiers, and there are two separate types of input, long and global. The long input contains the sequential tokens in the input text, and the global input contains tokens such as sentence markers, non-sequential input, or for our purposes, tokens representing Ontonotes-style annotations.

This allows us to create an Ontonotes-style annotator from already existing tools for each annotation task, and convert the output of this annotator into a set of global tokens, together with the relationships of these tokens to the sequential tokens in the input. The exact method of doing this is explained in the “System Architecture” section.

So, our hypothesis is that ETC models will perform better on DocRED when their input features are augmented with these Ontonotes-style annotations, than they will when training and evaluating on the base DocRED dataset. This hypothesis seems reasonable because, as illustrated above, the annotation tasks in Ontonotes intuitively seem to be relevant to the task of document-level relation extraction.

However, it is also possible that the annotation will not help. Perhaps our intuition is wrong, and the structural information provided by Ontonotes annotation is not actually relevant to this task, perhaps because most relations in DocRED cannot be reliably predicted by reasoning like in the “Bob and Jane” example above, or because such reasoning is not what is being learned by the model. Or, perhaps the annotations are redundant, and the structural relationships they represent are already being learned by the model. Finally, it is possible that the annotations are too often incorrect to provide information useful enough to reliably improve prediction.

System Architecture:
At a high level, our system consists of two components: an automatic OntoNotes-style annotator and ETC. We first input the text of DocRED into the annotator, resulting in a graph representing the OntoNotes annotation of the text, which is added as structured data to the DocRED dataset. Then, we train and evaluate an ETC model on the DocRED data both with and without the OntoNotes annotation.

OntoNotes Annotator:

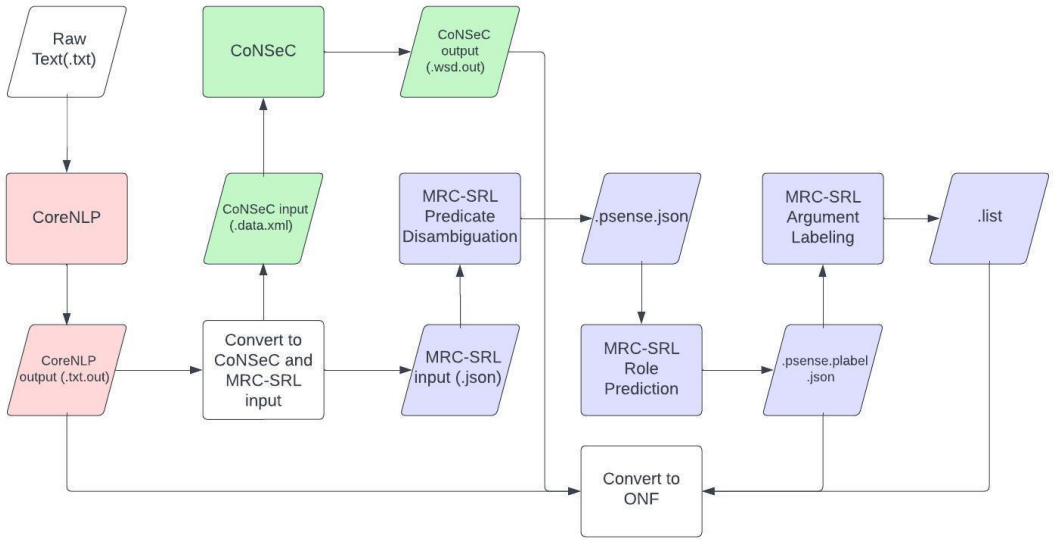

Figure 1: Annotator Flowchart
In this figure, the parallelograms represent intermediate files, the rectangles represent annotators, red represents the CoreNLP portion, green represents the CoNSeC portion, and blue represents the MRC-SRL portion.

The OntoNotes annotator consists of 3 major components: a CoreNLP pipeline [5], a CoNSeC model [3], and an MRC-SRL model [4]. The system architecture is shown in Figure 1. CoreNLP is the Stanford NLP tool which is used to do tokenization, part of speech tagging, constituency parsing, and coreference resolution [5]. Outputs from this are fed into the CoNSeC system which predicts a WordNet word sense for all nouns and verbs in the text, and into MRC-SRL which matches verbs to semantic frames, and predicts the spans for each entry to the frame. MRC-SRL has three components: predicate disambiguation, which determines which Propbank semantic frame matches a particular predicate, Role Prediction, which predicts which semantic roles (including auxiliary arguments such as location, time, purpose, etc) can be filled for this specific frame, and finally Argument Labeling, which identifies the spans of text for each argument [4].

