Automatic Extraction of Textbook Glossaries Using Deep Learning

Manish Singh
Stanford University
msingh9@stanford.edu

Matthew Boggess
Stanford University
mattbogg@stanford.edu

Supervised by Dr. Vinay Chaudhri

Abstract

Inquire is a research project at Stanford University to design an "intelligent" biology textbook that allows for more advanced interactions with the text by students. Currently, a knowledge base supporting this functionality must be extracted manually, which is prohibitively time-consuming. In this project, we focused on an important initial approximation of the full knowledge base extraction problem: automatically extracting glossaries from a textbook. We scraped a new dataset for this task from open source science textbooks and achieved a corrected F1 score of 0.74 for extracting glossary terms and an F1 score of 0.47 for extracting glossary term definitions. Though these scores are not immediately usable, our work has provided a useful foundation for future work on the Inquire project.

1 Introduction

Inquire is a Stanford research project to design a "prototype of an intelligent textbook that answers students' questions, engages their interest, and improves their understanding" [2]. This prototype provides advanced interactions with the textbook, such as automatically suggesting questions to test understanding when students highlight a section of text. Inquire's "intelligent" functionality is enabled by an underlying ontology that encodes key terms and their relationships in a structured knowledge base. Currently, this knowledge base is constructed manually, which is both difficult and time-consuming.

Extracting a complete knowledge base is a difficult problem since standardized lists of relevant terms for different subjects do not exist. Instead, we focus on a simpler approximation: automatically extracting textbook glossaries. A textbook's glossary represents an incomplete, yet expertly labeled subset of key terms and definitions for the textbook's subject matter.

Our contributions to the glossary extraction problem are as follows:

1. We have constructed a new dataset for the glossary extraction task by scraping chapter sentences and glossaries from open source science textbooks.
2. We have achieved a corrected F1 score of 0.74 on the glossary term extraction task using adapted versions of existing sequence labeling deep learning models.
3. We have achieved an F1 score of 0.47 on the glossary definition sentence extraction task using adapted versions of existing sequence classification deep learning models.

2 Related Work

Extracting key 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 [15, 21]. These workflows later evolved to incorporate machine learning techniques with algorithms like decision trees naturally replacing these manually constructed rules [7, 9]. 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 [6, 5]. [18] 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 key tokens in sentences such as names or locations (for a recent review of deep learning approaches to NER see [12]). We chose to base our efforts at extracting glossary terms on two notable NER models that have each achieved state of the art results in the past. The first model, proposed by Hovey and Ma, exploited both word context information and character sequence information with a combined convolutional and recurrent neural network architecture to achieve state of the art NER results in 2016 [13]. More recently, Google has released a model known as BERT that uses pre-trained representations that can be fine-tuned for a wide variety of NLP problems, including NER where it has achieved near state of the art results.

Definition sentence extraction is a binary classification problem that involves classifying each sentence in a corpus as a definition sentence or not. Initial machine learning approaches used algorithms such as decision trees to exploit the syntactic structure of definition sentences [23]. More recently, deep learning models have been applied to definition-related tasks. [10] used recurrent neural network architectures trained on definition data to produce two applications related to the inverse of the definition extraction problem. [14] introduced the problem of definitional modeling and used a recurrent neural network language model to generate definition sentences for input words. Anke et al. used a combination of convolutional neural networks and recurrent neural networks to exploit both local word phrase structures as well as longer distance relations between these local phrases for directly classifying definition sentences [4]. BERT has also been used successfully in a wide variety of sequence classification tasks and can similarly be fine-tuned as in the NER formulation. We focus our efforts here on applying the Anke et al. model and a fine-tuned BERT model to our glossary definition sentence dataset.

3 Approach

3.1 Glossary Term Extraction

We approached the glossary term extraction problem as a sequence labeling problem. This entails tagging each word/phrase in each sentence according to the BIOES tagging scheme commonly employed in NER tasks [19]. B, I, and E refer to the beginning, interior, and end of a glossary term phrase respectively. S refers to a singleton glossary term and O is the label for all other tokens not part of a glossary term.

