Why did they cite that?

by

Charles J. Lovering

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

May 2018

APPROVED:

Professor Jake Whitehill, Thesis Advisor

Professor Joseph Beck, First Reader

Professor Craig Wills, Head of Department

Abstract

We explore a machine learning task, evidence recommendation (ER), the extraction of evidence from a source document to support an external claim. This task is an instance of the question answering machine learning task. We apply ER to academic publications because they cite other papers for the claims they make. Reading cited papers to corroborate claims is time-consuming and an automated ER tool could expedite it. Thus, we propose a methodology for collecting a dataset of academic papers and their references. We explore deep learning models for ER and achieve 77% accuracy with pairwise models and 75% pairwise accuracy with document-wise models.

Acknowledgements

I would like to thank my advisor, Professor Jake Whitehill, for helping learn how to better approach research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The focus of this work is creating a model that recommends the sentences in a referenced document that provide the most evidence for a given claim. We name the process of recommending evidence for a claim as *evidence recommendation*. By focusing human attention to the most relevant portion of the cited text, an application of our model can save human effort. This work will help researchers with literature reviews and reviewers check conference submissions by automatically finding corroborating evidence to a claims in a referenced documents. Evidence recommendation can also help a reader better understand previous work that supports the citing document by providing evidence the reader may have missed or would not have had time to read. Evidence recommendation could also help readers determine which paper best supports a claim and if a paper was cited erroneously.

We analyze text at the sentence level. We define a *claim* to be a sentence citing material from another paper. Likewise, *evidence* for a given claim is a sentence in a referenced paper that corroborates the assertion made by the claim. Each citing document can have multiple claims. Each claim can refer to multiple different referenced documents, and each cited document can have multiple pieces of evidence.

1.1 Motivating Example

When Jane Doe is reviewing a paper on the trustworthiness of classifiers [44], she reads the following claim:

> `Data leakage, for example, defined as the unintentional leakage of signal into the training (and validation) data that would not appear when deployed [29], potentially increases accuracy.`

If she is unfamiliar with "data leakage", she may turn to the referenced paper [29]. Reading the entirety of the referenced paper would help her to confirm the definition and the motivation behind the term, but would require a lot of time. If she instead could look at a thresholded heat map on the paper where the color indicates a particular sentence likely has evidence that supports the claim, as demonstrated in Fig (1.1), she would be able to directly locate evidence to the claim. This will help Jane quickly find the corroborating evidence, and better understand the underlying material.

1.2 Related Work

An automated tool for evidence recommendation for academic papers has never been made to the best of our knowledge. There are many similar citation recommendation systems [23, 32, 45] that recommend papers to cite for some text as a utility for researchers. Some of these recommendation systems utilize the text surrounding the citation [5, 23]. This surrounding text is called the context [5, 23], and corresponds to term *claim* in this proposal. Alzahrani et al. [5] used the context for plagiarism detection and Aya et al. [6] used the context to classify the citation itself as one of

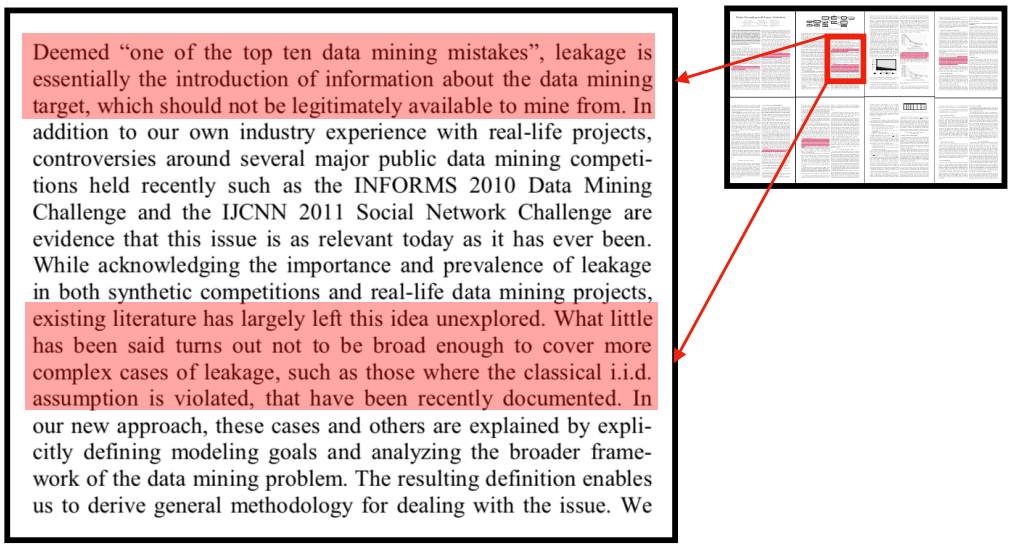

Figure 1.1: Demonstration of a thresholded heat map on the entirety of an academic paper where the temperature highlights potential *evidence* for a given *claim*. Left, is a zoomed-in example of candidate sentences. Right, is an overview heatmap that would allow a reader to locate potential evidence.

evidence, motivation, etc. These works operate on document level statistics, and as we work directly with sentences, we look at other works for guidance.

Question Answering [43] and Natural Language Inference [9] are two open problems in Natural Language Processing that are directly related to our task. Question answering (QA), a machine learning task where a model tries to find the answer to a question from within multiple documents, is similar to evidence recommendation. In fact, evidence recommendation is an instance of QA. The question takes the form of: “What sentences provides evidence for this $\{claim\}$?”. There are many different research efforts focused on improving QA models. The recent SQuAD dataset [43] has resulted in the creation of numerous new models [26, 36, 43, 52].

It has been argued that models capable of Natural Language Inference (NLI) [9] capture important semantic information and demonstrate some level of reasonable language understanding. NLI is the task of determining the relationship between sentences; whether a sentence contradicts, entails or is has neutral relationship to

the another. Recent efforts in models for NLI [15, 41, 48, 51] are being driven by the release of the SNLI [9] and MultiSNLI [40] datasets. For us to be successful in this project, our model must be capable of NLI because evidence recommendation requires quantifying the extent a sentence entails another sentence. This is complicated by the fact that surrounding sentences may provide either a contradictory or supportive contextual information. For example, an example *text* and entailed *hypothesis* from the SNLI dataset are reproduced below.

Table 1.1: Example of sentences from SNLI Dataset

Premise	**Hypothesis**
A soccer game with multiple males playing.	Some men are playing a sport.

For the SNLI task a model must infer that soccer is a sport and that multiple males are men. In evidence recommendation, the descriptions earlier in a document may have to be interpreted for the evidence provided by the sentence to be fully evaluated; i.e. the context of the document as a whole is important. A hypothetical example of this is shown below.

Table 1.2: Hypothetical Example of Evidence Recommendation

Evidence (Premise)	**Claim (Hypothesis)**
The children went outside. The males started playing a soccer game. That night their fathers played tennis.	Some men are playing a sport.

In this example, the second sentence (modeled after the text from SNLI) is not evidence for the claim because the males refers to children not adults. The last sentence is evidence because it does entail that male adults are playing a sport. This example indicates some of the difficulties of ER, like the importance of context, but the recent success in both QA and NLI inform us how to approach modeling this

type of problem.

There has been some recent work [27] in argument analysis that parallels our work. The application and focus of Hua et al. is to classify different arguments used in a debate. Fundamentally, this also involves finding and understanding evidence. There are some subtle differences in that argument classification focuses primarily on the type of argument, not necessarily its strength or relative importance. We will leverage their dataset to compare and evaluate our models in a different context.

Lastly, we utilize pairwise annotations as detailed in §4. In the learning to rank paradigm, the task of finding order among items, training models on pairwise annotations is a common approach [11]. Previous work in learning to rank is represented by Burges et al. with LambdaRank and LambdaMART [11, 12]. In this work, we attempt to wed the semantic interpretation of our NLI-based model, with the capacity of a ranking algorithm. Further exploration in fully utilizing these ranking methods, as mentioned in §8, is an exciting area for future work.

1.3 Challenges

There are multiple approaches to question answering, and relevant underlying methods like classification and ranking, but exactly how to formulate evidence recommendation is unclear because the application is novel. This consists of two primary challenges.

- Formulating a labeling task that ordinary people working on crowdsourcing websites can perform to provide us with meaningful labels.
- Developing a machine learning model that is able to automatically recommend evidence.

1.4 Scope of Work

We are interested in the semantic correlation between sentences. So, our approaches do not focus on matching numeric values. However, this is an important feature that could be added to a final product. Also, we are not concerned with determining the actual reason a publication was cited (e.g. for comparison, for evidence, etc.) or why a particular publication was chosen instead of other similar previous works. In addition to the citation recommendation described above, scientometrics [35] studies the impact of publications and the automatic indexing of citations [22]. These concerns are related to our work, but are a different type of problem that we do not try to solve.

1.5 Approach

We determine how to formulate our problem empirically. Candidate options include framing the task as classification, where labels indicate whether or not a sentence is a piece of evidence, and ranking, where the best evidence is labeled as most important. Classification allows for a more precise approach as evidence would be directly labeled as such. Ranking alternatively offers an ordinality among pieces of evidence that may better allow users to more quickly find the best evidence. We annotate our dataset via pairwise comparisons, as detailed in §4, because of the higher labeling accuracy that we obtain with ranking compared to classification. We propose exploring a range of deep neural models and building a prototype application that utilizes our trained model. Our approach can be broken down into the following tasks:

1. Define a data collection methodology for collecting a set of academic papers and their referenced academic papers.

2. Develop annotation tasks and select an annotation aggregation technique.

3. Determine our machine learning task based on our evaluation of different annotation techniques.

4. Design models for evidence recommendation.

5. Deploy a prototype application utilizing constructed models.

6. Evaluate our results.

1.6 Remainder of the Document

Background to relevant machine learning is given in §2. The data collection methodology is detailed in §3. We evaluate the different ways of annotating our dataset in §4. We empirically find that ranking is an effective formulation in §4.4 for our problem. We define our problem formulation and propose model architectures in §5. The evaluation for our models and our proposed prototype application (see §6) is in §7. The evaluation is presented in §7.1. We describe future work in §8 and conclude in §9. Lastly, the notation used in this document is defined in appendix 2.3.

Chapter 2

Background

In this chapter we review techniques relevant to our approach. First, we describe the machine learning task of ranking. Ranking applies to our problem for a two different reasons. We find empirically that annotators have the highest agreement when labeling via pair-based comparisons §4. Pairwise annotations allow us to either generate a ground-truth ranking use that to label our dataset. Alternatively, we can use the pairwise labels to train our models to directly learn a ranking function. We explore previous work in ranking to inform the tradeoffs of this decision. We also introduce current methods in natural language processing. Lastly, we define the notations used in this document.

