TOKN Phase I Report

Matthew Boggess
Stanford University

Han Lin Aung
Stanford University

Dr. Vinay Chaudhri
Stanford University

Abstract

TOKN is a joint research project between Openstax, Stanford, and Rice University to design open source knowledge graphs (KG) across multiple academic subjects to support educational efforts and tools such as "intelligent" textbooks. Here we present work completed by Stanford for phase I on automatic knowledge extraction. Specifically, we present a pipeline for automatically generating triples of the KG by first extracting key terms from textbooks (nodes of the KG) and then extracting relations between these terms (edges of the KG).

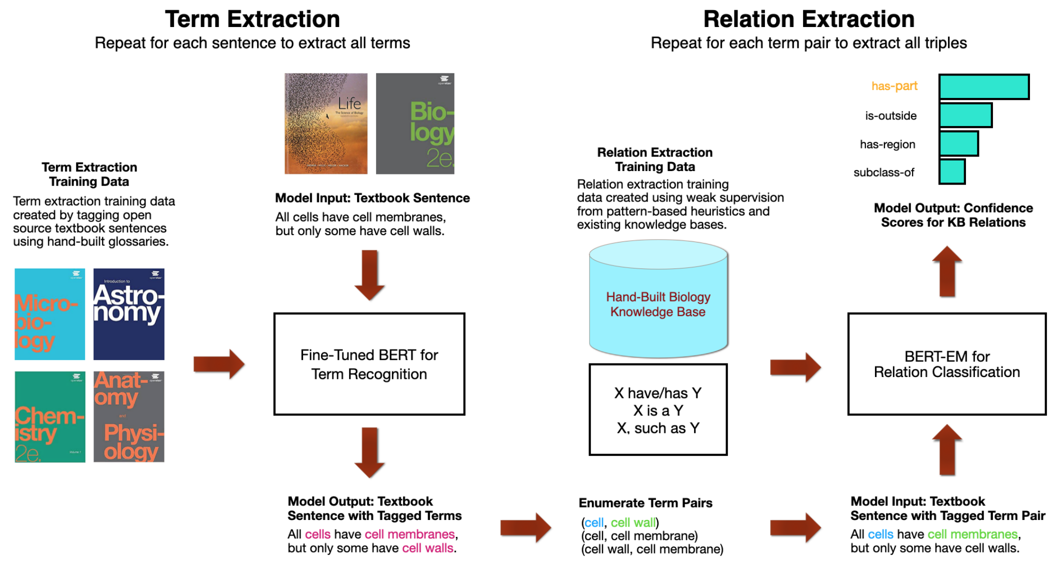
Term Extraction
Repeat for each sentence to extract all terms
Term Extraction Training Data
Term extraction training data created by tagging open source textbook sentences using hand-built glossaries.
Model Input: Textbook Sentence
All cells have cell membranes, but only some have cell walls.
Fine-Tuned BERT for Term Recognition
Model Output: Textbook Sentence with Tagged Terms
All cells have cell membranes, but only some have cell walls.
Relation Extraction
Repeat for each term pair to extract all triples
Relation Extraction Training Data
Relation extraction training data created using weak supervision from pattern-based heuristics and existing knowledge bases.
Hand-Built Biology Knowledge Base
X have/has Y
X is a Y
X, such as Y
Enumerate Term Pairs
(cell, cell wall)
(cell, cell membrane)
(cell wall, cell membrane)
Model Input: Textbook Sentence with Tagged Term Pair
All cells have cell membranes, but only some have cell walls.
BERT-EM for Relation Classification
Model Output: Confidence Scores for KB Relations
has-part
is-outside
has-region
subclass-of

Figure 1: Overview of Automated Knowledge Extraction Pipeline.

1 Overview of Approach

In a knowledge graph (KG) for an academic textbook, the nodes of the graph represent important concepts or terms for that subjects. Examples in biology would include cell, mitosis, etc. The edges represent relations that hold between pairs of terms. For example, a cell has-part cell wall. An important step in constructing a KG for an academic textbook thus consists of identifying all triples (pair of terms and a relation between them) encoded in the text. In this work, we present first steps towards an automatic pipeline for extracting these triples from textbooks. Figure 1 provides an overview of our pipeline.

The first task is to identify all of the terms in the text which will form the nodes of our KG. We call this term extraction. We cast this problem onto the well-studied NLP task of Named Enity Recognition (NER) where we train a model to tag all terms in a textbook sentence. Aggregating all of the terms tagged across all of the sentences in a textbook gives us a final list of terms for that textbook that will form the nodes in its KG.

The second task is to identify all of the relations that hold between every term pair from our term list that will form the edges in our graph. We call this relation extraction. We cast this problem onto the well-studied NLP task of sentence classification where we train a model to classify between a pre-set list of relation types given a sentence with a tagged term pair. Again, aggregating all of these relations across all sentence and term pair groupings in the textbook gives us a final list of relations for that textbook that will form the edges in its KG.

2 Data Sources

In order to train models to automatically extract these triples, we need labelled training data. To the best of our knowledge, there is no prior existing dataset for extracting structured knowledge from textbooks so here we present data that we extracted for these tasks. This data consists of a set of sentences and terms extracted from 10 academic textbooks across multiple science subjects listed in Table 1. All of these textbooks are open source, freely available on OpenStax [3] with the exception of Life Biology which we used do to prior work on constructing a knowledge base.

Sentences from each of the chapters were first parsed and cleaned for formatting errors. These sentences serve as the inputs to our models so that we can extract terms and relations using linguistic cues found in textbook language. The terms were extracted from the glossaries and indices of the

Table 1: **Textbook Dataset**

Textbook	# Sentences	# Terms
OpenStax Anatomy & Physiology	21706	3196
OpenStax Astronomy	18844	810
OpenStax Biology 2e	24544	2757
OpenStax Chemistry 2e	13799	954
Life Biology	16673	0
OpenStax Microbiology	16190	4149
OpenStax Psychology	9967	1086
OpenStax Physics Volume I	15005	462
OpenStax Physics Volume II	11779	466
OpenStax Physics Volume III	9250	580

Table 2: **Bio101 KB Relations**

Domain	Range	Family	Relation	Term Pairs
Entity	Entity	Taxonomy	subclass-of	18360
Entity	Entity	Taxonomy	synonym	12308
Entity	Entity	Meronym	has-part	6542
Event	Entity	Participant	result	3648
Event	Entity	Participant	raw-material	3526
Event	Entity	Participant	object	3021
Entity	Entity	Meronym	has-region	2976
Event	Event	Event Structure	subevent	2894
Event	Entity	Participant	base	2396
Event	Entity	Participant	agent	1807
Entity	Entity	Meronym	possesses	1744
Entity	Entity	Spatial	is-inside	1048
Entity	Entity	Spatial	encloses	1022
Event	Event	Event Structure	next-event	895
Event	Event	Event Structure	first-subevent	539
Entity	Entity	Meronym	element	533
Event	Event	Donor/Recipient	donor	470
Entity	Entity	Spatial	is-between	461
Event	Event	Donor/Recipient	recipient	383
Event	Entity	Participant	instrument	333
Entity	Entity	Spatial	is-at	307
Entity	Entity	Spatial	does-not-enclose	219
Entity	Entity	Spatial	is-outside	196
Entity	Entity	Spatial	abuts	162

textbooks and will help us create labels for term extraction and term pair inputs for relation extraction, which we describe in more detail in later sections.

Additionally, we have access to a manually curated knowledge base, Bio101 KB, that was constructed by subject matter experts (SMEs) on the first 10 chapters of Life Biology as part of the Inquire project [2]. This KB provides an additional list of 8136 biology terms, but even more importantly provides sets of term pairs with particular relations that hold between them (see table 2 for counts by relation type). Thus this KB will be an important source for providing labels in relation extraction as it is our main source of labelled relations between term pairs.