We compared two model architectures that have performed well on token classification tasks. The first model, proposed by Hovey and Ma, is pictured in the left panel of Figure 1 [13]. Their model combined both character-level representations and word-level embeddings into a single concatenated representation for each word. Word embeddings are generated using fixed pre-trained GloVe embeddings [17]. Character representations are generated using freely trainable character embeddings that are then fed into a convolution layer followed by a max pooling layer. Using both representations allows one to exploit both the word context encoded by the GloVe representations as well as the character sequence structure present in lots of scientific terms (i.e. oxygen and oxidation). This concatenated representation is then fed into a BLSTM layer that in turn feeds into a fully connected softmax output layer that produces a predicted tag for each token in the sentence. Hovey and Ma used a conditional random field output layer instead of a simple softmax to exploit the conditional nature of the BIOES tags. We did not add this layer in our re-implementation, but it could be explored in future work as it has been shown to improve performance.

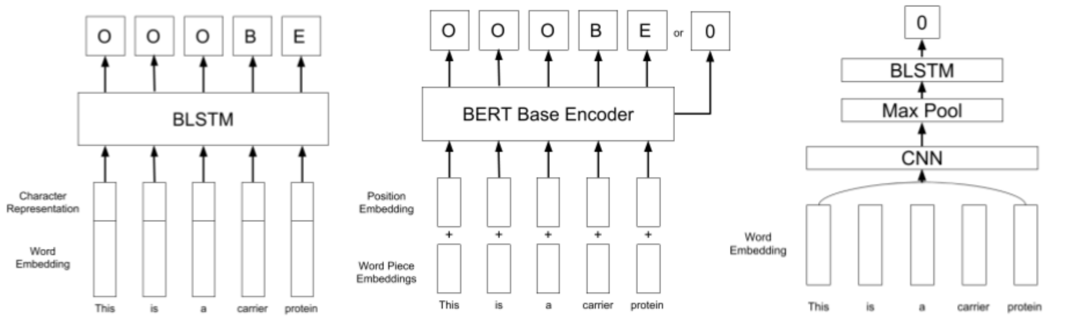

Figure 1: Model Architectures. **Left.** Diagram of the Hovey and Ma term extraction model [13]. **Middle.** Diagram of the BERT Base term and definition extraction models [8]. **Right.** Diagram of the Anke et al. definition extraction model [4].

More recently, Google has released a model known as BERT that is pictured in the middle panel of Figure 1 [8]. The BERT model (we use the huggingface PyTorch implementation of the smaller BERT Base version) involves an encoder consisting of 12 transformer attention hidden layers [22]. This encoder is pre-trained on a language modeling task that involves predicting masked words in sentences from the other non-masked words. The inputs to the encoder are a combination of WordPiece embeddings [24] as well as position indicators for each word. This pre-trained encoder and embeddings can then be fine-tuned for classification tasks. In our case, this involves placing a fully connected softmax output layer on the final hidden layer of the encoder that predicts the tag for each token. The only caveat 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 for each token in the input sentence so that we generate a single predicted tag for each token.

For both models, we use a cross entropy loss function that has masks to ensure that extra padding tokens used to make the sequence lengths all equivalent do not get included in the loss calculation.

A previous class project for Stanford's CS229 Machine Learning class made an initial attempt at the glossary term extraction problem for the Inquire project [11]. They formulated the problem at the n-gram level and fit more conventional machine learning models on unigrams, bigrams, and trigrams extracted from 15 chapters of the Life Biology textbook being used for the Inquire prototype. They achieved their best performance using a multilayer perceptron which achieved a corrected F1 score of 0.316, which we will compare against as a baseline.