The output of this system is a .onf (Ontonotes Normal Form) file for each input .txt file. Each of these files contains all the Ontonotes annotation info for its corresponding raw text file. These in turn are used to add to the ETC input as described in the next section.

ETC:

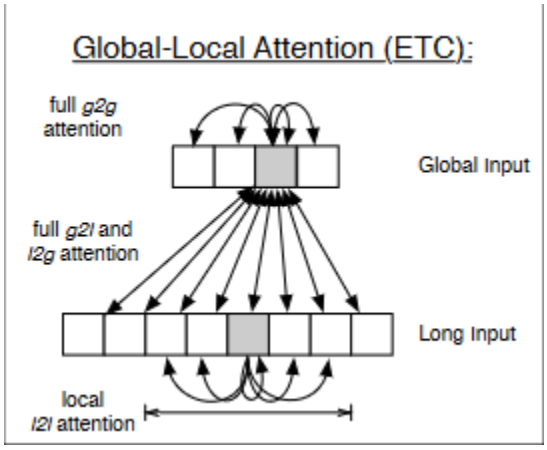

Figure 2: ETC Global-Local Attention
This image from the ETC paper shows the two input sequences and how they relate to the restricted l2l attention and unrestricted g2g, g2l, and l2g attention tensors shown in Figure 3.

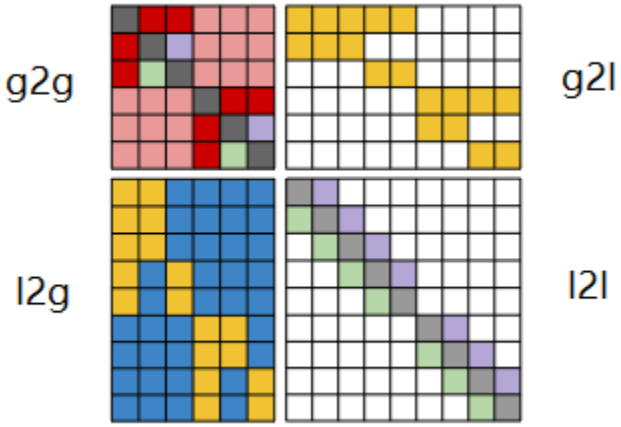

Figure 3: Attention tensors
Another image from the ETC paper, this shows an example of the attention tensors for a structured input. White entries are masked, and color entries represent different attention labels. We can see in the g2l section that hard g2l masking has been applied in this example.

Using the output .onf files from our OntoNotes annotator, we create a graph structure on top of the raw text tokens. We then add attention labels for the ETC attention layer which represent the edges in this graph, and add tokens to the global input representing the non-sequential graph nodes. The ETC architecture as described in (Ainslie et al., EMNLP 2020) seems suited for this kind of task, training on data with additional non-sequential structure.

The DocRED dataset includes raw text along with annotations of entities within the text. The labels we predict are WikiData relations attached to ordered pairs of these entities. In DocRED there are 96 different relations. An ordered pair of entities may have any number of relation labels, they could have no relation, one relation, or multiple. Therefore we treat this problem as a binary classification problem on ordered triples containing a subject entity, an object entity, and a relation type.

To adapt ETC to this task, we first encode the inputs as tensors as described below, and run it through a 12-layer ETC model, which outputs a 768-length embedding for each token in the input. Then, we take the embeddings of the global tokens corresponding to entities, and for each pair of entities, we concatenate their embeddings, and run the result through several final ReLU-activation Dense layers, resulting in a 96-length tensor for each pair. We interpret these output values as logits for the potential relation labels for this entity pair.

Now, we explain the input to ETC. The ETC model is structured using global-local attention. What this means is that, for each training and validation example, there exist two linear input tensors: long(or local) which represents the tokenized text input, and global, which contains tokens corresponding to nodes from the structured graph. These two inputs are shown in Figure

2 above. Then there are four 2d input tensors representing relative attention ids between input tokens, one for each combination long/global to long/global, denoted by l2l,l2g,g2l,g2g. An example of how these might look for a structured input is shown in Figure 3. Each entry in these tensors represents a possible edge in the structure graph, and the numeric value denotes which type of edge it is, or some default value for “no edge”. This is a generalization of relative position labeling that ETC is designed for, the only difference is that the labels are determined based on the edge type between the two nodes rather than their relative position in a sequence. Finally, there are four 2d input tensors representing attention masks on these four combinations. These contain 1s and 0s denoting which input tokens should attend to which other input tokens at all.