2.1 Ranking

Ranking sorts items into a desired permutation. Thus, the goal is to find a function h that is a bijective function that maps from $X \to X$ [57]. However ranking functions usually optimize a function which operates on a single item [57] for efficiency concerns. So, in order to find an ordinal relationship among items, a function $\psi : x_i \to \mathbb{R}$, is optimized such that if $\psi(x_i) > \psi(x_j)$ the rank of x_i is greater than that of

x_j [10, 16, 57]. Items are then sorted using their corresponding scores. There are three different high-level approaches for training the ranking function ψ.

2.1.1 Ranking Approaches

Pointwise models operate on a single instance, and are reduced to regression or classification paradigms [16]. MSE is a commonly employed loss (Eq. 2.1) where y_i is either $\in [0, 1]$ or assigned to one of $\{0, 0.5, 1\}$ [10] and regress real valued scores $\in \mathbb{R}$. Other losses like the hinge loss and log loss [16] are also common. In some cases, relevancy labels can be discrete like $\{0, 1, 2, \dots\}$ in which case cross entropy could optionally be used instead of MSE. In Eq. 2.1, x is a set of items and y is the set of labels (real-valued or discrete) for each of those items. Note that, as shown below in §2.1.2, ranking evaluation metrics use the raw values of the relevancy labels rather than viewing them as classes.

$$L(\psi; x, y) = \sum_{i=1}^{n} (\psi(x_i) - y_i)^2 \tag{2.1}$$

In our context, a pointwise model would operate on individual sentences. Candidate annotations for this type of approach include labeling each sentence as evidence or not, or optionally having different levels of relevance for each sentence, e.g. assigning a score $\in \{0, 0.5, 1\}$ to each sentence.

Pairwise models like RankSVM [24], RankBoost [21], and RankNet [10] are trained directly to order pairs. Burges [10] finds that the set of pairs do not need to be complete nor consistent to be effective. As noted by [16], the general form for pairwise loss functions is as follows:

$$L(\psi; x, y) = \sum_{(i,j):y_i > y_j} \phi(\psi(x_i) - \psi(x_j)) \tag{2.2}$$

In Eq. 2.2 x is a set of items, and y are the corresponding labels. Labels in y can be a binary values, a discrete relevancy value like in the pointwise formulation, or the position in the desired permutation. This formulation also works for a set of pairwise comparison annotations. The specific function ϕ depends on the approach (e.g. RankSVM uses a hinge loss). There is significant research exploring these models: kernel methods for improving RankNet's accuracy [33] and techniques for increasing its scalability [34] have been explored.

In the context of ER, each pair will be two candidate evidence sentences and the label will indicate which of the two sentences provide more evidence.

Listwise models also have been explored. Their loss function operates directly on the desired permutation. There are different formulations for listwise functions [11, 12, 14, 57]. As an example, the loss function of ListMLE [57] which penalizes items that have greater scores which are ranked lower than itself. However, even ListMLE's ranking function only operates on a single item. Each item in a sample (which is a set of items) is scored, and then the items are sorted into the output permutation. Because sorting is not differentiable, a likelihood loss is used as a "surrogate" as shown in Eq. 2.3. In Eq. 2.3 x is a sample of items to be ordered and y is desired permutation.

$$L(\psi, x, y) = -\log P(y|x, \psi) \tag{2.3}$$

$$P(y|x, \psi) = \prod_{i=1}^{n} \frac{\exp(\psi(x_{y(i)}))}{\sum_{k=i}^{n} \exp(\psi(x_{y(k)})} \tag{2.4}$$

In summary, the integral ranking function ψ for these different approaches operates on a single item and returns a real-valued score. This score is not typically probabilistic; its meaning is relative to other item scores.

2.1.2 Ranking Evaluation

The evaluation metrics of ranking are used to measure how well the integral ranking function that scores items translates into the desired permutations of items. Here we describe Mean Average Precision, which is often used for ranking paradigms with pairwise or binary labels, and Normalized Discounted Cumulative Gain, which works for multiple levels of relevancy.

For the following metrics we define π_ψ as the permutation generated by the ranking function ψ; $\pi_\psi(x) = \text{argsort}([\psi(x_i)|i = 1..|x|])$. Here, argsort returns the indices that would sort the scores. We also define l to return the labels from a permutation π as they index into y; $l(y, \pi) = [y_{\pi_i}|i = 1..|\pi|]$. Indexing into $l(\pi)_i$ returns the relevancy of the the i^{th} item in the ranking.

Mean Average Precision (MAP) is built on top of Precision@k [7]. Precision@k is the fraction of relevant documents found in top k documents. MAP is Precision@k averaged for k up to the number of relevant documents. This is defined for binary labels where relevant items ar have labels of 1 and other items have labels of 0. We define MAP in Eq. 2.5 as in [16] where n_1 is the number of items (derived from y) that are equal to 1 and I is the indicator function.

$$\text{MAP}(\psi, x, y) = \frac{1}{n_1} \sum_{s=1}^{n_1} \text{Precision@}s(l(\pi_\psi(x))) \tag{2.5}$$

$$\text{Precision@}k(\pi) = \frac{\sum_i^k I(\pi_i = 1)}{k} \tag{2.6}$$

Normalized Discounted Cumulative Gain (NDCG) [53] is a normalized measure that penalizes out-of-place items in a permutation. It is defined below in Eq. 2.1.2. It calculates the Discounted Cumulative Gain (DCG) which is a smoothed measure that penalizes highly ranked items ranked lowly. NCDG is normalized by the Ideal Discounted Cumulative Gain (IDCG) which is the maximal value of a DCG given a set of queries. DCG rewards ranking items with high relevancy scores higher, and discounts the scores of items lower in the ranking.

$$\text{NDCG}(\psi, x, y) = \frac{\text{DCG}(\psi, x, y)}{\text{IDCG}(\psi, x, y)} \tag{2.7}$$

$$\text{IDCG}(\psi, x, y) = \max \text{DCG}(\psi, x, y) \tag{2.8}$$

$$\text{DCG}(\psi, x, y) = \sum_{i=1}^{n} G(l(\pi_\psi(x))_i) \times D(i) \tag{2.9}$$

$$G(z) = 2^z - 1 \tag{2.10}$$

$$D(d) = \frac{1}{\log_2(d+1)} \tag{2.11}$$

$$\tag{2.12}$$

Note that $G(.)$ is an increasing function (i.e. gain) and that $D(.)$ is a decreasing function (i.e. discount) [16]. In practice, people are often interested in NDCG@k

which is the NDCG up to an index k. In these cases, $D(d)$ can be set to zero when $d > k$. NDCG is discontinuous and thus not differentiable, so it is not commonly used directly for training models. However, it is found that losses on most pointwise and pairwise losses (e.g. cross entropy of pairwise scores) approach the behavior of NDCG [16].

2.1.3 Generating Listwise Orderings from Pairwise Labels

Instead of using the raw annotations we collect in §4, we could preprocess these annotations into rankings. Wauthier et al. [55] proposes a simple algorithm Balanced Rank Estimation which estimates the rank $\hat{\Pi}$ of an item j among n items as follows:

$$\hat{\Pi}(j) = \frac{\sum_{i \neq j} s_{i,j}(2c_{i,j} - 1)}{p} \propto \sum_{i \neq j} s_{i,j}(2c_{i,j} - 1)$$

where $s_{i,j}$ is the binary variable indicating whether or not $c_{i,j}$ was measured, $c_{i,j}$ is a binary variable indicating whether or not item i has higher rank than j, and p is the probability that the comparison $c_{i,j}$ was measured. We considered this technique as it could help us better utilize our pairwise annotations. The benefit of this approach is that the labels would be unified and consistent. However [10] finds that the set of pairs trained upon do not need to be complete nor consistent. So, we forgo using this technique for simplicity.

2.2 Modeling Natural Language

Recent approaches to NLP use deep learning models like convolutional neural networks (CNN) [30] and recurrent networks such as long short-term memory (LSTM) [25] and the gated recurrent unit (GRU) [17]. These deep learning methods have had marked success [47] increasing previous benchmarks. Also they do not rely on

external dependencies (e.g. part-of-speech annotations [39]), with the exception of optionally using word embeddings.

2.2.1 Word Representations

Word embeddings are representations of words in real-numbered vector space. These representations created by training the embeddings on large corpora of documents, and map individual words to dense distributional vectors that capture semantic meaning. Comparatively, the traditional methods represent words as atomic units, typically via a *1 hot vector* [20]. These atomic representations do not capture semantic meaning. For example, the *1 hot vectors* of "dog" and "cat" have no relation, but their respective word embeddings may capture that they are nouns, that they are both pets, and that they are similar. This can be vizualized using t-SNE[1] [37]. There are several techniques used to create word embeddings. The most popular options are Word2Vec [38], which is trained with contextual information of a sliding window of words across a corpus of documents, and GloVe [42], which is trained using global co-occurrence statistics among words. There has also been success with character-aware models [31] that use convolutional neural networks to process the input that the character level.

Recent effective NLP models represent words with concatenated vectors from Glove [42] and character-aware convolutional models [31]. In [46] these representations are concatenated together and then passed through a highway network [50].

$$\tilde{x} = [\text{Char-CNN}(word); \text{Glove}(word)]$$

$$T(\tilde{x}) = \sigma(W_{(T)}\tilde{x} + b_{(T)})$$

[1] An excellent demonstration of t-SNE can be found at [54].

$$x = T(\tilde{x}) * \tilde{x} + (1 - T(\tilde{x})) * (W\tilde{x} + b)$$

This will encode semantic information derived from the co-occurrence statistics, as well as potential sub-word word information. This information may be from extracting morphemes or even capturing new out-of-vocabulary acronyms and names. Similarly, a recent work with ConceptNet [49] encodes information beyond statistical measures. By using external sources, ConceptNet attempts to *ground* word representations with pragmatic meaning. Specifically, rather than semantic meaning which is the meaning of a word given its context in language, (which corresponds with distributional and co-occurrence statistics), pragmatic meaning is directly tied to what the word means in the real world. Thus, ConceptNet attempts to encode knowledge graphs, characters, morphemes, and other embeddings like word2vec and GloVe. We do not use this in our work, but it is a potentially fruitful resource for future work.