3.2 Glossary Sentence Extraction

We approached the glossary sentence extraction problem as a binary sequence classification task where every sentence in the textbook was classified as a key term definition sentence or not. We compared two different sequence classification models on this task. The first is a network proposed by Anke and Schockaert in [4] and is outlined in the right panel of Figure 1. Their network used word2vec word embeddings to encode pre-trained contexts for each word. These embeddings are then fed through convolution layers to extract n-gram type local features in the sentence and then max-pooled to reduce feature size. These outputs are then fed into a BLSTM layer that can encode longer sequence information across these local n-gram features. The output states from both directions of the BLSTM are then fed into a single sigmoid output layer producing the probability of the sentence being a definition or not. They used dropout on the output of the max-pooling layer and on the output of the BLSTM. We have re-implemented this model with some minor differences. The most notable being that we have used GloVe embeddings rather than word2vec embeddings.

The second model is again a fine-tuned version of Google's BERT model. The details of the model are the same as described in the glossary term extraction section with the exception of the output layer. We followed the BERT authors' recommendation of using the output from a special 'CLS'

Table 1: Textbook Glossary Dataset Statistics

Textbook	Sentences	Glossary Terms	Glossary Sentences	Data Split
Open Stax Anatomy	17804	2948	2607	Train/Dev
Open Stax Astronomy	13384	306	493	Train/Dev
Open Stax Chemistry	8262	677	578	Train/Dev
Open Stax Physics	5500	362	322	Train/Dev
Open Stax Microbiology	11509	1492	1322	Train/Dev
Open Stax Biology	20529	1966	2101	Train/Dev
Life Biology	28662	1957	1999	Test

classification token prepended to each sentence and fed this output into a single sigmoid output layer that produces the desired definition sentence probability.

We additionally tried a model combining the two architectures. This model first feeds the sentences into the BERT encoder. The outputs of the BERT encoder are then fed into the Anke et al. architecture to produce the predictions. The idea with this combination was to combine the representational power of BERT with the syntactic feature extractor of Anke et al. For all models, we used a cross entropy loss function.

4 Experiments

4.1 Data

4.1.1 Textbook Glossary Data

We created a new dataset for the glossary extraction task. For training and development data, we scraped six open source science textbooks from Open Stax [3]. Finally, we obtained a copy of the Life Biology textbook being used for the Inquire project and processed it to use as a test dataset. For each textbook, we extracted chapter sentences and glossary terms from lists at the end of the textbook or individual chapters. We then tagged every word of every sentence using the BIOES tagging scheme for whether it was part of a glossary term/phrase or not. We extracted glossary definition sentences by finding sentences in the text that contained potential terms in bold. Each sentence was then tagged as either a glossary definition sentence or not. Sentence counts, glossary term counts, and glossary definition sentence counts are presented in table 1. The glossary term and definition counts can differ because sometimes glossary terms are not explicitly defined or additional non-glossary words are bolded in the text.

4.1.2 Additional Definition Sentence Data

We used two additional datasets that were used to evaluate the model in [4] to augment our definition sentence data:

- World Class Lattices (WCL) Dataset: This corpus contains 1871 definition sentences and 2847 non-definition sentences from Wikipedia [20].
- W00 Dataset: This is a corpus consisting of 2182 sentences with 751 definition sentences from workshop papers from the Association for Computational Linguistics Conference in 2000 [25].

These datasets were mixed into the training data to provide the definition model with additional positive samples to aid in training.

4.2 Evaluation Metrics

We report the F1 score, precision, and recall for both the glossary term and definition extraction tasks. Both are highly imbalanced classification problems and thus accuracy is not as meaningful of a metric. We use F1 as the primary metric, but also wanted to note that recall is arguably more important than precision. This is because it is easier to manually filter down the list of candidate