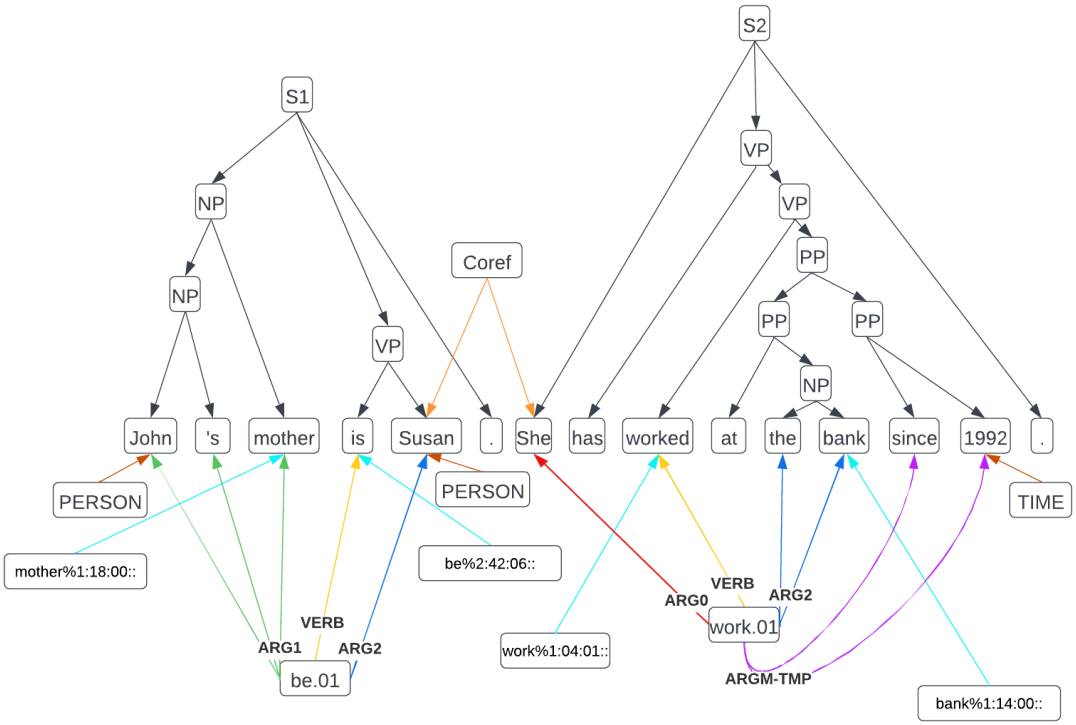

Figure 4: Annotation Graph Example
An example of the structure of an annotation graph created by the annotator. The sentence tokens in the center line are represented as local input, while the rest are represented as global input. The colors of the arrows represent the different attention labels between two tokens. In this example, black is constituency parsing, orange is coreference, light blue is word sense disambiguation, and the other colors represent different argument types for Propbank frames.

To adapt this to the task of DocRED Relation Extraction, we construct these input tensors as follows: The long input is a sequence of tokens. It is tokenized first with CoreNLP, for compatibility with the Ontonotes-style annotation, then the BERT WordPiece tokenizer converts this tokenization to integers and splits up any of the tokens not in its vocabulary, for compatibility with the pretrained ETC model. The long input is then padded with zeros. The global input is split up into sections, one for each type of annotation, as explained in Appendix 1.

The relative attention ids are filled in with information from the DocRED data, and from OntoNotes. An example of the type of graph which might be constructed from our annotations is shown in Figure 4. For g2g, sentences are related to each other with relative position labeling, while other nodes are related to the sentences they appear in. l2l is populated with relative position labels using the feature_utils module provided by ETC. g2l and l2g are nearly symmetrical, with global tokens being related to the long tokens that make them up, in the same way for sentences, entities, and the added annotations. However, we also use hard g2l masking, meaning that g2l attention is masked between global and long nodes which have no relation, while l2g attention simply has another relative position label for that case. For a more detailed explanation, see Appendix 1. For a visual example, see Figure 5.

Finally, the masks are created. These masks simply forbid attention between actual tokens and padding tokens, except for the g2l mask which forbids attention between tokens with no annotated relation to each other, since we are doing hard g2l masking.

Figure 5: Attention Matrix Example
This shows how the annotation example from Figure 4 would be encoded in the attention matrix. Here, white means attention is masked, dark grey means no particular relation, light grey means same token, tan means the sentence token contains the local token, in g2g blue and red indicate which sentence/constituency tokens contain which others, in g2g the colors represent relative positions, and the rest of the colors are as in Figure 4. Note that in a real example, each global token section would be padded, with padding tokens masked from actual tokens, to keep the size consistent.