2.2.2 Attention

Effective models in difficult tasks like MultiSNLI and SQuAD use attention [13,26,46]. Attention directly represents the relevance of an entity to a task. Plausibly, it enables some separation of concerns; early modules in a larger network are free to encode the meaning and context and high-level attention layers determine an entity's relevance. There are different approaches to determining attention. In general, candidate entities' representations are scaled by a probability distribution. The probability distribution is derived from the similarity between the entities and some context vector or query. We will formalize our application of attention in §5.

2.3 Notation

We use notation found in related work [10, 46, 57, 58], and summarize specifics in table 2.1 below.

Table 2.1: Notations

Symbol	Interpretation
$u \circ v$	element-wise multiplication of vectors
$[u; v]$	horizontal concatenation of vectors
$[u \| \| v]$	vertical concatenation of vectors
$\|S\|$	magnitude of set S
$z_{(l)}$	parameter pertaining to a given layer l.
$u^\intercal v$	transpose of u and dot product of u and v.
σ	Logistic sigmoid function.

Chapter 3

Dataset Collection

We collect a corpus of academic papers. We will need to annotate the dataset we collect, and this is described in §4. Annotation requires a massive amount of human labor. We explore crowdsourcing the annotation tasks to workers with Amazon Mechanical Turk [1] because of its potential to scale at a low cost. For the academic discipline with explore evidence recommendation with, we chose education. We do not expect our workers to have deep insights into the subject matter. However, we do anticipate that the terminology used in educational research papers will be more accessible than the jargon from other domains like theoretical physics or mathematics. Even though our workers may not fully grasp a paper's meaning, we hypothesize that they will understand enough to provide annotations that a machine learning classifier can use to automatically find relevant evidence.

We collected 500 papers from American Educational Research Association (AERA) [4]. The academic papers are converted into raw text from PDF using pdfminer [3]. From here, we tokenized the documents into sentences using NLTK [8] (a python library for text processing). We then parsed references and citations from the documents using regular expressions. We compared the results of our regular

Table 3.1: Summary of Document Statistics

Description	Size
Number of referencing documents	500
Number of referenced documents	1,100
Average number of sentences per referenced document	210

expressions against results from ParsCite [2], and found our more direct approach more effective in finding valid citations for our dataset which uniformly uses the APA format. In a pilot analysis of a small sample of files, we found our direct method was able to find and correctly parse a higher percentage of the citations. As the focus of this proposed study is the capability to recommend the best sentences, and not (at least initially) a generalized end-to-end system that parses and fetches documents, we move forward with our simplified approach.

After parsing these references from our initial 500 papers, we downloaded a subset of all the referenced documents. Full details can be seen in Table 3.1. The average number of sentences per referenced document is 210.

In order to reduce the number of sentences we need to label, we sampled sentences by their similarity from the referenced document to the claim. We do this because otherwise the cost to label all the sentences would be prohibitively high.

We determined similarity by using InferSent [19], a model which utilizes GloVe [42] word embeddings, and embeds sentences into vectors that capture semantic meaning. We selected evidence sentences to label based upon their cosine similarity to the corresponding claim's vector via $f_{sim} : c \times e \to \mathbb{R}$ as shown in Eq. 3.1.

$$c = \texttt{InferSent}(claim)$$

$$e = \texttt{InferSent}(evidence)$$

$$f_{sim}(c, e) = \cos(c, e) \tag{3.1}$$

In a pilot analysis we investigated the efficacy of using InferSent to capture the best pieces of evidence in a document within best k sentences. To do this, we hand-labeled a small sample of 5 different documents. We labeled the sentences to find the pieces of evidence in the document, so each sentence was labeled as evidence or not. We then found that when we ordered the sentences by similarity using InferSent, the top 10 sentences capture most of the evidence. This is shown in Figure 3.1. This is not a rigorous experiment, as this manual labeling is prohibitively expensive, but it gives us confidence in our sampling methodology.

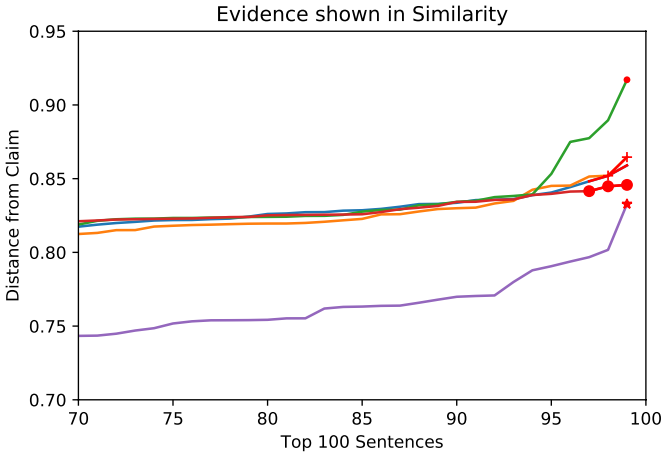

Figure 3.1: Red symbols indicate the sentences that provide the most important evidence to a claim within a document. Symbols are differentiated for clarity. Each line represents all the sentences in a distinct document.

3.1 Dataset Collection Discussion

After collecting this dataset we have a large amount of data that pertains to our task, i.e. the content of papers that cite each other. We also have selected a subset of the relevant sentences for labelling. However, this is not sufficient for training machine learning models for evidence recommendation. In the next section (§4) we discuss how we label this dataset.

Chapter 4

Dataset Annotation

The technical challenge we address in this section is how to define our problem. Our goal is to create a model that can determine which sentences in a referenced document support a claim from the corresponding referencing document. However, the machine learning task utilized to do this is flexible. The best machine learning task for evidence recommendation is an empirical question that hinges on how effective human labelers are on that task for our data. For us this is particularly important because the task is difficult, and so it is advantageous for us to frame the task in the way that works best for the labelers.

We explore two approaches for our task. First is classification. Classification is the giving an entity a discrete label from a predetermined set. Ranking is ordering items into some desired ordering. Ranking methods often are simplified to comparing pairs of items to determine which is more relevant. From here, the comparisons can be used to create a global ranking directly [28]. Alternatively, scores given to each sentence can be used to sort the sentences into a ranking. For each of these methods we need to annotate the dataset in different ways. The specifics for each method vary, but we can compare the quality of different annotation configurations using

metrics.

We also compare two aggregation methods. For each label, we have multiple annotators label that task. The methods we compare is majority vote and GLAD [56]. Majority vote takes the mode class between an odd number of votes, whereas GLAD determines the most likely label by harnessing the inter-labeler agreement among labelers to simultaneously infer the expertise of each labeler, and the difficulty and ground-truth of each item. These methods help smooth the noise from the annotations of malicious, unskilled, or simply incorrect workers.

4.1 Preliminary Unsatisfactory Results

Before beginning our formal experimentation we quickly found negative results with other methods. We include them here for completeness.

Unsatisfactory results with 4-class classification We also tried classification with 4 classes and found those labels to be random with Cohen scores of near 0 and area under AUC near 0.5.

Unsatisfactory results with entire-document tasks We first set the task as finding the single best piece of evidence in a referenced document. However, our experiments indicated that this task did not achieve a high throughput in terms of worker interest, and that answers we received were always just the first approximately related sentence in the document, if not random.

4.2 Experimental Design of Comparable Paradigms

To compare these methods, we build user interfaces for annotators to label sentences. For each configuration we manually label a small validation dataset. For each task, we have annotators label it multiple times. We will compare aggregations and paradigms by evaluating the results across several holistic metrics (described below).

Metrics We compare the annotation configures across the cohen score, the ROC, and the F1 Score. The cohen score is a measure of annotator agreement [18] which has a range from -1 to 1 where less than 0 indicates random annotations and 1 is perfect agreement. The area under the AUC is a metric to evaluate classifier output quality and the F1 scores is the harmonic mean between precision and recall. AUC is invariant to class imbalances [20], and F1 scores are comprehensive in that its value captures more nuance than accuracy.

4.3 Task Definition

Binary Labels for Classification We compare binary classification (BC) of a sentence into two different sets of discrete labels, { `relevant, not relevant` }(BC-R) and { `evidence, not evidence` } (BC-E). Each of these result in valuable annotations, but it is unclear which *wording* more clear to a labeler. Figures A.1 and A.2 show both annotation tasks for classification. These two tasks are similar with variations in the instructions and the prompt.

Binary Comparison for Ranking Labels Given two candidate sentences, and the evidence sentence, the task is to determine which sentence provides more evidence. The interface for the binary ranking (BR) task is similar to the classification interfaces,

but instead presents two candidate evidence sentences as shown in Figure A.3.

4.4 Evaluation of Annotation Tasks

We compare two paradigms, binary ranking (BR) and binary classification (BC), in Table 4.1. We get these results from evaluating annotations from crowdsourced tasks versus our ground truth annotations. Binary ranking outperformed binary classification across all three metrics. The best result, ranking with labels aggregated by majority vote on 5 votes, has a AUC is 0.69.

Table 4.1: Annotation Task Evaluation

Paradigm	Aggregation	Cohen Score	AUC	F1 Score
BR	Vote 3	0.29	0.64	0.64
BR	Vote 5	**0.39**	**0.69**	**0.69**
BR	GLAD 3	0.22	0.61	0.61
BR	GLAD 4	0.19	0.59	0.59
BR	GLAD 5	0.26	0.63	0.63
BC-E	Vote 3	0.16	0.61	0.55
BC-R	Vote 3	0.04	0.52	0.56

Table 4.2: These results are a comparison of annotations from crowdsourced tasks compared to our ground truth annotations. BR is binary ranking used for the ranking paradigm. BC-* is binary classification. BC-E task asks whether a sentence is evidence or not. BC-R task asks whether a sentence is relevant or not. Both tasks used the same data, but required different number of questions and tasks. There are 100 comparisons for BR and 115 tasks for BC. The aggregation indicates how the labels were formed: Vote indicates majority vote, whereas GLAD uses the optimal aggregation method [56]. The number that follows indicates the number of annotations per task that were aggregated.

4.5 Dataset Details

We used pairwise annotations to label our dataset. So far, we have had 7265 tasks labeled. We have aggregated this into 1453 majority-vote labels. Below is a

summarization of our dataset and notes on how we ensured no data leakage between splits.

1. We balanced the dataset so there are exactly equal number of positive and negative items, by reordering the items in a binary comparison. This allows us to look at accuracy as a meaningful metric without skew due to unbalanced class distributions.

2. We stratified the dataset so there is no crossover of referenced or referencing documents between train, validation and test splits. To do this, and split the partitions with appropriate sizes, we use a standard knapsack packing algorithm.