Table 2: Model Hyperparameters

Hovey Ma		Bert Term / Definition		Anke et al.	
Parameter	Value	Parameter	Value	Parameter	Value
Num Epochs	20	Num Epochs	5 / 10	Num Epochs	20
Batch Size	16	Batch Size	16 / 8	Batch Size	64
Learning Rate	5e-4	Learning Rate	2e-5 / 2e-5	Learning Rate	1e-3
Dropout Rate	0.3	Dropout Rate	0.3 / 0.5	Dropout Rate	0.5
Word Embedding	300	Num Hidden Layers	12	Word Embedding	300
Char Embedding	50	Num Attention Heads	12	LSTM Hidden Layer	100
LSTM Hidden Layer	200	Hidden Layer	768	Num CNN Filters	100
Num CNN Filters	30	Feed Forward Layer	3072	CNN Window Size	3
CNN Window Size	3			Max Pool Size	4

terms and sentences than it is to manually identify missing terms or sentences. For models with equal or close F1 scores, we thus favor recall over precision.

For the glossary definition extraction task, these metrics are derived from the predicted categories of definition sentence or not for each sentence. For the glossary term extraction model, we need a process for extracting candidate terms for an entire textbook from our model since it is designed to tag terms at the sentence level instead. We do this by proposing every single word/phrase that was tagged in at least once sentence by our model and comparing these candidate terms to the actual textbook glossary terms. One could also have used more sophisticated selection algorithms such as thresholding by the average confidence the model assigned to each term, but our analysis suggested such an approach would not be helpful.

Given that glossaries are only approximations of the full set of key concepts in a textbook, the model may be unfairly penalized for producing false positive terms that are actually valid scientific terms. We had a domain expert tag our outputted list of false positives as either legitimate false positives or actual true positives. We can then recompute our metrics with respect to the updated labels to get corrected versions of these metrics. It is important to note that this is an optimistically biased correction since we would also want to do a similar correction for true negatives that were actually false negatives. However, it is infeasible to do the correction in this direction since the list of potential false negatives to be corrected is every single other n-gram in the book not marked by the model. We additionally screened out parsing errors and plurals that penalized the model when these should have instead been taken care of in the data processing.

4.3 Experimental Details

All models were implemented and run using PyTorch [16] on Microsoft Azure Standard NV6 Virtual Machine GPUs.

4.3.1 Glossary Term Extraction

Hyperparameter settings for the Hovey and Ma model and the BERT model are reported in Table 2. For the Hovey and Ma model, we fixed most hyperparameters at the values reported in the paper with a few notable exceptions:

- We used the Adam optimizer and searched over several learning rates between 1e-5 and 1e-3. The original implementation used a simple stochastic gradient descent optimizer with additional gradient clipping.
- We searched over several dropout rates between 0 and 0.5 selecting a final value of 0.3 compared to 0.5 in the paper.
- We used a slightly larger batch size of 16 compared to 10.
- We ran for 20 epochs. Each epoch took approximately 1.5 minutes to run resulting in a running time of about 30 minutes for the model.
- We used the full 300-dimensional GloVe case-sensitive embeddings [17] compared to 100-dimensional versions used in the paper.

We used the huggingface PyTorch bert-base-cased implementation of BERT [1]. This comes with a tokenizer and a custom Adam optimizer that we used as well. We searched over learning rates between 2e-5 and 5e-5 as suggested in the BERT paper as well as dropout rates between 0 and 0.5. We ran each model for 5 epochs. Each epoch took approximately 20 minutes to train for a total running time of 1 hour 40 minutes.

4.3.2 Definition Sentence Extraction

Hyperparameters for the BERT and Anke et al. models are listed in Table 2. For the Anke et al. model, most hyperparameters were fixed at values reported in the paper with the exception of the following hyperparameters that were optimized over the given ranges:

- learning_rate = [1e-4, 1e-3, 1e-2]
- batch_size = [32, 64, 128, 256, 512]
- dropout_rate = [0.6,0.7,0.8,0.9]
- cnn_kernels = [3,4,5,6,7,8,9]

We used the same huggingface implementation of BERT. We did not have time to do hyperparameter searches for the BERT model, but fixed these at values suggested in the paper that worked reasonably. For the BERT + Anke combination, we used the BERT hyperparameters where they overlapped and each model's best individual hyperparameters.