Research Methodology:
The specific hypothesis being tested is that ETC performance on DocRED is significantly better when trained on auto-annotated data. For control, we will train and evaluate an ETC model on the DocRED dataset normally, so that the only input to the model is the raw text of the input documents along with information about sentences and the positions and types of entities in the text. This minimal annotation, which comes directly from the DocRED data, is required to measure the accuracy of the output, as the labeling of entity pairs requires accurately identifying all entity mentions in the text.

For the experiment, we will train and evaluate an ETC model on a version of the DocRED dataset, with tokens and structured information added to represent the information contained in an Ontonotes-style annotation. These annotations will be added with the auto-annotator. Unfortunately due to memory concerns, not all constituency parse nodes can be represented in the input data. The non-sequential nodes must be placed in the global input for ETC, and the memory cost increases quadratically with the global input size. Therefore only non-leaf nodes are included. Furthermore, the second largest annotation type, word sense disambiguation, could not be included alongside constituency parsing without reducing the size of the parsing annotation even more by only including noun-phrase nodes. Both the annotation with all non-leaf parse nodes but no word sense disambiguation, and the annotation with only noun-phrase parse nodes and word sense disambiguation, will be used and reported in the results.

The task of relation extraction will be interpreted as a binary classification problem on combinations of subject entity, object entity, and relation type within the input document, as explained in the System Architecture section above. Evaluation will be done on the DocRED devset. Performance will be measured using the F1 metric on this classification problem. Additionally we will report Ign F1 as defined in the DocRED paper. Ign F1 ignores those relations in the devset which also exist in the training set, for the purposes of calculating the F1 score.

We will attempt to train the model with two different loss functions. First, the most obvious, weighted thresholded binary cross-entropy. This loss function is simple, configurable with the hyperparameter of negative example weight, and suited to extremely imbalanced classification problems like this one. Second, we will try Adaptive Focal Loss, as defined in the KD-DocRE paper [7]. This loss function uses a 97th logit to define the threshold against which the others will be measured. It was designed for use on DocRED, and used in the KD-DocRE system which is near the current state of the art for DocRED [7]. It contains a hyperparameter, gamma, to be tuned.

Furthermore, there are some hyperparameters that we can tune in addition to those in the loss function. We can choose how many final pair-combination layers to use. We can choose the learning rate, the batch size, the number of epochs, and the prediction threshold (for Binary Cross-Entropy). Additionally, the ETC github provides models pre-trained with Contrastive Predictive Coding. These pre-trained models require the use of the same tokenizer and number

of layers that they were trained on. Because of resource constraints, we will use the smaller, 12-layer ETC pretrained model, etc_base_2x_pretrain, and so we must use the same tokenizer it uses: the BERT WordPiece tokenizer.

Results and discussion:

Hyperparameter Testing:

Table 1 shows the results for the hyperparameter testing runs we were able to complete with available resources. All results are on the validation data, as the DocRED test data has been released to the public without labels. All results are the highest measured in any epoch of the run, and for binary cross-entropy, the best prediction threshold. Performance did fluctuate epoch-to-epoch. For loss functions, x BCE means Binary cross-entropy with negative example weighting x, and x AFL means Adaptive Focal Loss with gamma x. In each box, both F1 and IGN F1(F1 score ignoring relations which are repeated from the training data) are reported in the format F1/IGN F1. For example, 0.58/0.55 means 58% F1, 55% IGN F1. Unless stated otherwise, runs are done with the Adam optimizer with learning rate 5*10^-5, over 15 epochs, with examples batched in groups of 5, 5 final layers, and pretrained with etc_base_2x_pretrain(meaning 12 layers, 12 attention heads, local radius 84, hidden size 768, hard g2l masking, and all the other settings inherited from that pretrained model). BASE means the task is done without additional Ontonotes annotation, ANNO means it is done with annotation with only noun phrases in constituency parsing, and CONSTITFULL means all non-leaf constituency nodes are included, but WSD is excluded. The reason for the decision to annotate like this is explained in the Research Methodology section.