Details on exactly how these data sources were used to create training data for each task are covered in the latter sections of this report.

3 Term Extraction

3.1 Related Work

Extracting terms from texts is an important problem in natural language processing known as Automatic Term Extraction (ATE). ATE originated with researchers manually creating rules to identify terms based on hand-crafted linguistic and statistical features [17, 24]. These workflows later evolved to incorporate machine learning techniques with algorithms like decision trees naturally replacing these manually constructed rules [6, 8]. To the best of our knowledge, ATE is not a well-studied problem in the deep learning literature. Some recent work has investigated using word embeddings to support ontological queries in the biomedical domain [5, 4]. [19] used recurrent neural networks to directly translate sentences into formal ontology representations, but this was limited to fairly simple sentence structures.

More work has been done with deep learning models in a closely related task to ATE known as Named Entity Recognition (NER). NER involves tagging token spans in sentences corresponding to special types of entities such as names or locations (for a recent review of deep learning approaches to NER see [14]). Following recent trends in deep learning in NLP, perormance on standard NER tasks has been dominated by transfer learning where pre-trained language models are fine-tuned for the NER task. One such example is the BERT language model which provides recommendations for fine-tuning it for NER [7].

3.2 Problem Formulation

Term extraction involves extracting import subject-specific concepts or terms from the text that will form the nodes of our KG. While there are multiple ways to frame this problem, we chose to leverage existing NLP models and research by casting this term extraction problem onto the well-characterized NLP problem of named entity recognition (NER). In NER, a model is provided a sequence of text and must label every token in the text as either being part of some special entity or just background text (for example, identifying persons in text). In our case, we only have a single entity type, words are either part of a term or not a term. We use the BIOES tagging scheme [20] for labeling what are terms as follows:

- B = beginning of term phrase
- I = interior of term phrase
- O = not part of term
- E = end of term phrase
- S = single token term

As an example input and output:

- Input: ['All', 'cells', 'have', 'a', 'cell' 'membrane', '.']
- Output: ['O', 'S', 'O', 'O', 'B' 'E', 'O']

The predicted terms from this sentence would then be 'cells' and 'cell membrane'. Thus the goal is to have a model that takes in chapter sentences from academic textbooks and produces a set of BIOES tags for that sentence denoting where the terms are. We can then aggregate the labeled terms across multiple sentences to build up a list of terms that form the full set of nodes for the KG.

3.3 Dataset

We constructed a novel dataset for this task using the textbook sentences and terms presented in Table 1. In order to train a model on this task, we need a set of labelled sentences where every sentence has a corresponding set of BIOES tags denoting every term occurrence in each sentence. Given we did not have a hand-labelled dataset, we make use of the terms lists we extracted from the glossaries/indices of the textbook and use these to automatically tag and label the sentences. In order to be robust to simple lexical modifications such as pluralization, capitalization, etc., we used Spacy to preprocess the sentences and terms and matched on lemmatized forms with some additional handling for special cases such as acronyms, hyphenation, etc.

Table 3: Term Extraction Data Splits

Split	Sources	Sentences	Terms
Train	All textbooks in table 1 except OpenStax Biology and Life Biology	58057	7041
Dev	OpenStax Biology Ch. 4 Sect. 2, Ch. 10 Sect. 2 & 4	206	283
Test	Life Biology Ch. 39	608	500

Glossaries/indices are highly accurate but incomplete sources of terms for creating a comprehensive KG. Thus there is a high amount of noise in the training data as many valid terms are not correctly labelled and instead are labelled negatively. This is a tradeoff as manually labelling all of the sentences requires a large time investment by SMEs.

Eight of the textbooks presented in Table 1 were pooled together into a training set (all textbooks except the OpenStax Biology 2e and Life Biology textbooks) and labelled using all of their terms pooled together. To evaluate our models, we had biologist SMEs hand label a dev set consisting of three sections from OpenStax Biology 2e (Chapter 4 section 2, Chapter 10 sections 2 and 4) and hand label a test set consisting of Chapter 39 from Life Biology. Finally, we removed all sentences from the training set that contained any of the terms from the dev and test set. The sentence and term counts for the resulting splits are shown in Table 3. These splits were designed to pool as much data as possible into the training split and then see how well the term extractor can generalize to new terms from a new textbook on a subject not in the training set.

3.4 Model

3.4.1 Model Architecture

The core of our model is a pre-trained language model released by Google known as BERT [7]. We use the smaller BERT base model. The BERT model consists of an encoder consisting of 12 transformer attention hidden layers [26]. This encoder is pre-trained on a language modeling task that involves predicting masked words in sentences using the remaining non-masked words. The inputs to the encoder are a combination of WordPiece embeddings [29] as well as position indicators for each word. For more details on BERT, see the original paper. This pre-trained encoder and embeddings can then be fine-tuned for different classification tasks. The advantage of fine-tuning a large pre-trained language model such as BERT is that the language model representations create a very rich prior starting point that has encoded a lot of useful information about the language structure that can then be fine tuned for a particular task with much less data than might otherwise be required. In the case of NER, fine-tuning involves adding a fully connected softmax output layer on the final hidden layer of the BERT encoder that produces a probability distribution across the five BIOES tags for each token. The predicted tag for that token is then just the tag with the highest probability and the tag probabilities can be used to form confidence scores for the different predictions. One major caveat with this implementation is that the WordPiece embeddings often break up tokens into multiple "pieces", breaking the one-to-one relationship between the encoder output layer and the original tokens in the sentence. We follow the recommendation in the BERT paper of only predicting tags from the first WordPiece token for each original token in the input sentence so that we generate a single predicted tag for each token.

3.4.2 Implementation & Optimization

All models and training code were implemented using Pytorch [18]. We used the huggingface transformers bert_base_cased implementation of BERT [1]. Training was performed on a single GPU available on a p2x.large AWS instance configured with the Ubuntu 16.04 Deep Learning AMI.

We use cross entropy loss as the loss function and minimize the average negative log likelihood loss per batch. The cross entropy loss was optionally weighted to account for class imbalance giving higher penalty to missing a term label in order to bias the model towards labelling terms. Using a weighted loss function increases recall at the expense of precision and vice versa. A comparison between the two revealed that weighting the loss function performed much better overall.

Optimization was performed using mini-batch gradient descent with the Adam optimizer. We used the custom implementation of the Adam optimizer with packaged with the transformers library. We

Table 4: Term Extraction Model Hyperparameters

Parameter	Description	Value	Search Range
# Epochs	# of passes through the data for training	1	N/A
Batch Size	# of sentences in each batch	16	N/A
Learning Rate	# learning rate for Adam optimizer	3e-5	[1e-5, 3e-5, 5e-5]
Dropout Rate	# dropout rate for output layer of BERT	0.3	N/A
Max Sentence Length	Max # of tokens in a sentence	256	N/A
Balance Loss	Whether to penalize the loss function higher for missed term labels	True	[True, False]

Table 5: Term Extraction Results

Metric	Dev	Test
	Token Level	
Macro Recall	.73	.52
Macro Precision	.75	.60
Macro F1	.74	.56
Micro Recall	.75	.53
Micro Precision	.75	.64
Micro F1	.75	.58
	Term Level	
Recall	.65	.60
Precision	.67	.51
F1	.66	.55

leave the Adam parameters at their defaults except that we opt not to correct the bias as in the original BERT paper and we search over learning rates in [1e-5, 3e-5, and 5e-5] as recommended in the BERT paper. Early stopping was utilized which halts optimization if validation loss does not improve on the next epoch.

Table 4 details the hyperparameters settings for the model that performed best on the dev set. An exhaustive search was not performed, instead we only tried varying the learning rate and whether the loss was balanced. All of the BERT parameters were not varied but an additional dropout layer was added to the BERT output layer prior to the classifier layer.