5 Results

5.1 Quantitative Results

5.1.1 Glossary Term Extraction

Results for the glossary term extraction models are reported in Table 3. For the Hovey and Ma model, we have additionally compared performance with only custom word embeddings (BLSTM) and with only GloVe embeddings (BLSTM-GloVe) to evaluate the importance of the pre-trained word embeddings and character representations. We also report the false positive corrected scores for the BERT model. The F1 scores for the BERT and Hovey and Ma models were roughly equivalent so we deferred to the BERT since its recall was higher.

We see that using pre-trained word embeddings significantly improves performance as expected. The character representations in the Hovey and Ma model additionally allow the model to identify several more terms that it was previously missing. Somewhat surprisingly we saw that BERT and the Hovey and Ma model performed similarly. We had expected BERT to perform better, though we believe it technically still produced better results by producing a better recall. With the domain expert false positive corrections, the results significantly improve. This reveals the severe limitation of having an incompletely tagged dataset. We discuss this in further detail in the analysis section.

5.1.2 Glossary Definition Extraction

Results for the glossary definition extraction models are reported in Table 3. We first ran each of the models using only the WCL and W00 datasets respectively and selected the best performing models for each as baselines to compare against. For the Anke et al. model we additionally compared the contribution of GloVe embeddings as compared to custom trainable embeddings.

We see that all of the models beat the baseline performance using only the highly structured additional definition datasets. This suggests that the models are able to make use of the additional information specific to the textbook definition data. For the Anke et al. model we see that GloVe embeddings significantly improve performance as expected. The BERT models well outperformed the Anke model. However, the best model was combination of the two that could exploit both BERT's pre-trained representations and the syntactic specific architecture of Anke et al. In particular, this combined model performed better in terms of precision, which suggests that the added syntactic filtering helped screen out false definition candidates.

Table 3: Glossary Extraction Results

Term Extraction			
Model	F1	Recall	Precision
Baseline w/ Correction	0.316	0.421	0.254
BLSTM	0.340	0.453	0.272
BLSTM-GloVe	0.400	0.431	**0.370**
Hovey Ma	**0.432**	0.527	0.366
BERT	0.430	**0.541**	0.357
BERT w/ Correction	**0.741**	**0.708**	**0.778**

Definition Extraction			
Model	F1	Recall	Precision
WCL Baseline	0.16	0.27	0.12
W00 Baseline	0.26	0.23	0.29
Anke-Custom	0.30	0.27	0.34
Anke-GloVe	0.38	0.39	0.37
BERT	0.44	0.46	0.42
BERT + Anke	**0.47**	**0.46**	**0.48**

5.2 Analysis

5.2.1 Glossary Term Extraction

A severe limitation of our current approach to the problem is that glossaries represent highly incomplete sets of key terms for a textbook. As an example of how this affects performance, we have broken down the false positives tagged by the BERT model into sub-categories in the upper left panel of Figure 2. We see many more valid glossary terms (as corrected by a domain expert) than actual false positives. This means that the model is being unfairly penalized despite outputting legitimate terms in a majority of the cases. The upper left panel of Figure 2 also shows several other false positive categories related to dataset processing. For example, we currently do not account for plurals or morphological variations (depolarized vs. depolarization) and there were several parsing errors that led to parts of key terms or key terms with erroneous characters appended getting tagged.

The incomplete tagging creates an arguably bigger problem for training the model. Incomplete tagging on the training data means that the model is often getting conflicting training signals. In the worst case, we can look at the fraction of key terms in the test data that were present in the training data, but not tagged as key terms (upper right panel of Figure 2). We see that a much larger proportion of the false negatives conflict with the training data in this way. It is understandable that it will be more difficult for the model to learn these terms when it is only being shown them in contexts where they are not tagged as key terms during training.

The lower right panel of Figure 2 shows the model confidence scores (probability of the correct tag(s)). We see that the model is biased towards high confidence for true positives, but not false positives. This suggests that using a threshold to screen out false positives will likely not work well.