Table 1: Hyperparameter testing results

Run Parameters	BASE	ANNO	CONSTITFULL
0.25 BCE	0.566/0.532	0.573/0.543	0.564/0.535
1.0 BCE	0.567/0.532	0.569/0.537	0.561/0.530
0.5 AFL	0.575/0.541	0.576/0.544	0.587/0.558
1.0 AFL	0.575/0.543	0.585/0.556	0.569/0.537
1.5 AFL	0.585/0.553	0.560/0.527	0.572/0.539
1.0 AFL, 2 Final Layers	0.571/0.537	0.571/0.540	0.580/0.549
1.0 AFL, 10 Final Layers	0.565/0.535	0.557/0.530	0.577/0.546
1.0 AFL, AdamW	0.589/0.560	0.593/0.563	0.580/0.551
1.0 AFL, 2e-5 Learning Rate	0.578/0.548	0.575/0.544	0.577/0.545
1.0 AFL, 1e-4 Learning Rate	0.536/0.508		
1.0 AFL, 30 Epochs	0.589/0.557	0.590/0.561	0.580/0.550
1.0 AFL, Batch size 1	0.583/0.551	0.577/0.546	0.579/0.546
1.0 AFL, Batch size 10	0.573/0.539		
1.0 AFL, Batch size 1, 2e-5 Learning Rate, AdamW	0.593/0.567	0.594/0.563	0.593/0.562

Table 1 documents the runs that resources allowed, searching for good hyperparameters for the final statistical test. We found that performance seemed better when using Adaptive Focal Loss, we chose gamma=1.0 arbitrarily since it seemed to work just as well as other values, and performance seemed better with a batch size of 1, a learning rate of 2e-5, and using the AdamW optimizer, which was used for training ETC in the original paper, instead of Adam. Combining these hyperparameters together into a run yielded the best results seen so far, so we decided to use these as our hyperparameters for the significance test. A full coordinate search would have been better, of course, but it would have been prohibitively time-consuming. Increasing the number of epochs to 30 improved scores by about 1%, but this slight improvement was not enough to justify doing this for the significance test, as this would double the time cost of the significance test.

Significance testing:

For each setting, BASE, ANNO, and CONSTITFULL, we ran 30 training runs with the chosen hyperparameters: 1.0 gamma Adaptive Focal Loss, 15 epochs, batch size 1, AdamW optimizer, 5 final layers, 2e-5 learning rate. These runs differed from each other due to the random

element introduced by the Glorot Uniform initialization of the final layers and the shuffling of the order of training examples(the ETC layer initialization was not random because we loaded the pretrained ETC model). We perform an unpaired two-tailed t-test to compare BASE to ANNO, and another to compare BASE to CONSTITFULL, to determine if there is a $p<0.05$ significant effect on the result when adding annotations to the DocRED data. We repeat this for both the F1 and IGN F1 measurements.

Table 2: T-test results

Test	**Result**	**P-value**	**95% Confidence Interval**
F1 ANNO-BASE	Significant decrease	.0193	[-0.0043,-0.00039]
F1 CONSTITFULL-BASE	Insignificant decrease	.0832	[-0.0037,0.00023]
IGN F1 ANNO-BASE	Significant decrease	.0051	[-0.0050,-0.00093]
IGN F1 CONSTITFULL-BASE	Significant decrease	.0226	[-0.0046,-0.00036]

For each of these experiments, we find that the mean score of the training with BASE input was higher than the mean score with annotated input. These differences were found to be statistically significant in all of the experiments, except for when comparing the F1 performance of BASE to CONSTITFULL, where it was not quite statistically significant. The test has shown that our annotations actually increase the score slightly. The decrease in score is very small, however, being less than half a percentage point with 95% confidence. Several explanations for this slight decrease are possible: perhaps the annotations distract the training from more relevant information in the base data, or perhaps they cause the learning to be slightly slower, needing more than the allotted 15 epochs to reach a peak. Regardless, what this tells us clearly is that the annotations we added to the data were not useful for improving performance, since they were not capable of overcoming whatever small effects may have caused this slight decrease.

Conclusion:

ETC is an architecture designed to make use of structured data in natural language processing, and has been shown to be capable of performing at several points above baseline on document-level relation extraction on DocRED, however, the addition of automatically-annotated Ontonotes-style structured information to the dataset did not improve, and even slightly decreased, the performance of ETC on said dataset. Although by the nature of machine learning research it is difficult to say for certain why any particular result occurred, there are several potential explanations for this negative result.

First, it could be that the NLP pipeline tasks represented by the annotations are already adequately represented in the pretrained BERT model, and therefore the annotations do not add new information. This view is supported by Tenney et. al. [8], which uses a probing classifier to test how well contextual information related to the classical NLP pipeline tasks is represented in the pretrained BERT encoder layers. They found that the classifier, which only had access to the

token spans it was classifying, and none of the context required to do well on these pipeline tasks, improved significantly when given access to frozen, pretrained BERT encoder layers being run on the data. They were also able to determine which BERT layers were most important to which tasks by looking at the weights assigned to each layer by the classifier. This indicates that BERT is already representing information useful for these pipeline tasks when its encoder is run on unannotated text. It is possible that during finetuning, our ETC model was able to learn what it needed to learn from the annotations, from this information contained within the pretrained model, rendering the annotations redundant.