3.5 Results

Term extraction is a highly imbalanced classification task (the vast majority of words are not terms) and thus simple accuracy is misleading. Here instead we focus on precision, recall, and the F1 score. We use F1 as the primary metric, but also note that recall is arguably more important than precision in this case as it is easier for a human to remove erroneous terms from a list than it is for them to go through the text and add missed terms.

There are two levels at which we define evaluation metrics. The most natural level for the way in which we have defined the model is to compute metrics at the token level. For every token in every sentence, the model predicts a BIOES tag and we compare this to the actual tag computing precision, recall, and F1 averaged over the BIES tags. However, the true goal of the model is to get a valid list of terms for the provided text. Thus to get term level metrics we extract all terms that the model predicted at least once for the given text and compare these to all of the labelled terms computing the same recall, precision, and F1 scores. These metrics are reported for the dev and test sets in Table 5 for the model that performed best on the dev set.

The model generally performs better at the token than the term level. This is because the token level gives partial credit to the model if it gets part of a phrase. Additionally, the token level is sensitive to the frequencies of various terms and gets boosted if a model successfully predicts a common term many times. Since many common terms are often single words and phrases are harder for the model, this also gives the model an additional boost.

As to be expected, the model struggles more with phrases than single word terms. One particular shortcoming is that it will often either over-extend or under-extend a phrase. For an example of over-extending, instead of extracting 'tumor suppressor gene' it adds a non-essential adjective onto the front 'faulty tumor suppressor gene'. For under-extending, an example is only extracting the volume-ratio from surface-to-volume-ratio term. This is to some degree a reflection of the uncertainty in what qualifies as a term. For example, 'abnormal p53 gene' was tagged as a term, but 'faulty tumor suppressor gene' wasn't. Many valid terms are compositional (i.e. cell and eukaryotic cell) and so it is difficult to understand when modifiers on a base term constitute a separate term or just some additional context on the base term for the given sentence. Additionally, the model struggles with more general terms. For example, false positives include work, plane, fact, etc while false negatives include condition, career, contact. Many general terms can take on important specific meanings (for example, work has a specific physics connotation), but it is to be expected that the model struggles to sort these out correctly.

3.6 Discussion

The work presented here is preliminary and still a ways away from being sufficiently accurate to use. Here we briefly discuss some of the main challenges and next steps to consider beyond the specific model results discussed in the previous section.

Incomplete tagging of training data: Glossaries and indices are incomplete and thus the training data has many valid positive examples not labelled. Solutions to this could include using crowdsourcing to obtain more fully labelled training data, using Snorkel to combine multiple sources to try to expand labelling coverage, or searching for additional resources (Wikipedia, terminology lists) to build more complete term lists to tag with. It is unclear what impact this incomplete tagging has currently, but it is likely that phrases are undertagged in the training data.

Ambiguity of What is a Term: NER is typically done with very specific entity types (people, drugs, locations, etc.). In contrast, a term is a very broad concept including all kinds of different types of things across multiple parts of speeach. This could make it challenging to generalize and one could consider trying to subdivide terms up further into different subtypes so that a model could learn more specific patterns for more specific entity types. This would also aid downstream relation classification by helping constrain the domain upon which a classifier needs to operate. We have started to work with a simple sub-type classification distinguishing between events and entities, but this could be further expanded upon. This would require additional entity type labels as well however.

Domain general language and more complicated concepts: Many events/processes have either overly complex representations in text (duplication of a cell by fission) or overly simple, domain general representations (i.e. attach). These aren't really terms in the sense that we just want the word attach as a node, but it is important to capture these versions of representation for downstream relation extraction.

Resolving term representations (entity resolution): Many terms have completely different forms of reference that can't be solved with simple lemmatization. There needs to be some way to resolve these into single nodes. This could be done by trying to cluster embedding representations, using common relation structure, or maybe there are some graph algorithms that can denoise in this way and detect duplicate representations of nodes. We additionally introduce a synonym relation to partially relieve this burden, but this relation can only be extracted in cases where the two representations appear in the same span of text. In addition, the surface forms in the text often do not map easily onto the actual concept (i.e. the text uses eukaryotic when it is talking about eukaryotic cells). Examples include acronyms: DNA - deoxyribonucliec acid; alternate names: p53 - p53 protein; uncommon plurals: mitochondrion - mitochondira; and more.

Constructing full lexicons for terms: In addition to resolving completely different representations of the same term, it is important to have lexical information for each of the different terms in order to be able to use them in knowledge applications. This includes determining pluralizations, acronyms, different verb tenses, nominalizations, and so on. We have a tagger that pools all mentions of a term to the same lemma base form and records information such as the part of speech, but this stops well short of a full lexicon.

4 Relation Extraction

4.1 Related Work

Relation extraction is a long standing, difficult problem in NLP with many applications. In its simplest form, which we consider here, it consists of extracting a semantic relationship between two entities (typically the entities are identified ahead of time). There are many different ways to approach the relation extraction problem. For example, sometimes one has a target set of pre-defined relation types or other times one tries to openly extract relations directly from text in what's called open information extraction. In this project, we already had a target set of relations and so we did not investigate open information extraction methods.

Many unsupervised approaches for relation extraction were initially developed that are often classified into pattern-based and distributional-based methods. Pattern-based approaches identify lexico-syntactic patterns that can be matched to term pairs in sentences indicating a particular relation type, such as a taxonomy [12] or meronym [10] relations, holds between the two terms. However, these methods suffer low recall and are sensitive to minor variations in text as they cannot generalize beyond these patterns. Distributional approaches instead looked at the relative distribution of differents entities in texts and used these to infer relations. For example, the Distributional Inclusion Hypothesis [9] extracted subclass relationships by assuming that subclass words only appear in some contexts of the superclass word, but the superclass word should in appear in all of the subclass word contexts.

With the recent advancement of supervised learning and in particular deep learning models for NLP, much recent work and improvement has been in supervised relation extraction. One of the most direct ways to formulate the relation extraction task as a supervised learning task is where the input data is sentences and term pairs and the output is a classification over a pre-defined set of relation types. The standard benchmark task is task 8 of the 2010 SemEval task set: Multi-Way Classification of Semantic Relations Between Pairs of Nominals [13]. See http://nlpprogress.com/english/relationship_extraction.html for a leaderboard of state of the art on this task. As with many other NLP tasks, current state of the art are modified versions of pre-trained language models such as BERT-EM [9] or R-BERT [28].

The most common limitation in relation extraction however is a lack of labelled training data as each relation requires its own set of examples and one typically has many relations they would like to extract. There are many different approaches to try to alleviate this problem short of investing a lot of time and money to hand label data. The first, which we have already touched upon, is using transfer learning. Large pre-trained language models have encoded a lot of language information that is useful for a wide variety of NLP tasks evidenced by their ability to generally outperform other models with less data required. Perhaps most encouraging is the emergence of pre-training approaches that more directly cater to and encode relational information. BERTEM+MTB is an example of such an approach that achieves better results than just using the standard BERT pre-training [9].

An additional approach for limited training data is re-formulating the relation extraction problem in the context of limited training data. FewRel is a dataset benchmark that has done exactly this by formulating relation extraction as a few shot classification task where a model only has a few (typically 5-10) instances of each relation to learn from [11]. Models for few shot tasks typically learn in terms of similarities and thus when they have to classify a new example they identify which relation it is most similar to. BERT-EM+MTB is the current state of the art on FewRel (leaderboard: http://www.zhuhao.me/fewrel/) suggesting the power of having relational pre-training for the limited training data setting. While we did not explore the few shot approach in this work, this could be a good direction to explore especially if the BERT-EM+MTB pre-trained embeddings get released.

Other approaches include weak supervision approaches that rely on less time-intensive approaches for getting labelled data. For example, distant supervision uses an existing knowledge base to label relations between term pairs in sentences [16]. However, not all sentences that contain a term pair with a given relation actually express that relation, making these distantly supervised labels noisy. Mintz introduced this technique by implementing a method of training supervised models on these labels by grouping all sentences containing a term pair into a "bag" and training on these bags with the assumption that at least one sentence in the bag expressed the relation. More recently, a new dataset benchmark for distant supervision has been proposed known as the NYT corpus [23] and new deep learning models have been created on this dataset for the distant supervision approach.