3. We created an *augmented* dataset with relatively cheap data points. For items that were only ranked positively, we added binary comparisons where these positively ranked items are positively compared versus a sample of points that were previously discarded from within the document as less relevant. This allows us to automatically grow our dataset. However, this reduces the meaning of the labels.

4. We examine our models performance on other aggregation techniques as well. Namely, we reduced our primary dataset to a smaller subset termed "high majority" where we only kept items that were voted at a rate of at least 4 out of 5. We also examine "unanimous" where we only keep items that have labels agreed upon by all workers.

The current magnitudes of the dataset are presented in Table 4.3. We could increase the size of the augmented dataset almost arbitrarily.

We show the counts for the document versions annotations below in Table 4.4.

Table 4.3: Dataset Item Counts

Dataset	Train Size	Validation Size	Test Size
majority	1017	201	256
high majority	457	111	93
unanimous	109	26	20
augmented	7780	1565	1393

Table 4.4: Document Dataset Sizes

Train (# of Documents)	Validation (# of Documents)	Test (# of Documents)	
citation	52	11	14

Chapter 5

Machine Learning Models for Evidence Recommendation

In this chapter we propose machine learning models for evidence recommendation.

5.1 Problem Formulation

After empirically evaluating different paradigms and aggregation methods, we found a comparison based method most effective. We extract a set of tasks $T = \{(\tau_i)\}_{i=1}^{N} \mid \tau_i = \{c_i, e_i^1, e_i^2\}$ from collected dataset (see §3) where c_i is a claim, e_i^1 and e_i^2 are candidate evidence sentences that are to be directly compared. Each task τ_i is labeled by annotators as one of `0` or `1`. When y_i is `0` it indicates e_i^1 is more relevant, and e_i^2 otherwise. Each task has multiple labels from different mechanical turk workers. Finally, we have a labeled dataset D that consists of pairs of tasks τ_i and labels y_i.

$$D = \{(\tau_i, y_i)\}_{i=1}^{N}$$

We approach evidence recommendation by providing an order among a pair of

candidate evidence sentences, i.e., denoting which sentence provides more evidence. Using this, we can recreate a global ranking among candidate evidence sentences to find the strongest pieces of evidence. One can either use a method to recreate a global ranking [28] or instead uses the outputted scores for each sentence to sort the sentences. We opt for the latter because of its simplicity.

5.2 Loss function

The objective for each model ψ is to output a real value that corresponds to the relative rank of the item; $\psi : c \times e \times \theta \rightarrow \mathbb{R}$. The parameters of the model θ are learned through different training schemes and are dependent on the architecture of the model ψ. We elide θ from further notations for clarity. The loss for each model ψ described below is as follows.

$$L : D \rightarrow \mathbb{R} \qquad\qquad L(D) = \frac{1}{|D|} \sum_{\tau_i, y_i \in D} L_i(\tau_i, y_i) \tag{5.1}$$

In Eq. 5.1 we averaged the total loss across all labeled tasks.

$$L_i : \tau \times y \rightarrow \mathbb{R}, \quad L_i(\tau, y) = \mathcal{H}(\psi(c, e^0), \psi(c, e^1))^y \times \mathcal{H}(\psi(c, e^1), \psi(c, e^0))^{(1-y)} \tag{5.2}$$

The loss for each annotated instance in the dataset is the hinge loss, Eq. 5.3, of the model's score for the candidate evidence sentence that is labeled as providing more evidence compared to the evidence sentenced that is labeled as providing less evidence.

$$\mathcal{H} : s_1 \times s_2 \to \mathbb{R}, \qquad \mathcal{H}(s_1, s_2) = \texttt{max}(\texttt{0}, \xi + s_1 - s_2) \tag{5.3}$$

Again, Eq. 5.3 is the hinge loss, with the margin ξ `= 1`. However, in practice when we train our models we use a more compact definition (see Eq. 5.4). We keep the original definition for completeness.

$$L_i^* : \tau \times y \to \mathbb{R}, \qquad L_i^*(\tau, y) = \mathcal{H}^*(\psi(c, e^0), \psi(c, e^1), y) \tag{5.4}$$

$$\mathcal{H}^* : s_1 \times s_2 \times y \to \mathbb{R}, \qquad \mathcal{H}^*(s_1, s_2, y) = \max(0, -y * (s_1 - s_2) + \xi) \tag{5.5}$$

Eq. 5.4 captures our requirements for the loss. A pair of compared scores s_1 and s_2 are compared. If y is positive then the first score (s_1) should be ranked higher, otherwise s_2 should be ranked higher.

5.3 Model Formulation for Direct Pairwise Comparison

This section covers the model definition for a model that processes a candidate evidence sentence and a claim sentence and is trained in a pairwise fashion. We use the pairwise loss defined above in Eq. 5.4.

5.3.1 Base Implementation

A recurrent network (a GRU or a LSTM) individually processes the claim and the candidate evidence sentence. The final hidden state for each sentence is concatenated

together along with an element-wise multiplication of these vectors. This joined representation is then run through a linear layer. This is demonstrated in Figure 5.1. This captures representations of both sentences and their interaction.

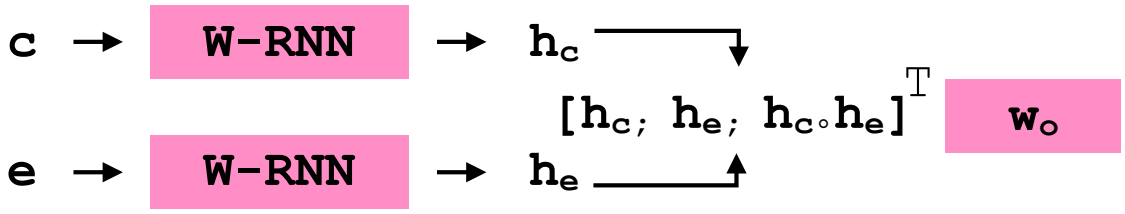

Figure 5.1: Sentence encodings are concatenated together.

In these figures W-RNN is one of RNN, GRU, or LSTM. For our initial versions we select the final hidden state as the sentence representation for both the claim sentence c and the candidate evidence e.

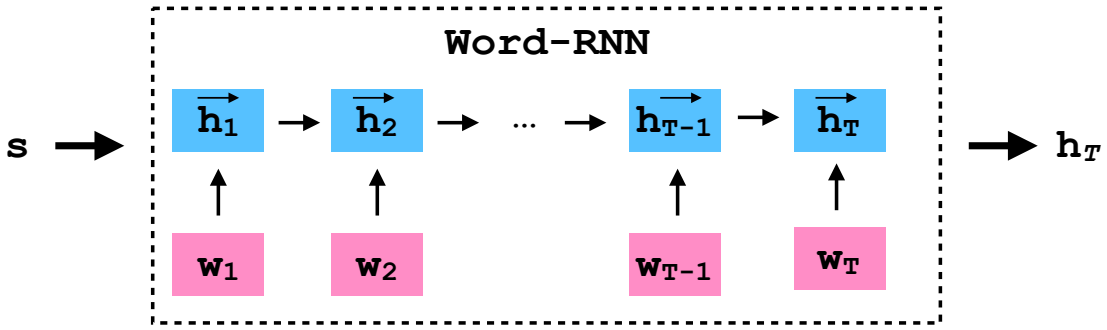

Figure 5.2: Recurrent network overview.

It is common for recurrent networks to process sequences in both the forward and backward directions [46]. Bi-directional models are often able to better capture semantic information. These output hidden states from both directions are concatenated together for the final hidden state output. The final hidden states for each direction when concatenated together have a dimensionality of $d_{(intra)}$. The word *intra* refers to the intra sentence context that this recurrent network encoded.

Lastly, our approach to modeling the interaction between the claim and the evidence is one of several different operations we could have applied. Other options include concatenation, hadamard product, cross product, and cosine similarity between the final claim and evidence representations. We opted for the concatenation of the claim representation, evidence representation, and hadamard product of the claim and evidence representation.

Formally, our base implementation is defined below for a single task of a claim c and evidence e. The J words in the evidence $\{e_1, e_2, \dots e_J\}$ and the I words in the claim $\{c_1, c_2, \dots c_I\}$ are the words are embedded using Glove [42]. Each word vector is $\in R^{d_{(word)}}$ and the output dimensionality of the recurrent networks is $d_{(intra)}$. In our implementations, these are hyperparameters, but informally $d_{(word)} = 200$ and $d_{(intra)} = 80$. Note that a more powerful representation could be constructed by additionally using a character encoding as mentioned in §2.

$$c \in \mathbb{R}^{I \times d_{(word)}} \tag{5.6}$$

$$e \in \mathbb{R}^{J \times d_{(word)}} \tag{5.7}$$

$$C = \text{BI-RNN}(c) \in \mathbb{R}^{I \times d_{(intra)}}$$

$$E = \text{BI-RNN}(e) \in \mathbb{R}^{J \times d_{(intra)}}$$

$$C_l = \text{SELECT-LAST}(C) \in \mathbb{R}^{d_{(intra)}}$$

$$E_l = \text{SELECT-LAST}(E) \in \mathbb{R}^{d_{(intra)}}$$

$$V = [C_l; C_l \circ E_l; E_l] \in \mathbb{R}^{3d_{(intra)}}$$

$$y = f_{(o)}(V) = W_{(o)}^{\intercal} V + b_{(o)} \in \mathbb{R}^1$$

As we noted, the underlying RNN implementation can be either GRU or an LSTM. Both architectures have similar performances [17] across many different tasks. We use an LSTM as defined in [17]. For the bidirectional architectures, we use a SELECT-LAST function to extract and concatenate the last hidden state in terms of time step for both directions.

5.3.2 Bidirectional Attention Flow for Ranking

Below we present a formulation of attention modified from Bidirectional Attention Flow (BiDaf) [46]. In this section we describe how we create a contextual representation of the task, that accounts for what is important in the evidence with respect to the claim and what is important to the claim with respect to the evidence. This formulation is more natural for our problem than learning a general contextual vector that is used to apply attention (as in [58]). A generalized context vector is less applicable in the case where scores for task are relative only to the input (the claim), not predetermined classes (like a static representation of a cat). This model architecture is illustrated in Figure 5.3.

Word Representation

The claim c and the evidence e are represented as sequences of word embeddings (like in Eqs. 5.6, 5.7).

Contextual Embedding

The contextual information of both the claim and the evidence is captured via a bi-directional recurrent network. Again, this is the same formulation as the base model. Note that this layer of the model is bi-directional the output dimensionality for each time step would be $2d_{(intra)}$, but the hidden size for each direction is set to

$\frac{1}{2}d$. Claims and evidence are processed by different recurrent layers (i.e. there is no weight sharing). However, we do not use the SELECT-LAST function, and instead will operate on the entire vector of hidden states.