Secondly, it may be that the annotation quality is not sufficient to provide useful information to the model. If correct annotation requires the annotator to extract similar information from the dataset as is required for relation extraction, and the annotator is not better at doing this than the model itself, it may not have information to offer which is useful on average. Unfortunately, this study does not have the time or resources to hand-annotate the DocRED dataset with gold-standard Ontonotes annotation to see if this would improve things, so for the moment this hypothesis cannot be tested. For this same reason, the annotation quality on the DocRED data specifically cannot be tested, so we must rely on the results given by the authors of each system on the datasets they were trained on. CoNSeC and MRC-SRL are the current state-of-the-art on Senseval and CoNLL2005, with 82.7 F1 and 90 F1 respectively. Stanford CoreNLP does not have specific numbers for performance on constituency parsing or coreference datasets, but it is a well-known tool for the classical NLP pipeline. It cannot be determined with confidence whether these performances are good enough, or how well they apply to the DocRED data specifically, but for now it is unclear that an annotator can do much better than the one used in this project with current techniques, so if this is the reason for the lack of performance, it does not seem to be fixable.

It is also possible that the model cannot make sense of the encoding of this added structured information, because it was not present in the data for pretraining. It is clear that the pretraining done by the authors of the ETC paper is essential for the performance of ETC on this task, as without it, the performance drops by about 15 points. Although the pretraining included global tokens, with the value 1 denoting a sentence, the features and tokens for entities and Ontonotes annotations were added and converted to a token representation using a method designed for this particular project (as described in Appendix 1), and therefore were not present in pretraining, and would have to be learned by the fine-tuning job. If this is why the result was negative, the obvious way to fix it would be to auto-annotate the original BERT datasets and pretrain on those. However, doing this would be far beyond the resources of this study, so this hypothesis cannot be tested at the moment either.

Another possible explanation is that the performance did not improve because the full annotation information could not be added to the input due to memory concerns. However, this explanation seems unlikely, because every part of the annotation, except for the leaf nodes from the parses, was included in some of our runs, and no significant performance increase was observed. This is despite the fact that all of our annotated runs included the semantic role labeling annotations, which were the intuitive reason for why we thought these annotations

might help in the first place. It does not seem likely that even though there is no performance increase under these circumstances, the performance would suddenly increase just because all of these previously ineffective annotations were put together.

As there is one leaf node for every long token, and ETC is designed to scale well only when there are more long tokens than global tokens, there were simply too many leaf nodes to include them all, and these leaf nodes are how the part of speech tagging is represented in Ontonotes. This inability to include part of speech tagging in the annotations may be another reason for the negative result. If this is the case, the architecture would need to be completely changed, because there are as many part-of-speech tags as there are sequential tokens, and ETC is designed to scale well only with the long input, which must be encoded as a sequential BERT or RoBERTa tokenization for the pretraining to be effective. Some other method of encoding the part-of-speech information in the input would need to be devised.

In light of the negative result, future work in this area would focus on determining an explanation for why this approach does not seem to work, and seeing whether the approach could be salvaged with some sort of modification. The first thing I would do, if I had the necessary resources, would be to hand-annotate the DocRED dataset with gold standard Ontonotes-style annotations, or perhaps annotate some part of the Ontonotes dataset with DocRED relation extraction, so that this experiment could be repeated with gold-standard Ontonotes-style annotations. If gold annotations still did not improve performance, that probably means this approach is out, because it means the annotations are either irrelevant or redundant. If they did, however, there may be some hope for this approach with better annotation tools.

Another possible future direction would be to change the methodology and use these annotations as an additional source of loss for the ETC model, rather than as direct input to the ETC model. The idea would be to take predictions about the annotations as output from the model and measure these against the automatic(or gold if available) annotations, to add more loss terms. This would, in theory, direct the fine-tuning towards learning these intermediate tasks, and possibly finding ways to use these to improve the main task.

Appendix 1: Input formatting

This appendix contains explanations for each input tensor of exactly how the input is encoded in said tensor. In the code, each tensor has a first dimension of size 1 for batching purposes. We will ignore this first dimension here for simplicity.

Long Input:

The long input tensor, token_ids, is length 800. It consists of BERT WordPiece tokens, followed by padding zeros. The raw text of the DocRED example is first tokenized with CoreNLP, then each CoreNLP token is converted into one or more BERT WordPiece tokens, using the AutoTokenizer which comes with ETC.