Table 6: Relation Labels.

Relation	Bio101 KB Mapping	Description	Example
SUBCLASS	subclass-of	X is a subclass of Y	nucleus -> organelle
SUPERCLASS	subclass-of	Opposite direction for SUBCLASS	organelle -> nucleus
SYNONYM	synonym	Synonyms refer to the same underlying concept (acronyms, alternate names, etc.)	deoxyribonucleic acid -> DNA
HAS-PART/REGION	has-part, has-region	X has a part or region Y	cell -> cell membrane
PART/REGION-OF	has-part, has-region	Opposite direction of HAS-PART/REGION	cell membrane -> cell
OTHER	other	Term pairs for which none of the above relations hold	mitosis -> nucleus

For example, [15] learn an attention distribution over sentences within a bag to upweight sentences expressing the relation and downweight those that aren't.

Finally, another approach involves combining different information sources such as pattern recognizers or distant supervising KB's together to produce a set of labels. Snorkel is a framework that has standardized this process [21]. Here one can create separate sets of noisy labels using Hearst's pattern recognizer's or a KB for distant supervision and then Snorkel combines these different labeling sources into a single label that attempts to denoise the various labels. These labels can then be used with standard supervised relation extractors.

4.2 Problem Formulation

A relation is defined as a triple consisting of two terms and a relation type holding between those two terms. Thus relations represent directed edges in our knowledge graph and relation extraction consists of determining all of the different relations that hold between every pair of term so that we can connect the nodes in our graph. Here we have focused on taxonomic and meronymic relations though the approach outlined here could be extended to additional relation types for which sufficient training data can be generated. See Table 6 for a list of our relations and their mappings to the Bio101 KB relations in Table 2.

We formulate the relation extraction problem as a supervised task where each input data instance is a sentence with two terms identified in the sentence and each output is one of the six relations listed in Table 6 (see the appendix for a pure distant supervision formulation that didn't get to a good working point). As an example input and output:

- Input: ['All', '[TERM1-START]', 'cells', '[TERM1-END]', 'have', 'a', '[TERM2-START]', 'cell' 'membrane', '[TERM2-END]', '.']
- Output: HAS-PART/REGION

Thus this is a multi-class sentence classification problem where we have the additional input information of which term pair in the sentence is being classified (same as the SemEval Task 8 formulation). The major limitation with this formulation is that we need a sufficiently large, labelled training dataset, which we do not have. As noted in the previous section, there are many different ways to address a lack of training data. Here we use the Snorkel weak supervision framework to produce labels, which is described in greater detail in a later section.

4.3 Dataset

In order to perform this classification, we need a set of sentences with term pairs tagged. We created a new dataset where sentences were extracted from two Biology textbooks: OpenStax Biology 2nd Edition and Life Biology. We additionally need a list of Biology terms to generate term pairs in these sentences. Here we used a list of Biology terms we collected that include the glossary/index of the OpenStax Biology textbook and all of the biology terms from the Bio101 knowledge base.

Table 7: Relation Extraction Data Splits

Split	Sources	Instances	Unique Sentences	Unique Term Pairs
Train	Rest of OpenStax Biology and Life Biology	167093	29830	105236
Dev	OpenStax Biology Ch. 4 and 10, Life Biology Ch. 5 and 11	11935	2065	8642
Test	Life Biology Ch. 39	3106	491	2582

Table 8: Relation Distributions

Split	OTHER	SUBCLASS	SUPERCLASS	SYNONYM	HAS-PART/REGION	PART/REGION-OF
Dev	88%	1.7%	2.1%	0.5%	3.8%	3.8%
Test	97%	0.5%	1.4%	0.1%	0.5%	0.5%

Every sentence was then tagged with this list of terms using the Spacy NLP processing package. Tagging was done using lemmatized forms to be robust to simple lexical modifications such as pluralization. Additionally, we excluded all tagged terms who were not nouns at the root of a noun phrase as these are not valid terms that can have a taxonomic or meronymic relation. For each term pair, the first term in the pair is always the first term that appears in the sentence. We tag the closest occurence of the two terms in the sentence in the event that one or both of the terms appears in the sentence more than once. For each sentence, we then enumerate all term pairs in that sentence and each sentence, term pair pairing represents a single training instance.

All of these sentence, term pair instances were pooled together and then split into a train, dev, and test set. The dev set consists of all sentences from the cell structure and cell cycle chapters of both textbooks (Ch. 4 and 10 of OpenStax Biology, Ch. 5 and 11 of Life Biology). The test set consists of all sentences from Ch. 39 of Life Biology. All term pairs in both the dev and test set were removed from the remainder of the dataset which then formed the training set. Table 7 shows the instance counts along with the number of unique sentences and term pairs per split. Note that here we define unique term pairs taking ordering into account so (cell, cell membrane) is distinct from (cell membrane, cell).

Additionally, for the dev and test set it is important to have hand labelled relation labels so we can evaluate the model (see the next section for details on how we get labels for the training split). Unfortunately, manually labelling every instance even for these smaller splits is quite involved. As a partial solution, we created a set of gold standard hand labels by taking all of instances that were predicted to have a non OTHER relation by any source (KB, model, label functions) and then doing some partial correction for false positives. This is only a temporary solution to get rough estimates for how the model is performing and eventually it will need to be evaluated on a fully labelled set via crowdsourcing to report metrics with full confidence. In particular, we expect recall to be overestimated as there was no easy way to search the negative instances for additional false negatives and precision may be underestimated as miscellaneous sentences containing a term pair not expressing a relation may not have gotten corrected. Table 8 gives the distribution of the relations for the dev and test splits. Note that this is a highly imbalanced classification problem. Note also that the dev set has a particularly high percentage of meronym relations due to the cell structure chapter being very meronym dense and having good support from the Bio101 KB.

4.4 Weak Supervision

Our training dataset consists of sentence-term pairs instances, but importantly it is missing relation labels which are necessary to train a supervised model. Due to the lack of labelled training data, we employ a form of weak supervision to create labels using the Snorkel framework [21]. The motivation for this approach is that hand-labelling sufficient training data (strong supervision) is prohibitively time intensive. Instead, we can try to label the data programmatically (weak supervision). This produces labelled data much more quickly at the expense of accuracy and coverage in those labels. Snorkel provides a framework for doing this labelling which we adopted:

Table 9: Label Functions.

Label Function	Description	Relations	Coverage
Pattern-Based			
Has Pattern	X have/has Y	HAS-PART/REGION	.4%
In Pattern	X in the/a/an Y	PART/REGION-OF	.6%
Possesive Pattern	X's Y	HAS-PART/REGION	.2%
Contains Pattern	X contains/containing Y	HAS-PART/REGION	.2%
Consist Pattern	X consists/consisting of Y	HAS-PART/REGION	< .1%
Part Of Pattern	X is/are part(s) of	PART/REGION-OF	< .1%
Is A Pattern	X is a/an Y	SUBCLASS	.5%
Such As Pattern	X such as Y	SUPERCLASS	.5%
Including Pattern	X including Y	SUPERCLASS	.1%
Called Pattern	X called Y	SUPERCLASS	.2%
Especially Pattern	X especially Y	SUPERCLASS	< .1%
Appos Pattern	X, a/an Y	SUBCLASS	.1%
And Other Pattern	X and/or other Y	SUBCLASS	.1%
Are Pattern	X are Y	SUBCLASS	.4%
Symbol Pattern	X[,-]Y[,-]	SUPERCLASS	.2%
Also Known Pattern	X also known as Y	SYNONYM	< .1%
Also Called Pattern	X also called Y	SYNONYM	< .1%
Parens Pattern	X (Y)	SYNONYM	.3%
Plural Pattern	X (plural/singular = Y)	SYNONYM	< .1%
Term-Based			
Term Modifier	eukaryotic cell, cell	SUBCLASS, SUPERCLASS	1%
Term Subset	oncogene gene	SUBCLASS, SUPERCLASS	.2%
Distant Supervision			
Bio101 KB other	no relation in Bio101 KB	SYNONYM	66%
Bio101 KB has-part	has-part in Bio101 KB	HAS-PART/REGION, PART/REGION-OF	1.4%
Bio101 KB has-region	has-region in Bio101 KB	HAS-PART/REGION, PART/REGION-OF	.3%
Bio101 KB subclass	subclass in Bio101 KB	SUBCLASS, SUPERCLASS	1.3%
Bio101 KB synonym	synonym in Bio101 KB	SYNONYM	.3%

1. First one must define a set of labelling functions where each labelling function takes in a sentence-term pair instance and outputs either a relation label or a special ABSTAIN response indicating that that labelling function cannot make a judgement on the given instance. For relations that are directional, we define separate relation classes for both directions of the relation. The relation classes we use here are listed in table 6.