Attention

A similarity matrix $S \in \mathbb{R}^{I \times J}$ is constructed, where S_{ij} is the similarity between the representation of claim word i and evidence word j. For clarity, C_i and E_j refer to the i^{th} and j^{th} contextually encoded word embeddings for claim word i and evidence word j.

$$S_{ij} = f_{(s)}(C_i, E_j) \in \mathbb{R}^1.$$

This similarity is calculated by a learned affine function $f_{(s)}$.

$$f_{(s)} : \mathbb{R}^{d_{(intra)}} \times \mathbb{R}^{d_{(intra)}} \to \mathbb{R}, \qquad f_{(s)}(u, v) = w_{(s)}^{\intercal}[u; v; u \circ v] + b_{(s)}.$$

Next, we construct attended representations with attention flowing from evidence to claim and claim to evidence. This was an important idea from [46]. However, our representation is different because our task has a simpler output (a score), and thus only requires a summarization of the context.

$$\alpha_{(c)} = \text{softmax}(\max_{row}(S)) \in \mathbb{R}^J$$
$$\grave{e} = \sum_j \alpha_{(c)j} C_j \in \mathbb{R}^{d_{(intra)}}$$

The same is computed for the claim.

$$\alpha_{(e)} = \text{softmax}(\max_{col}(S)) \in \mathbb{R}^I$$

$$\grave{c} = \sum_i \alpha_{(e)i} E_i \in \mathbb{R}^{d_{(intra)}}$$

Modeling

We model the attended vectors with concatenation. There are a lot of other options here, but simple concatenation was effective in [46]. The authors of [46] also included the previous unattended representations, but we opt for a more simplified approach. There is room in future work to explore this decision further.

$$v = [\grave{e}; \grave{c}; \grave{e} \circ \grave{c}] \in \mathbb{R}^{3*d_{(intra)}} \tag{5.8}$$

Output

The final output layer $f_{(o)}$ introduces a wealth of possible formulations. We use a simple affine function.

$$f_{(o)} : \mathbb{R}^{3d_{(intra)}} \rightarrow \mathbb{R}^1, \qquad f_{(o)}(v) = w_{(o)}^\intercal v + b_{(o)}$$

Another reasonable option would be a dot product with a learnable weight vector without a bias. If we find that the output should have a restricted domain a reasonable choice would be tanh(.).

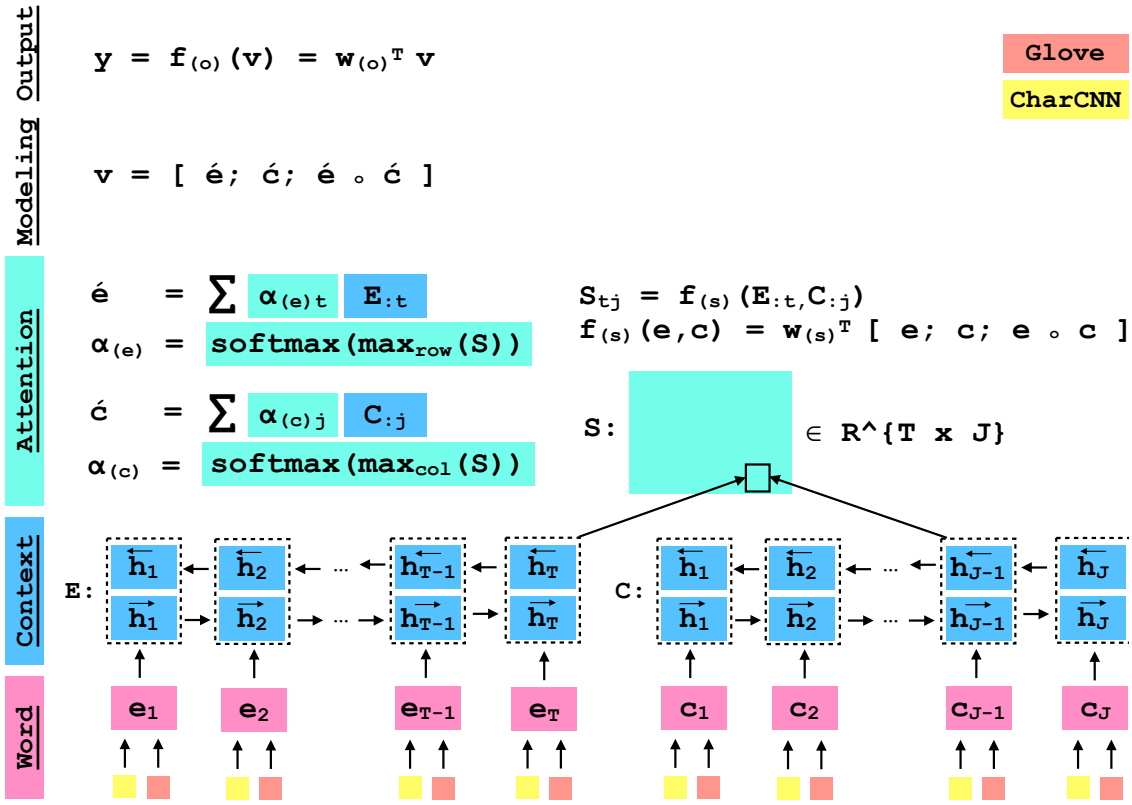

Figure 5.3: Attention that flows from both evidence to claim and claim to evidence. Best viewed in color.

5.4 Document-based Model Formulation

This section covers the goal, loss, and model definition for a model that processes an entire document and a claim. The goal of our model is to determine a total ordering among all sentences in a document to a given claim. Any discrepancy between labels of sentences is an error, and not indicative of a partial ordering. We model this in our model by outputting a single score in scalar space for each sentence.

Specifically, we are given a document D consisting of L evidence sentences e, each of varying number of words T_l from $l = 1 \ldots |L|$. A subset of sentences in this document are compared in a binary fashion, where the sentence that contained more evidence was labeled as such. This is not a complete set of combinations.

$$D = \{e_l\}_{l=1}^{L} \mid T_l = |e_l|$$

We have a set of annotations A_D^c (eq. 5.9) with respect to claim c for a given document D. Each annotation consists of a task τ and a label y.

$$A_D^c = \{(\tau_i, y_i)\}_{i=1}^{N} \mid \tau_i = \{e_i^1, e_i^2\} \tag{5.9}$$

For a task τ, e^1 and e^2 are candidate evidence sentences that are to be directly compared. Each task τ_i is labeled by annotators as one of `0` or `1` by y_i. When y_i is `0` it indicates e_i^1 is more relevant, and e_i^2 otherwise. This is the same as in the pairwise formulation §5.1

We develop a model which encodes sentences and then ranks the relative importance of all sentences. The encoding is done hierarchically, first at the word level and then the sentence level. This approach is similar to [58], however we have an ordinal ranking head which outputs a value for each sentence. We detail the model in §5.4.2.

The best training regime for this model is not immediately clear. There are two primary complications. First, the training labels within a document are sparse. Second, even for the labels we do have, they are binary comparisons and as such the loss cannot be immediately evaluated when the score is regressed. As described in below in §5.4.1, we process the entire document and incur a loss only for annotated sentences in the document.

5.4.1 Loss function

The objective for our model ψ is to output a real value that corresponds to the relative rank of the item for each of the L candidate evidence sentences in a document.

$$\psi : D \times c \times \theta \rightarrow \mathbb{R}^L$$

The parameters of the model θ are learned and are dependent on the architecture of the model ψ. We elide θ from further notations for simplicity. The loss for each model ψ described below is as follows for an annotated corpus $C = \{A^c_{Di}, D_i\}_{i=1}^N$ of N sets of annotations A with respect to a claim c and document D.

$$L : C \rightarrow \mathbb{R}, \qquad L(C) = \frac{1}{N} \sum_{A^c_D, D \in C} L_D(A^c_D, \psi(D, c)) \tag{5.10}$$

$$L_D : A^c \times \hat{Y} \rightarrow \mathbb{R}, \qquad L_D(A^c, \hat{Y}) = \frac{1}{|\tau, y \in A^c_D|} \sum_{\tau, y \in A^c_D} L_\tau(\tau, y, \hat{Y}) \tag{5.11}$$

$$L_\tau : \tau \times y \times \hat{Y} \rightarrow \mathbb{R}, \qquad L_\tau(\tau, y, \hat{Y}) = \mathcal{H}^*(\hat{Y}_{e^1}, \hat{Y}_{e^2}, y) \tag{5.12}$$

The loss is averaged across the corpus as shown in Eq. 5.10. For each set of annotations, the model ϕ computes the scores for all the sentences in the document. Then the loss is averaged across all the tasks in the set of annotations in Eq. 5.11. From here, L_τ is the very similar to the loss used pairwise model (see Eq. (5.2). Specifically, as shown in Eq 5.12, it instead indexes the results from the vector of scores for each sentence $\hat{Y}$ computed by ϕ. The hinge loss $\mathcal{H}^*$ is defined above in Eq. 5.2.

5.4.2 Hierarchical Attention for Document-Wise Rankings

The document models are built on top of our pairwise approach, and evaluated in a pairwise manner. Note, that this model builds on top of the pairwise formulation defined in §5.3. This formulation is similar to that describe by [58].