Global Input:

The global input is divided into sections based on the type of information being represented. Each section fills up with however many there are in a given example, and the rest is a zero buffer for that section. The sections for annotated input are shown in the following table. For non-annotated input, only the Sentence and Entity sections are included. For fullconstit input, the WSD section is excluded, and the Constit section is expanded to 385 entries.

Range	**Data type**
0-29	Sentence
30-79	Entity
80-99	Coref
100-189	SRL
190-359	Constit
360-614	WSD

Entries to the global input are interpreted according to the following table. The part in parenthesis indicates what information is being encoded by the choice of value within the range. So for example, 4-9, Entity Token(Entity Type) means that the values 4-9 represent Entity Tokens, and each value in the range represents one of the 6 entity types in the DocRED dataset. The model's vocabulary size is 30522, so all these values fit in the vocabulary.

Value	**Meaning**
1	Sentence Token
4-9	Entity Token(Entity Type)
10	Coreference Token
12-89	Constit Token(Penn Treebank Node Type)
90-22874	WSD Token(WordNet Sensekey)
22875-26061	SRL Token(PropBank Frame)

g2g Relative Attention IDs:

g2g_relative_attention_ids is a tensor of size (n,n), where n is the length of the global input. For 0<=x<n, 0<=y<n, the entry (x,y) represents a relation between the xth global token and the yth global token. The meaning of the value is shown in the following table.

Value	Meaning
0-24	Relative position label between annotations of the same type
25	Overwritevalue
26	Global input entry x is contained in global input entry y
27	Global input entry y is contained in global input entry x

The relative position labels use the same convention found in ETC's feature_utils.py, with max distance of 12 because this is what the pretrained model used: 0 means x=y, 1-11 means x is 1-11 before y in the list, 12 means x is 12 or more before y, 13-23 means x is 1-11 after y, 24 means x is 12 or more after y. Overwritevalue in this context is the default value, it means that either x or y is padding, or x and y are in different sections of global input and don't have a relation between them. 26 and 27 are used for relations between sentences and other annotation types, to indicate which sentence a particular annotation occurs in (for entities and coreference it may be multiple). 26 and 27 are also used for the parsing annotation. In this case, 26 means x is above y in the parse tree, and vice versa for 27. The relative attention vocab size is 32, so these values can fit.

g2l and l2g Relative Attention IDs

If n is the size of the global input, g2l has size (n,800), and l2g has size (800,n). For 0<=x<800, 0<=y<n, the entry (x,y) in l2g and the entry (y,x) in g2l represent the relationship between the xth long token and the yth global token. The meaning of these values is shown in the following table:

Value	Meaning
0	No relation(g2l)
1	No relation(l2g)
2	x is part of sentence y
3	x is part of entity y
4	x is part of one of the mentions of coreference chain y
5-29	x is part of an SRL argument of frame y(Argument type)
30	x is part of the span of constituency parse node y
31	x is a word with sense y

The relative attention vocab size is 32, so these values can fit.

l2l relative attention IDs:
These were generated entirely with ETC feature_utils.py, as the long input is still just a series of BERT WordPiece tokens. The tensor has shape (800,169), as the attention radius around each token is limited to 84. It is filled with relative position labels as described in the g2g section, with max distance 12. Each entry (x,y) represents the relative position between long token x and long token (x+y-84).

Masks (l2l,l2g,g2l,g2g)
The masks are the same shapes as their corresponding relative attention id tensors. Each entry (x,y) is 0 or 1, 0 meaning that the attention between x and y should be ignored, 1 meaning that it should be taken into account. Except for the g2l mask, the masks are value 1 everywhere except where x or y is padding, to screen off attention between real tokens and irrelevant padding tokens. The g2l mask is also zero everywhere where the g2l relative_att_ids indicate no relation, to implement hard g2l masking.

Appendix 2: T-test training run scores
The following table shows the results of the runs used for the t-test. We report F1 and IGN F1 scores for 30 runs for each of the three input types: BASE, ANNO, CONSTITFULL(CFULL in the table). The F1 and IGN F1 measurements were taken on the same runs and are therefore not independent of each other, but the BASE, ANNO, and CONSTITFULL measurements were taken on different runs and are independent of each other, which is why we use an unpaired t-test.