2. Second, these label functions are applied to every term pair-sentence instance producing a set of labels for each instance.

3. Third, the set of labels for each instance need to be aggregated to form a single label.

Step 1 requires the generation of a set of label functions. We generated a set of label functions using several different approaches. For a full set of label functions see Table 9. Here we describe the main classes of these label functions in greater detail:

1. Pattern-Based Label Functions: Pattern-based labelling functions look for sentence structures that are indicative of a particular relation between terms. These are similar to matching strings with regex patterns, but instead we match patterns on the dependency parse path between the two terms using Spacy's dependency parser. For example, the has pattern label function labels two terms in a sentence with the HAS-PART/REGION relation if the dependency path between them consists of a single step which is the verb has or have.

2. Term-Based Label Functions: Term-based labelling functions use heuristics on the terms themselves independent of the sentences they appear in. For example, if two terms end with the same base word but one has an additional modifier in front of it, this suggests a taxonomic relation (i.e. eukaryotic cell SUBCLASS cell).
3. Distant Supervision Label Functions: Distant supervision labelling function use an already existing knowledge base as a lookup to label relations between terms. Here we use the Bio101 knowledge base that was manually built on the first 10 chapters of the Life Biology textbook [2].

Step 2 is handled by functionality provided in the Snorkel package. In addition to labelling each instance with all of the label functions, Snorkel also provides the coverage for each label function (see table 9). Aggregating all of our label functions provides an overall 66% coverage of our training dataset so we are still losing almost a third of our training data.

Step 3 requires a method of aggregating the labels for each instance. There are two ways of doing this aggregation that we investigate here:

- Majority Vote (Hard Labels): We can produce a single hard label for each data instance by taking a majority vote across labelling functions. This means that the relation that was labelled by the most labelling functions becomes the label for that instance. In the case there is a tie or no label functions applied, then the label is ABSTAIN and this instance can't be used for training.
- Snorkel Label Model (Soft Labels): We can produce a probability distribution across relations (soft label) by using the label model included with the snorkel package [22]. The label model learns a latent underlying confidence for each label function using the co-occurences amongst each of the label functions produced in step 2. Label functions that are estimated to be more reliable are given more weight and those that are less reliable are given less weight. Thus when multiple label functions label the same instance, their label votes are combined based on their reliability estimates. For example, an instance that receives a certain label from three label functions will have a higher confidence in that label than one that receives a single label or conflicting labels. Thus the labels output by the label model are a probability distribution over the relation classes rather than a single chosen relation.

The soft labels in theory can help mitigate the effect of noisy labelling functions by increasing the influence of more reliable instance labels and decreasing those that are less reliable. In contrast, the hard labels provide no distinction between instances, but have the advantage of being more easily interpretable. We compare the performance of training supervised models with both the hard and soft labels. Training a model with these labels should outperform both the knowledge base and the other pattern/heuristic label functions as it should be able to generalize from both while making predictions on examples that are outside the coverage of the KB/heuristics.

4.5 Model

4.5.1 Model Architecture

We use a new variant of BERT known as BERT-EM [25], which modifies a pre-trained BERT model for supervised relation classification. In BERT-EM four new additional tokens are added to denote the start and end of both terms from the term pair in the sentence. If term 1 ranged from token i to token j and term 2 ranged from token l to token m then our new input representation for the sentence would be:

x = $[x_1$, ..., [TERM1-START], x_i, ..., x_j, [TERM1-END], ..., [TERM2-START], x_l, ..., x_m, [TERM2-END], ..., $x_n]$

These new tokens are shared across every sentence and thus allow a generalized representation of the locations of the two terms in a term pair to be learned across sentences. This modified input sequence is then run through a pre-trained BERT Base encoder. The final encoder layer representations for the TERM1-START and TERM2-START tokens are then concatenated and fed into a linear layer that is run through a softmax to produce a probability distribution over relations. Using the term start tokens

Table 10: Relation Extraction Model Hyperparameters

Parameter	Description	Value
# Epochs	# of passes through the data for training	2
Batch Size	# of sentences in each batch	16
Learning Rate	# learning rate for Adam optimizer	5e-6
Max Sentence Length	Max # of tokens in a sentence	256

helps ensure greater generalizability as the model cannot just memorize individual words as easily as when directly using the representations of the terms themselves.

4.5.2 Implementation & Optimization

All models and training code were implemented using Pytorch and Python. We used the huggingface transformers bert_base_cased implementation of BERT [27]. Training was performed on a single GPU available on a p2x.large AWS instance configured with the Ubuntu 16.04 Deep Learning AMI.

We used a cross entropy loss function and minimized the average negative log likelihood loss per batch. The loss function is the same with both the soft and hard labels, but the target distribution for the cross entropy loss changes. For the hard labels, it is 0 for all classes except the true label. For the soft labels, the target distribution is the probability distribution provided by the soft labels.

Optimization was performed using mini-batch gradient descent with the Adam optimizer. Early stopping was utilized which halts optimization after the validation metrics had not improved an epoch. Table 11 details the most important hyperparameters and there settings when best observed performance on the dev set was achieved. Given that the dev set is not properly labelled, no systematic search over the hyperparameters was performed. Instead a few reasonable values were tried and the best performing set was selected.

4.6 Results

As with term extraction, relation extraction is a highly imbalanced classification task (the vast majority of term pairs have no relation) and thus simple accuracy is misleading. Again we instead focus on precision, recall, and the F1 score.

There are two levels at which we define evaluation metrics. The most natural level for the way in which we have defined the model is to compute metrics at the instance level. For every sentence, term pair instance, the model predicts a relation and we compare this to the actual relation computing precision, recall, and F1 for each class independently as well as averaging. However, the true goal of the model is to get a valid list of triples for the provided text. Thus to get term pair level metrics we extract all term pairs that the model predicted a relation other than OTHER at least once for the given text and compare these to all of the hand labelled term pairs for each relation computing the same recall, precision, and F1 scores.

We compare models trained using both the hard and soft labels. Additionally, we have two baselines to compare the models against. First, we can look up every term pair in the Bio101 knowledge base and compute the same metrics using it as our classifier. Second, we can use the majority vote from all of the pattern-based label functions as a classifier and compute the same metrics again. Note that these two baselines often abstain, but since a judgement needs to be made on every term pair for evaluation we default abstained responses to the majority OTHER relation. Table 11 compares micro-averaged precision, recall, and F1 scores across all relations besides OTHER (macro averages look similar) between the hard and soft labels at the instance level and then all four labellers at the term pair level. As we can see the model trained on the soft labels performs the best, outperforming both the hard labels and the KB and pattern baselines. Table 12 shows the results broken down by individual relations classes when trained using the soft labels.