Sentence Representation

The sentences are represented as in §5.3. Note that the evidence dimension now includes the number of sentences in the cited document L. In the following equations, I is the number of words in the claim sentence and J is the maximum number of words in a candidate evidence sentence.

$$c \in \mathbb{R}^{I \times d_{(word)}}$$

$$e \in \mathbb{R}^{L \times J \times d_{(word)}}$$

After encoding the words, we encode the intra sentence context using bidirectional recurrent networks. The difference between this and the approach in §5.3 is that the claim is first repeated (tiled) for each of the sentences in the document.

$$C = \text{BI-RNN}(c) \in \mathbb{R}^{I \times d_{(intra)}}$$

$$C_{tiled} = [C | x \in 1 \dots L] \in \mathbb{R}^{L \times I \times d_{(intra)}}$$

$$E = \text{BI-RNN}(e) \in \mathbb{R}^{L \times J \times d_{(intra)}}$$

As above, we apply **intra-sentence attention** with the bidirectional attention between the words within each candidate evidence sentence and the claim. The resulting representations are summarized across the number of word dimensions.

$$\grave{e} = \text{BIDIRECTIONAL-ATTENTION}(E, C_{tiled}) \in \mathbb{R}^{L \times d_{(intra)}}$$

$$\grave{c} = \text{BIDIRECTIONAL-ATTENTION}(C_{tiled}, E) \in \mathbb{R}^{L \times d_{(intra)}}$$

We then model the sentences representations in the document with our standard approach of concatenating the tensors and their element-wise product.

$$v = [\grave{e}; \grave{e} \circ \grave{c}; \grave{c}] \in \mathbb{R}^{L \times 3d_{(intra)}}$$

Inter-Sentence Contextual Embedding

We follow the same pattern as in the pairwise formulation; the context between the sentences is encoded using a recurrent network. This works in the same way as the contextual embedding between words. The output dimensionality of the inter sentence contextual embedding for each sentence is d_{inter}.

$$V = \text{BI-RNN}(v) \in \mathbb{R}^{L \times d_{(inter)}}$$

Inter-Sentence Attention

The attention at the sentence level is based off a learned representation. We use a formulation of attention, as in [58], where c is a learned vector. We do not reuse a representation of the claim again here because the network has already encoded the

claim into its representation. This learned vector enables the attention mechanism to determine if a given sentence successfully provides evidence for the claim it incorporates.

$$
\begin{aligned}
U_{(a)} &= \tanh(W_{(a)}^{\intercal} V + b_{(a)}) \in \mathbb{R}^{L \times d_{(inter)}} \\
\alpha_{(a)} &= \text{softmax}(U_{(a)}^{\intercal} c) \in \mathbb{R}^{L \times d_{(inter)}} \\
\grave{v} &= V \circ \alpha_{(a)} \in \mathbb{R}^{L \times d_{(inter)}}
\end{aligned}
$$

Modeling

The final layer is a learnable affine layer.

$$
y = f_{(o)}(\grave{v}) = W_{(o)}^{\intercal} \grave{v} + b_{(o)} \in \mathbb{R}^{L}
$$

Model Definition

The complete model architecture, with high-level operations for the different modules, is shown below.

$$c \in \mathbb{R}^{I \times d_{(word)}}$$

$$e \in \mathbb{R}^{L \times J \times d_{(word)}}$$

$$C = \text{INTRA-SENTENCE-EMBEDDING}(c) \in \mathbb{R}^{I \times d_{(intra)}}$$

$$C_{tiled} = [C | x \in 1 \dots L] \in \mathbb{R}^{L \times I \times d_{(intra)}}$$

$$E = \text{INTRA-SENTENCE-EMBEDDING}(e) \in \mathbb{R}^{L \times J \times d_{(intra)}}$$

$$\grave{e} = \text{INTRA-SENTENCE-ATTENTION}(E, C_{tiled}) \in \mathbb{R}^{L \times d_{(intra)}}$$

$$\grave{c} = \text{INTRA-SENTENCE-ATTENTION}(C_{tiled}, E) \in \mathbb{R}^{L \times d_{(intra)}}$$

$$v = [\grave{e}; \grave{e} \circ \grave{c}; \grave{c}] \in \mathbb{R}^{L \times 3d_{(intra)}}$$

$$V = \text{INTER-SENTENCE-EMBEDDING}(v) \in \mathbb{R}^{L \times d_{(inter)}}$$

$$\grave{v} = \text{INTER-SENTENCE-ATTENTION}(V) \in \mathbb{R}^{L \times d_{(inter)}}$$

$$y = f_{(o)}(\grave{v}) = W_{(o)}^{\intercal} \grave{v} + b_{(o)} \in \mathbb{R}^{L}$$

Chapter 6

Prototype Application

We will create a web application that allows users to quickly find evidence in supporting documents for claims they highlight. The web application will make requests to a server that will preform the necessary evidence recommendation and ranking. Users will be able to interact with the website to fully leverage our proposed capabilities.

The proposed prototype will avoid working directly with PDFs. We will only display results upon a text version of an input document. Our application will convert their document into text.

The user will then select a referencing document and a referenced document. The user will then be able to select a sentence from the referencing document. After processing the referenced document, the user will have a few different options for viewing the related evidence. One view will list the top k strongest pieces of evidence sentences, where k is chosen by the user. Another view will highlight the sentences highlighted directly in a text version of their document.

Screenshots can be seen in the Appendix A.2.

Chapter 7

Evaluation

We report the pairwise accuracy, the percentage of correctly classified pairs. Each pair is classified as correct if the score for the appropriate piece of evidence is higher. Given that the dataset is based on comparisons, we are able to balance it perfectly. This prevents skew of classes and allows us to interpret our results more easily.

When partitioning the dataset into train, test and validation splits we ensured that there was no data leakage. To do this, we did not allow and documents (citing or cited) to be present in more than one partition. We attempted to split the data as follows: 15% of the data purely for testing (exactly once), 15% for validation testing, and 70% for training. However, our restriction on the stratification leads these percentages to only be approximate. These exact percentages are reported in Table 4.3.

7.1 Results

We present the results for different abalations of our pairwise model below in Table 7.1. The models here were all trained on the same hyperparameters; they were not specifically optimized for a specific model or abalation.

Table 7.1: Performance on Citation Dataset

	rnn	bidirectional	attention
majority	0.56875	0.6125	0.544
high majority	0.773	0.600	0.640
unanimous	0.677	0.754	0.646
augmented	0.625	0.625	0.725

Table 7.2: Performance on Citation Dataset with Document Model

base	inter-sentence attention
0.748	0.5971

The performance on the document-wise models are listed below in Table 7.2. We do not show the results for the models with word level attention. The basic abalation does not use intra-sentence nor inter-sentence attention. The inter-sentence attention abalation only applies attention between sentences. We can see that the interactions and context between sentence gives the model more power for the base implementation. However, the attention mechanism was not effective and likely overfit. Increasing data and regularization could help with this issue.

We also evaluate these same abalations, with no further hyperparamater optimization, on a modified version of the argument dataset found in [27]. This isomorphic dataset was developed for a similar purpose and its format allows us to evaluate our approach in a different domain. We present these results in Table 7.3. The size of this dataset is 32,000 training instances and 32,000 test instances. We see similar performance on this dataset.

Table 7.3: Performance on Argument Dataset

rnn	bidirectional	attention
0.615	0.615	0.650

Chapter 8

Future Work

As we investigated this new problem, we noted several promising areas for future work. We organize these ideas (below) by theme.

8.1 Next Steps

A direct next step is to increase the dataset size. Our dataset is complicated and dense; it provides a difficult learning environment excellent for testing and exploring new methods. Further investment (time and money) would allow for more experimentation in this rich cross section of natural language understanding and ranking coupled with a highly utilitarian application. This would also allow for more meaningful hyperparameter optimization as the models would be less likely to overfit. As our results show, our model was more effective on the argument dataset which was larger. This is likely not the only difference. From reading samples from both datasets, the language in the academic documents is much more verbose and the connections less clear. It seems like this dataset is inherently more difficult. However, increasing the dataset size would be one way to confirm this. Other small additions, like using richer word representation like character encodings [30] or ConceptNet

could directly improve performance.

8.2 Alternative Approaches

From our results, it seems our formulation of the problem was successful. However, it is possible other approaches could be more effective and it would be interesting to further investigate this. Specifically, in our approach we train directly from pairwise annotation. Experimenting with generating a ranking from binary labels as a preprocessing step as demonstrated in [55] may prove effective. Also, LambdaMART [11] is an listwise ranking approach built on an ensemble of trees that estimate the gradients. Directly modeling the natural language understanding and incorporating it into this approach is promising.

8.3 Expert Evaluation

Our models do not perform with an extremely high accuracy. However, it would be valuable to develop baselines of human performance. This practice is not uncommon [43]. It would be interesting to compare the performance of a psychology major, graduate student and professor to see the performance level of the model.

8.4 User Study

We already have developed a user interface as part of this project. Using this interface in user studies would illuminate and ground the impact of this project. In particular, the results the model returns, while not perfect, may already prove to be *good enough*. If the model is able to highlight relevant text the reader may be able to quickly understand the material and read surrounding context, even if this is not

the *most* relevant text in the document. As user study on this project could focus both on whether or not the recommended sentences provide evidence to the claim as understood by the reader, and if the prototype is effective in helping the users quickly understand the referenced document.

Chapter 9

Conclusion

In this work we explore Evidence Recommendation, a new machine learning task. We build a new dataset using crowdsourcing methods and annotate our dataset with a pairwise comparison paradigm. We demonstrate some success with deep learning models achieving an accuracy of 77.7%. However, our proposed additional features and architectures did not provide a statistically significant improvement over our base model. We hypothesize increasing the size of the dataset will help better leverage these mechanisms. We also develop a prototype application, providing a foundation for future work.

Appendix A

Visuals

A.1 Task Screen Shots

This section consists of screen shots of tasks presented to the Mechanical Turk workers.

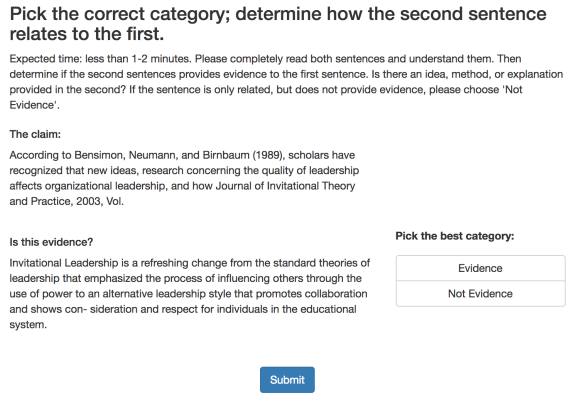

Figure A.1: The annotation task (BC-E) for Mechanical Turk workers to classify a candidate sentence into either `evidence` or `not evidence`.

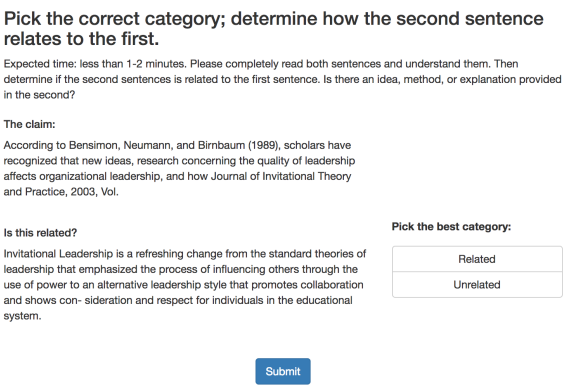

Figure A.2: The annotation task (BC-R) for Mechanical Turk workers to classify a candidate sentence into either `relevant` or `not relevant`.