BASE F1	ANNO F1	CFULL F1	BASE IGN F1	ANNO IGN F1	CFULL IGN F1
0.5868804615	0.5956238227	0.5921402363	0.5590209136	0.5667187885	0.5584469704
0.5921828649	0.5929828629	0.5926718645	0.5640985641	0.5624641365	0.5625918784
0.5942381233	0.5878051421	0.5924007729	0.5641914045	0.5572348347	0.5613378796
0.5914067628	0.5891207429	0.593207061	0.5612094178	0.5572649879	0.5640740839
0.5918634622	0.5880248046	0.5918102027	0.5633708676	0.5584796276	0.5607832606
0.5893098245	0.5899139931	0.5891258581	0.5601482552	0.5594798642	0.5597344651
0.6006335742	0.5909363033	0.5928921445	0.5704277734	0.5620221274	0.5612685485
0.5906891631	0.5903224825	0.5932245611	0.5617174989	0.5612040973	0.56267632
0.5885498938	0.5901047386	0.585338037	0.5601779638	0.5599613156	0.5544246386
0.5914028398	0.5835648307	0.5938609522	0.5584703147	0.5544314167	0.5672208708
0.5997109132	0.5951979229	0.5942029976	0.5703058123	0.5688504301	0.5655328158
0.5967766745	0.5947725336	0.5914592768	0.5683977483	0.5660470454	0.5626250067
0.5912463907	0.5976388625	0.591427684	0.5653168224	0.5682084146	0.5651679549
0.5979783278	0.5990889093	0.5898226808	0.5661447259	0.5673819528	0.557447912
0.5915715465	0.5925695029	0.5928257417	0.5621904669	0.5645531026	0.5605425029
0.5957032858	0.5927470985	0.5984986482	0.5646068917	0.5636556502	0.5674750749
0.5947830271	0.5943905713	0.5954631411	0.5640593913	0.5653220878	0.565649642
0.5899435545	0.5929019278	0.5929381567	0.5580468249	0.5637538243	0.5630476968
0.5938467702	0.5914032242	0.5942719718	0.5686083705	0.5618911129	0.564461175
0.5979382716	0.591718902	0.5955802929	0.5675499956	0.5595755361	0.5699006887
0.5920580292	0.5873893259	0.5836772599	0.5632373991	0.556704595	0.5556588056
0.5958510271	0.5891207735	0.5952333305	0.5675697349	0.55769824	0.5649982879
0.5960609001	0.5899194922	0.5941229787	0.5664699454	0.5587069052	0.565478257
0.5987090833	0.5854595752	0.5885105014	0.5702955624	0.5556262936	0.5556280705
0.5977813258	0.5931370405	0.5900945094	0.5684494382	0.5635112174	0.5580343563
0.6007806285	0.5928907546	0.5865271199	0.5709869561	0.5632583385	0.5549810439
0.5910373166	0.5913792864	0.5954144937	0.5606550173	0.560575796	0.5659808869
0.5967382321	0.5933999831	0.594713647	0.5671995592	0.5635562927	0.5663632263
0.5940042254	0.5867446319	0.5853060544	0.5668787047	0.5561651924	0.5551405778
0.586409268	0.5959273551	0.5978742504	0.5595903203	0.5660457557	0.5676927078

References:

[1] Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A Large-Scale Document-Level Relation Extraction Dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, Florence, Italy. Association for Computational Linguistics.

[2] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding Long and Structured Inputs in Transformers. In *Proceedings of the 2020 Conference on Empirical*

Methods in Natural Language Processing (EMNLP), pages 268–284, Online. Association for Computational Linguistics.

[3] Edoardo Barba, Luigi Procopio, and Roberto Navigli. 2021. ConSeC: Word Sense Disambiguation as Continuous Sense Comprehension. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1503, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[4] Nan Wang, Jiwei Li, Yuxian Meng, Xiaofei Sun, Han Qiu, Ziyao Wang, Guoyin Wang, and Jun He. 2022. An MRC Framework for Semantic Role Labeling. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2188–2198, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

[5] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven 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, Baltimore, Maryland. Association for Computational Linguistics.

[6] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards Robust Linguistic Analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

[7] Qingyu Tan, Ruidan He, Lindong Bing, Hwee Tou Ng. 2022. Document-Level Relation Extraction with Adaptive Focal Loss and Knowledge Distillation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1672–1681, Dublin, Ireland. Association for Computational Linguistics.

[8] Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

[9] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. https://arxiv.org/abs/2303.12712. Accessed May 2023.

[10] OpenAI. 2023. GPT-4 Technical Report. https://arxiv.org/abs/2303.08774. Accessed May 2023.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. 2017. Attention Is All You Need. https://arxiv.org/abs/1706.03762. Accessed May 2023.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/1810.04805. Accessed May 2023.

[13] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://arxiv.org/abs/1907.11692. Accessed May 2023.

[14] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, Zihao Wu, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, Bao Ge. 2023. Summary of ChatGPT/GPT-4 Research and Perspective Towards the Future of Large Language Models. https://arxiv.org/abs/2304.01852. Accessed May 2023.