It is important to remember that these splits are not fully hand labelled and it is unclear to what degree these metrics would change upon full labelling. I strongly recommend not over-interpreting the results, especially for the test set given there are only 10-40 examples for each relation type. However, from these results we tentatively see that the soft labels provide an improvement over the hard labels for training currently. This improvement comes in recall as the precision actually drops.

Table 11: Relation Extraction Model Comparison.

Source	Dev Precision	Dev Recall	Dev F1	Test Precision	Test Recall	Test F1
Over Instances						
BERT-EM Hard Labels	.62	.37	.47	.43	.63	.51
BERT-EM Soft Labels	.62	.42	.50	.31	.76	.44
Over Term Pairs						
BERT-EM Hard Labels	.61	.50	.55	.43	.66	.52
BERT-EM Soft Labels	.61	.54	.57	.30	.76	.43
Bio101 KB	.82	.44	.57	.57	.24	.34
Label Functions	.67	.44	.53	.43	.66	.52

Table 12: Relation Extraction Soft Label Results.

Relation	Dev Precision	Dev Recall	Dev F1	Test Precision	Test Recall	Test F1
Over Instances						
SUBCLASS	.71	.78	.75	.31	.89	.46
SUPERCLASS	.61	.67	.64	.52	.77	.62
SYNONYM	.62	.75	.68	.25	1.00	.40
HAS-PART/REGION	.63	.34	.44	.14	.40	.21
PART/REGION-OF	.46	.15	.23	.21	.86	.33
Over Term Pairs						
SUBCLASS	.71	.80	.75	.32	.88	.47
SUPERCLASS	.62	.69	.66	.53	.76	.62
SYNONYM	.58	.70	.63	.25	1.00	.42
HAS-PART/REGION	.60	.50	.55	.15	.50	.22
PART/REGION-OF	.39	.19	.26	.17	.80	.28

A lot of this is probably due to the fact that with only one negative labelling source, the model puts less certainty in the negative examples compared to positive examples where multiple label functions overlap. This allows the model to better learn the boundary for less certain examples where the hard labels greatly separate all positive and negatives despite there being a lot of noise in these labels. As expected the KB has the highest precision. Its recall suffers from the fact that it abstains on a lot of term pairs since it was only built on one of the two chapters included in this split. Similarly, the model outperforms the label functions by increasing recall, which is the goal of using the Snorkel framework to train a model that can generalize beyond a fixed set of patterns.

Looking at the results by individual class reveals that the model performs much better on taxonomic than meronymic relations currently. This is probably primarily due to a combination of having more label functions for taxonomy as well as less iteration on cleaning up the labels for the meronymic instances. In the dev set, the cell structure chapter is also extremely dense with meronymic relations highlighting a lack of recall. For example, 'cell' shows up in a large number of the sentences and has numerous meronymic relations in that chapter. These splits need to be fully labelled and full hyperparameter searches need to be done before any true judgements can be made.

As expected, false positives are often due to an over-reliance on pattern recognition. For example, 'the cell membrane is a phosholipid bilayer' implies a taxonomic relation by the sentence structure, but in reality is meronymic. Additionally, the model can often get tripped up by more general terms that shouldn't be considered. For example, 'cells have structures called X, Y, and Z' causes the model to assign the meronymic relation between 'cell' and 'structure' when it should really be identifying X, Y, and Z. Other false positives are due to domain inconsistencies. For example, subevent relations look a lot like meronymic relations oftentimes, but the involved terms are events rather than entities. These errors could be alleviated by trying to restrict the term pairs in consideration based on some subtype classification (i.e. only include entities). False negatives are harder to characterize. They include a wide variety of sentences and generally just represent less common sentence structures that the model wasn't able to encode.

4.7 Discussion

We have only scratched the surface of extracting relations for textbook KGs. Here we discuss current limitations and potential next steps to consider.

Expanding Labelling Sources: Adding more label sources improves the label model's ability to denoise and also increases coverage of the training data. This could include more patterns for relations as well other distant supervision sources. For example, Wordnet might be a good additional source of taxonomic and meronymic relations. Wikipedia also has a lot of information on academic subjects that may be scrapeable. Another key aspect is to find more negative label sources. It is much easier to identify a pattern for a relation than to identify a general negative label and currently the KB is the only source of negative labels. Crowd sourcing is another good option as well (see next point).

Integrating with Crowdsourcing: Pattern-based heuristics can only get a model so far and distant supervision is limited to what the knowledge bases cover. Crowdsourcing is the typical manner of collecting labelled data for training. Crowdsourced labels can be incorporated quite naturally into the Snorkel framework by treating each crowd labeller as a new label function. Thus it might be possible to try to use the crowdsourced labels in a complementary way to other label functions to reduce the amount of labelling required and fill in the gaps where programmatic labels are lacking.

Expanding to more Relations: There are many more relations relevant to textbook KG's than just taxonomic and meronymic relations. For some relation types (i.e. spatial relations), it is as simple as expanding the approach here by adding new label functions. For other relation types (i.e. participant relations), it is less clear how the current tagging/extraction scheme will allow identification of these relations. Also, while there are a reasonable number of taxonomic and meronymic relation examples, other relations are not as prevalent or don't have as clear of text patterns, which will make it harder to get sufficient training data for them.

Expanding to More Textbooks: The current set of relations and data are very tightly tied to the Bio101 KB and are only trained on Biology text. It will be important to expand the domain of the training data and also ensure that our relation set is general enough to cover multiple textbook subjects.

Coreference Resolution & Context Expansion: Currently the model can only detect relations between term pairs that appear exactly as recorded in the same sentence. However, many relations are expressed across a larger text span than a single sentence. Additionally, coreference resolution can help improve identification by resolving pronouns to the actual terms they refer to. These modifications would help expand the relations that could be extracted from the text.

Graph Postprocessing & Inference: The model produces a list of triples, but it is unclear exactly how comprehensive/coherent this set is until they are all connected in a graph. In doing so, graph processing techniques could potentially be employed to prune false positive relations and try to infer false negative relations. In this way, the relation extractor from text could act as a seed graph that could then be modified to greater accuracy.

Alternate Modelling Approaches: Snorkel is not the only weak supervision framework to consider. We attempted to use a distant supervision architecture for this task originally (see appendix), but did not get it working correctly. Additionally, one could consider the few shot learning formulation of the problem. These formulations are designed for limited training data scenarios and so maybe there are some modelling approaches there that could work well for our relation extraction.

5 Appendix: Distant Supervision Relation Extraction

This approach did not get to a reasonable working point, but I have included the partial write up here for additional reference. The snorkel approach was switched to instead since its setup was easier making both the modeling and error analysis less complicated. A big issue with this approach was finding the right data splits. It is important to balance the bags so that the model sees enough reasonable negatives and not just negative term pairs that have absolutely no relationship.

5.1 Problem Formulation

In our approach for relation extraction, a relation is defined as a triple consisting of two terms and a relation holding between those two terms. Thus relations represent directed edges in our knowledge graph and relation extraction consists of determining all of the different relations that hold between every pair of term so that we can connect the nodes in our graph. We use a subset of the CLIB relations used in the Inquire knowledge base as our target relation types (CITE/CORRECT). In an initial attempt to automate this extraction, we cast the problem onto the well studied NLP problem of sentence classification where we pass in a sentence containing two terms and try to predict a relation holding between those two terms in the sentence with two major modifications:

1. We need to additionally pass in the position of the two terms to our models. This allows the model to identify which terms it is trying predict the relation between.
2. We do not have an explicitly labeled set of sentences with term pairs to learn from. Instead, we have an existing knowledge base for 10 chapters of Life Biology that gives us many relations that hold between many different term pairs, but these are not linked to specific sentences in the text. Thus, we use a form of weak labeling known as distant supervision. Using the relations identified in the knowledge base, we assume that at least one sentence with any given term pair in the knowledge base exhibits the relation that holds between those terms. Thus we can use the set of all sentences containing a term pair to try to predict which relations hold between that term pair.