A.2 Interface Screen Shots

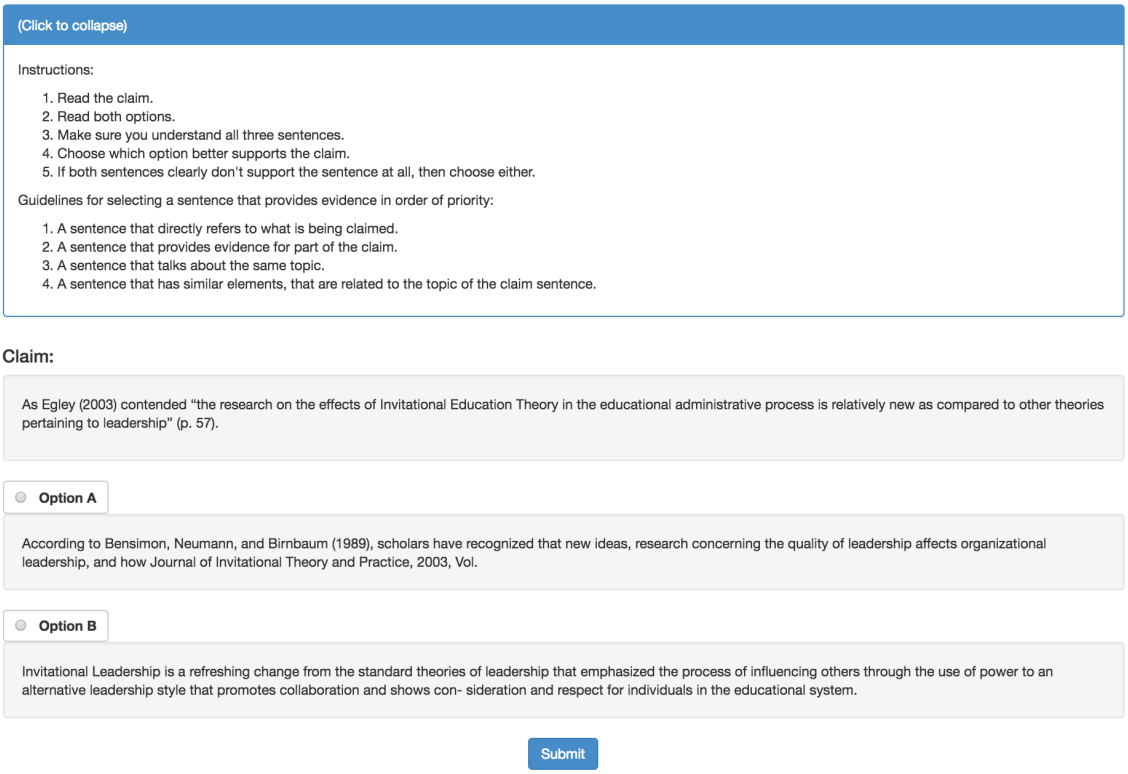

Figure A.3: The annotation task for Mechanical Turk workers to compare candidate sentences for a ranking paradigm.

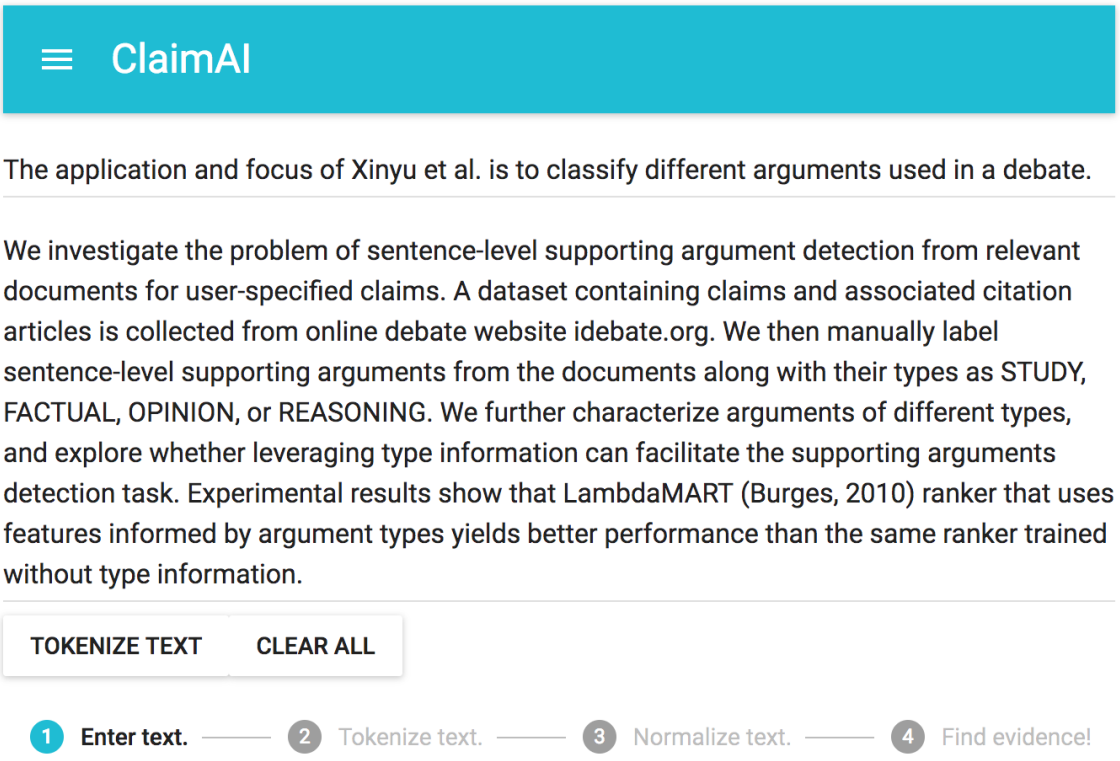

Figure A.4: Text added prototype.

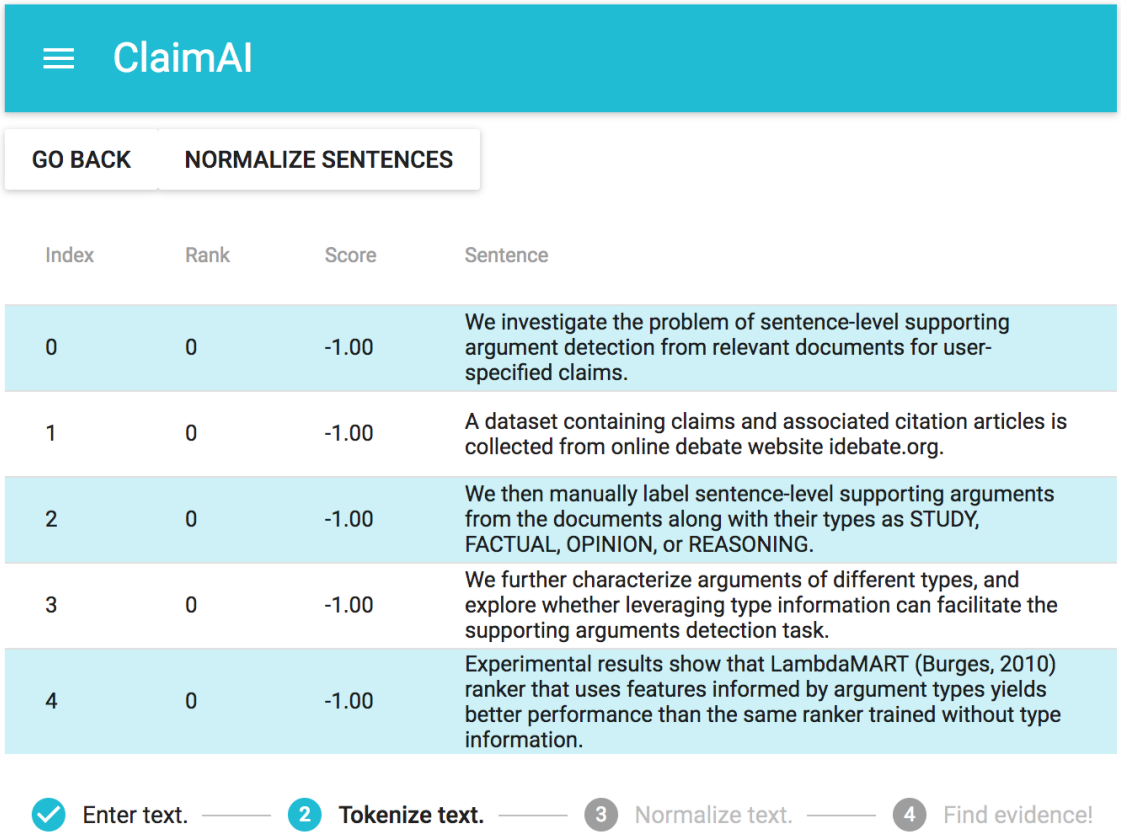

Figure A.5: Text is tokenized.