With these modifications, rather than classifying individual sentences for a term pair we are classifying an entire bag of sentences and we adjust our models accordingly.

5.2 Data Preparation

In order to create our distantly supervised data for training the model, we make use of an existing knowledge base (CITE INQUIRE) manually constructed on the first 10 chapters of the Life Biology textbook previously mentioned. This knowledge base has hand selected terms for the nodes as well as many different types of relations selected between the nodes. However, it does not link these relations to specific sentences and so we must use this knowledge graph to weakly label these sentences ourselves. As a source of sentences, we combined all chapters sentences from chapters 1-10 and 39-52 of Life Biology in addition to all of OpenStax Biology 2nd edition.

Constructing the relation extraction training data proceeded as follows:

1. For each type of relation, all pairs of concepts (nodes) in the Life Biology knowledge base that were labeled as holding this relationship were extracted. 2. For each concept pair, all possible text representations of each concept in the pair were enumerated to maximize the chances of matching sentences with the expressed concepts. For example, if the first concept had three different text forms and the second concept had two different forms, this would create 6 possible pairings for this concept pair that can each be checked against the text. 3. All text forms of concepts from the preceding step were passed into a tagger that tagged each sentence with all concepts that were present in the sentence. Tagging was done on tokenized and lemmatized versions of the text (Stanford NLP tokenizer and lemmatizer) to ensure robust matching insensitive to pluralization or other lexical modifications present in the text. 4. For each pair of terms, any sentence in which both terms were tagged was included as a potential sentence expressing the relation that holds for that term pair. The terms within the sentence were then annotated so their positioning would be available to the models. In the event that a term in the term pair was tagged more than once in the same sentence we restricted attention to the occurrences of the terms in the sentences that were closest together (had fewest words in between).

As a result of this process, for each relation type we get a set of term pair "bags" where each bag is a set of sentences that had the same pair of terms tagged. The input to the model will be an individual bag with the goal to predict all relation types expressed by the term pair this bag belongs to.

Table XX provides an overview of different relation types and how much data was obtained for each as a result of this process.

While the above process provides us with labelled bags for all the different relation types, it is important to also have negative examples of term pairs with no relation so that the model can

Table 13: **Relation Extraction Data**. Data counts and information for the relation extraction task. Each relation maps from an event/entity term (domain) to an event/entity term (range). Bags Possible is the number of term pair representations that were generated for each relation that in theory could be matched to the text (this is not the number of distinct concepts for each relation, which would be fewer). Bags matched is how many of these pairs had at least one sentence where both terms were tagged. Avg Sentences / Bag denotes the average number of sentences tagged for each term pair.

Domain	Range	Family	Relation	Total Term Pairs	Matched Term Pairs	Avg Sentences per Term Pair
Entity	Entity	Taxonomy	subclass-of	20179	1409	3.2
Entity	Entity	Meronym	has-part	6583	963	7.7
Event	Entity	Participant	object	3050	347	6.7
Event	Entity	Participant	result	3873	338	5.0
Event	Entity	Participant	raw-material	3707	264	4.9
Entity	Entity	Meronym	has-region	3707	255	6.2
			base**	2421	246	6.9
Event	Entity	Participant	agent	1817	187	4.7
Entity	Entity	Spatial	is-inside	1078	167	6.4
Event	Event	Event Structure	subevent	3008	140	2.4
Entity	Entity	Spatial	encloses**	1050	136	7.1
Entity	Entity	Meronym	possesses	1897	120	5.7
Entity	Entity	Meronym	element	535	93	7.4
Entity	Entity	Spatial	is-between	467	80	6.5
Event	Event	Event Structure	next-event	929	77	3.6
			donor**	458	73	3.5
			recipient**	458	73	3.5
Entity	Entity	Spatial	is-at	304	51	6.9
Entity	Entity	Spatial	abuts	166	34	7.4
Event	Entity	Participant	instrument	339	30	4.7
Event	Event	Event Structure	first-subevent	555	30	3.0
Entity	Entity	Spatial	does-not-enclose**	217	23	8.4
Entity	Entity	Spatial	is-outside	195	22	8.5

discriminate against non-relations. For negative examples, we took all term pairs in the knowledge base that had no specified relation between them and created bags using the same process, but with a "no-relation" relation label to form another class that the model could try to predict.

The last step in the data preparation was to split into separate training and evaluation subsets in order to effectively evaluate our models. For this goal, we held out 15% of the bags for each relation separately as a hold out test set for final evaluation of the model and another 15% as a validation set for evaluation of the model during model development. This left 70% of the bags for each relation type for training. Since the number of no-relation bags is much higher than all of the positive example classes, we matched the number of no-relation bags and all other relation bags in each split so that the model was seeing equal number of no relations and some form of relation bags.

5.3 Model Architecture

TODO: spot edit this, fix model figure

Figure XX provides an overview of our model architecture for relation extraction, which we call RDIST-BERT. Here we discuss the different components of our model architecture

5.3.1 Data Input

As described in the previous section, our training data is distantly supervised meaning that we don't have exact labels for individual sentences. Instead we have relation labels for "bags" of sentences that all contain the same term pair with the assumption that at least one of those sentences is a good expression of the provided relation between the two terms. A single input to our model is thus a bag of sentences and the output is a relation label between the two terms tagged in every sentence of that bag. Panel A of figure XX depicts what a bag looks like.

Each sentence in the bag is wordpiece tokenized [CITE] to be compatible with BERT and special tokens are added to denote the start and ends of the terms in the term pair to allow the model to access representations of the terms (see panel B of figure XX for an example and further discussion below). Note that just as each sentence has a varying number of tokens, each bag also has a varying number of sentences depending on how many times a term pair was tagged. Thus we pad/truncate each sentence to a specified maximum sentence length using special padding tokens and we also pad/truncate each bag to a specified maximum number of sentences using padding sentences composed entirely of padding tokens. The result is that every bag is the same size and then we use attention masks to ignore tokens/sentences that are padding and thus not relevant for the model downstream.

5.3.2 BERT

The core of our model is the BERT pretrained language model [CITE]. Specifically, we use the pre-trained weights from the cased version of BERT base available in the huggingface transformers package for pytorch (bert_base_cased). However, BERT is designed to take in text on the order of a single sentence, but our data is in the form of bags of sentences that are too large to pass into BERT at once. Consequently, we pass each sentence in the bag through BERT individually. This results in context-specific vector representations for each individual token in each of the sentences of a bag (see panel C of figure XX).

At this point, a standard sentence classification task would take the sentence classifier representation output of BERT (represented by the [CLS] token) and run this through a classifier layer to be fine-tuned. However, we have two additional complications in our problem formulation:

1. We want to explicitly capture information about the term pairs in each sentence and not just look at the sentence as a whole. This will allow the model to use information about the nature and location of the term pair in the sentence to more accurately classify the relation.
2. We have labels for bags of sentences, not individual sentences so we will have to find a way to aggregate the sentence information taking into account the fact that not every sentence will be a good example of the relation for that term pair.

5.3.3 R-BERT

To address the first complication, we follow the approach of R-BERT presented by Wu & He [CITE] to modify a pre-trained BERT model for relation classification. We present a slightly modified version of their account here, but encourage you to read their paper for further details. The core of their approach is to augment the [CLS] token output representation from each sentence with additional output representations for each of the terms in the term pair within that sentence. There are two steps taken to do so:

1. The first step is to add additional special input tokens denoting the start and end of each of the terms in the sentence (denoted by the [TERM1] and [TERM2] special tokens respectively, see panel B of figure XX). This allows the model to learn with special knowledge of where the term pairs are located in the sentence.
2. The second step is to take the average of the output representations for all of the tokens in each term and pass these term representations downstream in addition to the CLS sentence classifier representation.

Each of these three output representations (the CLS and both terms) are additionally passed through a non-linear activation function and then a fully connected layer before finally being concatenated to form a single vector representation for each sentence. The result of this step is a single vector representation for each of the sentences in the bag. Panel D of figure XX shows this process graphically.

Mathematically, let H^{S1} be the final hidden state (assume BERT hidden state dimensionality d) for sentence 1 of a given bag. Thus, H^{S1} is composed of a d-dimensional vector for each token in the sentence. Assume that H_i^{S1} to H_j^{S1} are the hidden state vectors for term1 and H_k^{S1} to H_m^{S1} are the hidden state vectors for term2 (this excludes the special tokens surrounding them to denote their position) and the CLS token is the first hidden state vector H_0^{S1}. We'll define H_{term1}^{S1}, H_{term2}^{S1}, and H_{cls}^{S1} to be the new output representations as follows:

$$H_{term1}^{S1} = W_{term} \times tanh(\frac{1}{j-i+1}\sum_{t=i}^{j} H_t^{S1}) + b_{term}$$
$$H_{term2}^{S1} = W_{term} \times tanh(\frac{1}{m-k+1}\sum_{t=k}^{m} H_t^{S1}) + b_{term}$$
$$H_{cls}^{S1} = W_{cls} \times tanh(H_0^{S1}) + b_{cls}$$

Here both terms share the same parameters, but the cls representation has its own. The weight vectors are d x d matrices and bias terms are d x 1 vectors. The output of each of the new representations is thus a d x 1 vector. To create a single vector representation for each sentence we simply concatenate these three vectors into a single vector $H^{S1'} \in \mathbb{R}^{3d}$:

$$H^{S1'} = concat(H_{cls}^{S1}, H_{term1}^{S1}, H_{term2}^{S1})$$

.

5.3.4 Neural Attention over Sentences in a Bag

At this point we have transformed our input bag of n sentences into a set of n 3d-dimensional vectors $\{H^{S1'}, ..., H^{Sn'}\}$. However, we only have a single label for the entire bag and so we must now find a way to combine information across sentences to produce a single prediction. In doing so, it is important to keep in mind that not all sentences in a particular term pair bag will clearly exhibit the relation between that term pair (though we assume at least one sentence in the bag adequately does). Ideally, a solution to this problem will be able to upweight sentences that clearly express the relation and downweight noisy sentences that do not. In order to achieve this, we augment our model with selective attention over the sentences in a bag as first proposed Line et al. [CITE]. Here I present a slightly modified account of their approach for our situation.

The key idea with selective attention is that we will learn a weighted sum over sentences in a bag that gives greater weight to sentences that better express the relation between the terms and gives less weight to noisy sentences that do not express the relation. Mathematically, this will reduce our bag representation to a single vector $H^{bag} \in \mathbb{R}^{3d}$:

$$H^{bag} = \sum_{i=1}^{n} w_i H^{Si'}$$

The key is how to learn the weights w_i. Assume there are k different possible relation classes and for the current example we know that the target relation is relation j. To learn the weights, we use a form of attention to learn scores for how well each input sentence representation matches a learned representation for the given jth relation. A higher score means that sentence is more useful for identifying the given relation. To calculate the score for each sentence, we use a simple dot product between the relation representation $r_j \in \mathbb{R}^d$ and the input sentence representation:

$$score(H^{Si'}, r) = H^{Si'} \cdot r$$

The weights for each of the sentences is then just a normalized version of the scores using a softmax computation:

$$w_i = \frac{e^{score(H^{Si'}, r)}}{\sum_{j=1}^{n} e^{score(H^{Sj'}, r)}}$$

The representations for the different relation types are thus parameters to be learned. For k different relation types being classified between, we can construct a relation representation matrix $R \in \mathbb{R}^{kxd}$ where each row j is the relation representation r_j for relation j.

This relation matrix R serves a dual purpose in that is also presents a weight matrix taking the bag representation from d dimensional to k dimensional space to produce scores for each relation ($o \in$ which can then be run through a softmax layer to get a probability distribution across the different relation classes.

$$o = RH^{bag} + b_r$$
$$P(relation\ j) = \frac{e^{o_j}}{\sum_{i=1}^{n} e^{o_i}}$$

This has shown how to get predicted probabilities for each relation class when we know ahead which relation class is the correct one. However, this is only the case during training and during prediction we won't know which r_j relation representation to use. In this case, we calculate a bag representation as described above for every relation to get k different bag representations $\{H_1^{bag}, ..., H_k^{bag}\}$, one for defining attention using each relation query vector. Finally to get the output relation scores for each relation we take the dot product of the bag representation for that relation attention and the relation query vector:

$$o_j = H_j^{bag} \cdot r_j$$

The output probabilities for each of the relations are obtained by taking the softmax transformation of this new o vector.

References

[1] Huggingface pytorch implementation of bert. `https://github.com/huggingface/pytorch-pretrained-BERT`.

[2] Inquire: An intelligent textbook. `http://web.stanford.edu/~vinayc/intelligent-life/`. Accessed: 2019-03-15.

[3] Openstax: Free textbooks for download. `https://openstax.org/subjects/science`. Accessed: 2019-02-15.

[4] Mercedes Arguello Casteleiro, George Demetriou, Warren Read, Maria Jesus Fernandez Prieto, Nava Maroto, Diego Maseda Fernandez, Goran Nenadic, Julie Klein, John Keane, and Robert Stevens. Deep learning meets ontologies: experiments to anchor the cardiovascular disease ontology in the biomedical literature. *Journal of biomedical semantics*, 9(1):13, 2018.

[5] Mercedes Argüello Casteleiro, Maria Jesus Fernandez Prieto, George Demetriou, Nava Maroto, Warren J Read, Diego Maseda-Fernandez, Jose Julio Des Diz, Goran Nenadic, John A Keane, and Robert Stevens. Ontology learning with deep learning: a case study on patient safety using pubmed. In *SWAT4LS*, 2016.

[6] Merley Conrado, Thiago Pardo, and Solange Rezende. A machine learning approach to automatic term extraction using a rich feature set. In *Proceedings of the 2013 NAACL HLT Student Research Workshop*, pages 16–23, 2013.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Jody Foo and Magnus Merkel. Using machine learning to perform automatic term recognition. In *LREC 2010 Workshop on Methods for automatic acquisition of Language Resources and their evaluation methods, 23 May 2010, Valletta, Malta*, pages 49–54. European Language Resources Association, 2010.

[9] Maayan Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 107–114. Association for Computational Linguistics, 2005.

[10] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 1–8. Association for Computational Linguistics, 2003.

[11] Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. *arXiv preprint arXiv:1810.10147*, 2018.

[12] Marti A Hearst. Automated discovery of wordnet relations. *WordNet: an electronic lexical database*, 2, 1998.

[13] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid O Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38. Association for Computational Linguistics, 2010.

[14] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *arXiv preprint arXiv:1812.09449*, 2018.

[15] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, 2016.

[16] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[17] Youngja Park, Roy J Byrd, and Branimir K Boguraev. Automatic glossary extraction: beyond terminology identification. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[18] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[19] Giulio Petrucci, Chiara Ghidini, and Marco Rospocher. Ontology learning in the deep. In *European Knowledge Acquisition Workshop*, pages 480–495. Springer, 2016.

[20] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the thirteenth conference on computational natural language learning*, pages 147–155. Association for Computational Linguistics, 2009.

[21] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *The VLDB Journal*, 29(2):709–730, 2020.

[22] Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4763–4771, 2019.

[23] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer, 2010.

[24] Francesco Sclano and Paola Velardi. Termextractor: a web application to learn the shared terminology of emergent web communities. In *Enterprise Interoperability II*, pages 287–290. Springer, 2007.

[25] Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. Matching the blanks: Distributional similarity for relation learning. *arXiv preprint arXiv:1906.03158*, 2019.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[27] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[28] Shanchan Wu and Yifan He. Enriching pre-trained language model with entity information for relation classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2361–2364, 2019.

[29] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.