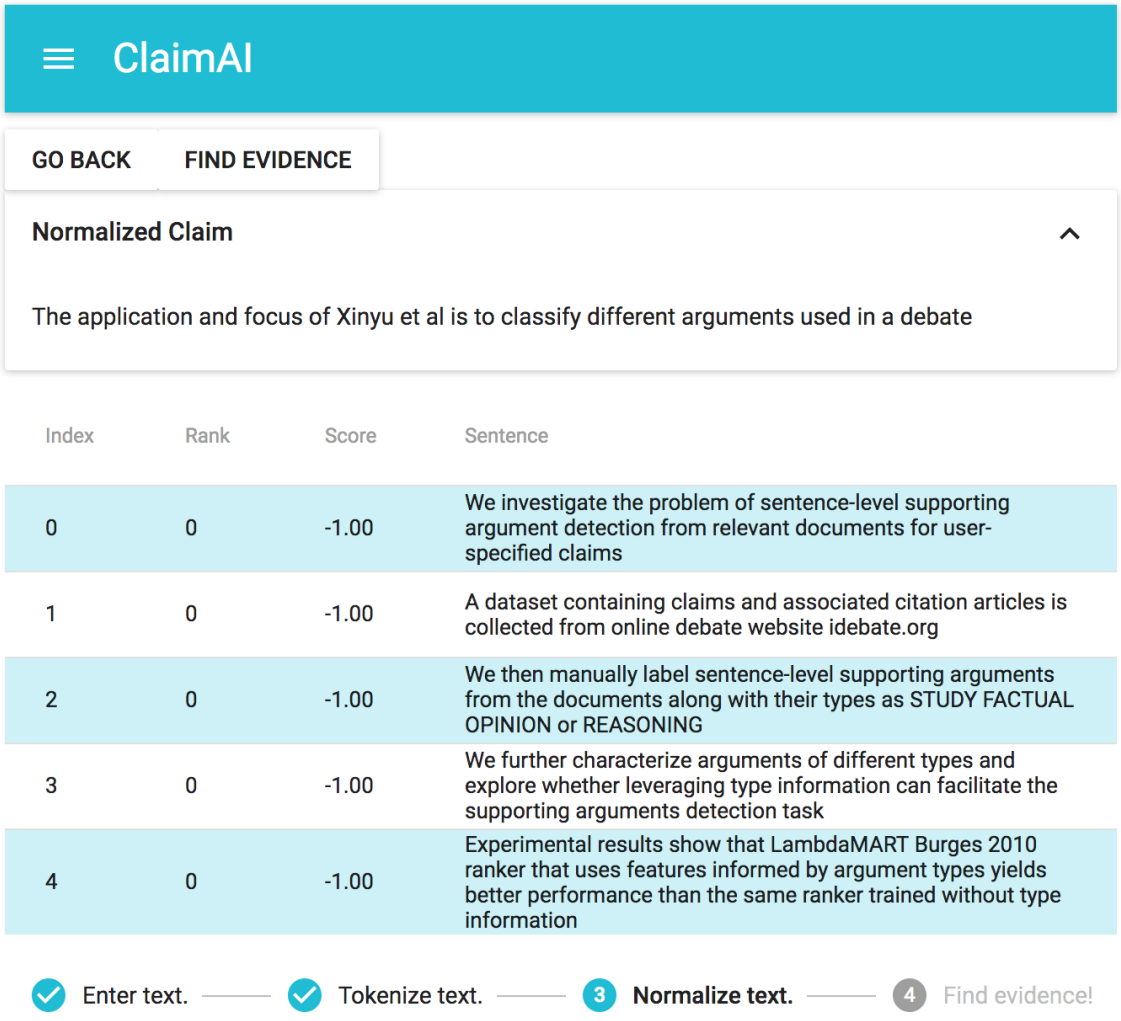

Figure A.6: Text is normalized.

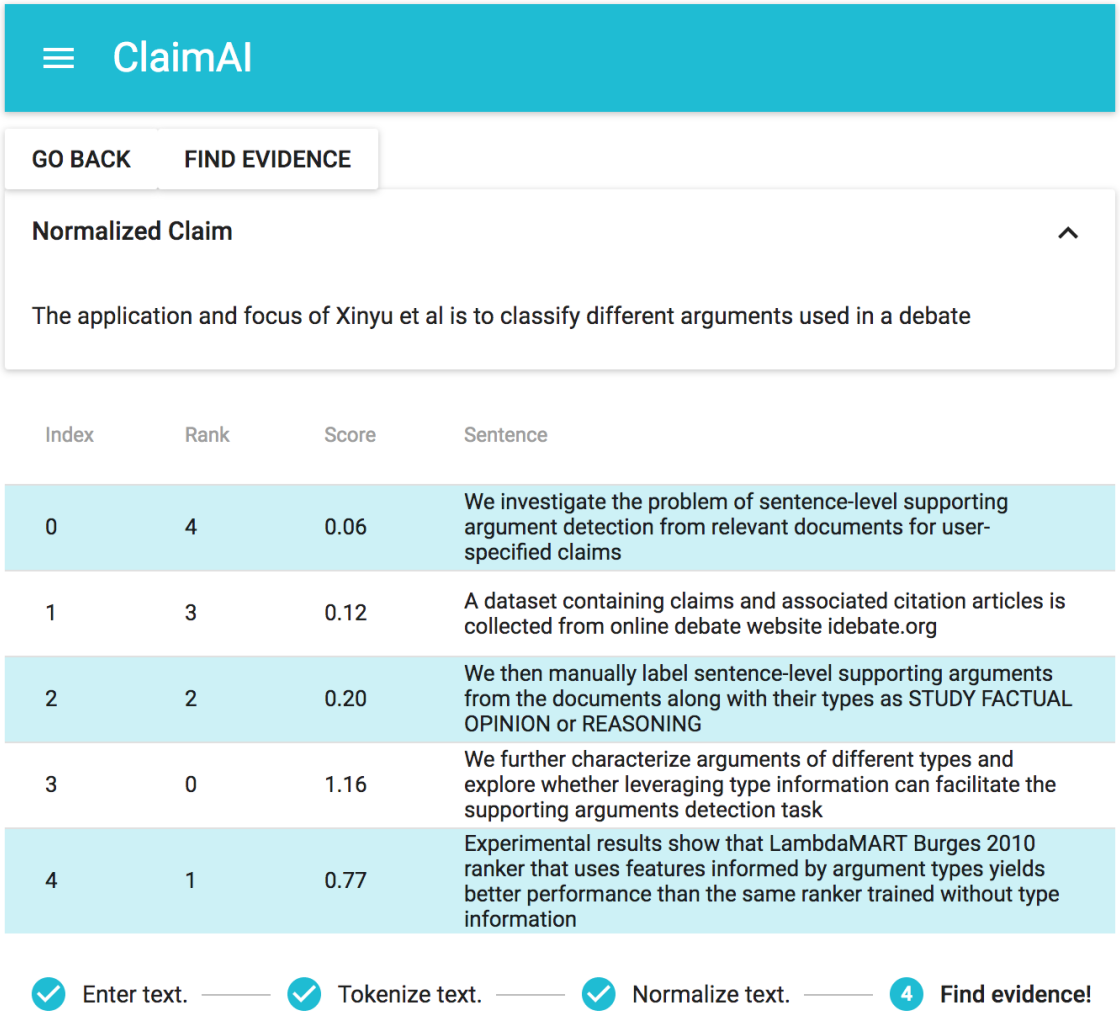

Figure A.7: Evidence is found. In this image, the model was not trained.

Bibliography

[1] Amazon mechanical turk. https://www.mturk.com/mturk/welcome.

[2] Parsecit. http://aye.comp.nus.edu.sg/parsCit/. Currently website is down.

[3] pdfminer. https://euske.github.io/pdfminer.

[4] American education research association. http://www.aera.net/Publications/Online-Paper-Repository/AERA-Online-Paper-Repository, 2017.

[5] S. Alzahrani, V. Palade, N. Salim, and A. Abraham. Using structural information and citation evidence to detect significant plagiarism cases in scientific publications. *Journal of the Association for Information Science and Technology*, 63(2):286–312, 2012.

[6] S. Aya, C. Lagoze, T. Joachims, et al. Citation classification and its applications. In *Proceedings of the International Conference on Knowledge Management*, pages 287–298, 2005.

[7] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[8] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[9] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[10] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

[11] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[12] C. J. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200, 2007.

[13] W. Cao, A. Song, and J. Hu. Stacked residual recurrent neural network with word weight for text classification. *IAENG International Journal of Computer Science*, 44(3), 2017.

[14] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

[15] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen. Recurrent neural network-based sentence encoder with gated attention for natural language inference. *arXiv preprint arXiv:1708.01353*, 2017.

[16] W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma, and H. Li. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems*, pages 315–323, 2009.

[17] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[18] J. Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[19] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[20] W. B. Frakes and R. Baeza-Yates. *Information retrieval: data structures and algorithms*. Prentice Hall PTR, 1992.

[21] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

[22] C. L. Giles, K. D. Bollacker, and S. Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98. ACM, 1998.

[23] Q. He, J. Pei, D. Kifer, P. Mitra, and L. Giles. Context-aware citation recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 421–430. ACM, 2010.

[24] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *MIT Press*, 2000.

[25] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] M. Hu, Y. Peng, and X. Qiu. Reinforced mnemonic reader for machine comprehension. *CoRR, abs/1705.02798*, 2017.

[27] X. Hua and L. Wang. Understanding and detecting supporting arguments of diverse types. *arXiv preprint arXiv:1705.00045*, 2017.

[28] S. Jagabathula and D. Shah. Inferring rankings under constrained sensing. In *Advances in Neural Information Processing Systems*, pages 753–760, 2009.

[29] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):15, 2012.

[30] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[31] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.

[32] O. Küçüktunç, E. Saule, K. Kaya, and Ü. V. Çatalyürek. Diversifying citation recommendations. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):55, 2015.

[33] T.-M. Kuo, C.-P. Lee, and C.-J. Lin. Large-scale kernel ranksvm. In *Proceedings of the 2014 SIAM international conference on data mining*, pages 812–820. SIAM, 2014.

[34] C.-P. Lee and C.-J. Lin. Large-scale linear ranksvm. *Neural computation*, 26(4):781–817, 2014.

[35] L. Leydesdorff and S. Milojević. Scientometrics. *arXiv preprint arXiv:1208.4566*, 2012.

[36] X. Liu, Y. Shen, K. Duh, and J. Gao. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*, 2017.

[37] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[38] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[39] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[40] N. Nangia, A. Williams, A. Lazaridou, and S. R. Bowman. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. *arXiv preprint arXiv:1707.08172*, 2017.

[41] Y. Nie and M. Bansal. Shortcut-stacked sentence encoders for multi-domain inference. *arXiv preprint arXiv:1708.02312*, 2017.

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

[43] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[44] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.

[45] A. Ritchie. Citation context analysis for information retrieval. Technical report, University of Cambridge, Computer Laboratory, 2009.

[46] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[47] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM, 2015.

[48] T. Shen, T. Zhou, G. Long, J. Jiang, S. Wang, and C. Zhang. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. *arXiv preprint arXiv:1801.10296*, 2018.

[49] R. Speer and J. Lowry-Duda. Conceptnet at semeval-2017 task 2: Extending word embeddings with multilingual relational knowledge. *arXiv preprint arXiv:1704.03560*, 2017.

[50] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[51] Y. Tay, L. A. Tuan, and S. C. Hui. A compare-propagate architecture with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*, 2017.

[52] W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198, 2017.

[53] Y. Wang, L. Wang, Y. Li, D. He, and T.-Y. Liu. A theoretical analysis of ndcg type ranking measures. In *Conference on Learning Theory*, pages 25–54, 2013.

[54] J. I. Wattenberg M., Viegas F. How to use t-sne effectively. `https://distill.pub/2016/misread-tsne/`, 2017.

[55] F. Wauthier, M. Jordan, and N. Jojic. Efficient ranking from pairwise comparisons. In *International Conference on Machine Learning*, pages 109–117, 2013.

[56] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pages 2035–2043, 2009.

[57] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM, 2008.

[58] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489, 2016.