5.2.2 Glossary Definition Extraction

The glossary definition extraction is limited in a similar manner to the term extraction task. For example, in the lower left panel of Figure 2, we see false negatives split into several categories for a sample of 50 false negatives that was used as an estimate of the breakdown for the full set of false negatives. Almost 40 percent of the false negatives are sentences in which the term was introduced and discussed, but not explicitly defined. For example: "**Soils** have living and nonliving components.".

Additionally, we had several parsing errors that led to clear non-definition sentences getting tagged. These suggest that our assumption for using bolded key terms in sentences as candidate definition sentences is potentially not accurate enough in many cases. For the true false negatives, the model

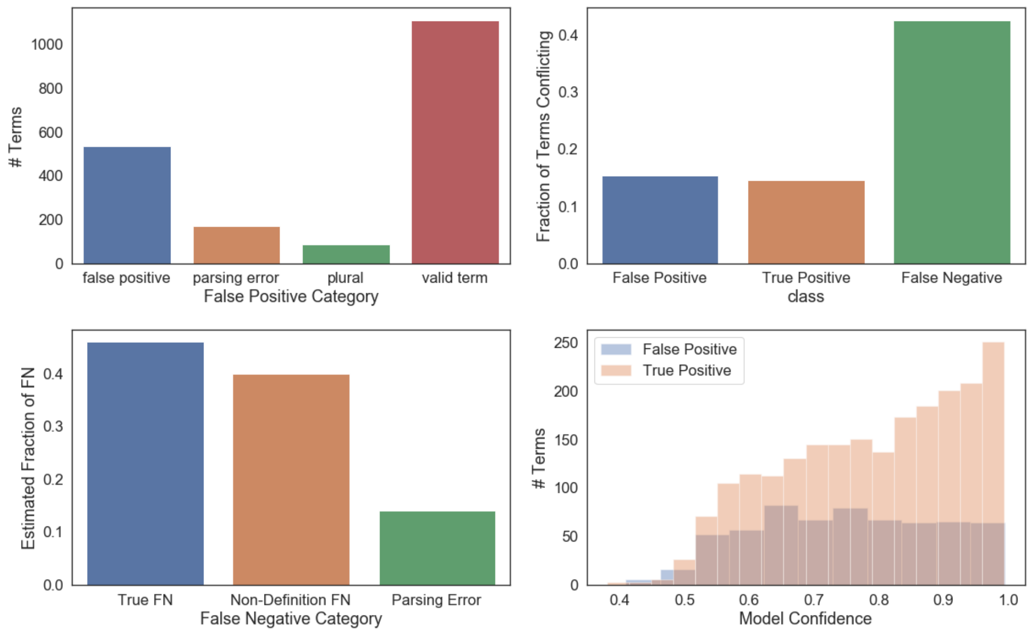

Figure 2: Error Analysis. **Upper Left.** Breakdown of false positives by category for term extraction as determined during the correction process. **Upper Right.** Fraction of terms that were not tagged as key terms in the training data split by error type. **Lower Left.** Fraction of false negatives by category for definition extraction. **Lower Right.** Comparison of model confidence scores between false positives and true positives for term extraction.

struggled with definition sentences that did not follow a more common definition format (is a, is called, etc.). For example: "**Pair rule genes** divide the embryo into units of two segments each."

The definition extraction was also limited by the incomplete nature of the glossary as in the term extraction problem. An estimated nearly 50 percent of false positives were definition sentences that defined reasonable scientific terms that were not explicitly tagged as glossary terms. Correcting for these could have revealed a similar increase in performance as in the glossary term extraction task.

6 Conclusion

While the performance of our models are not immediately usable, we have contributed a new dataset for the project and achieved reasonable initial results with a deep learning approach to the glossary extraction problem. We have observed BERT outperform smaller, more specialized models on both term and definition extraction.

However, it currently seems that performance is primarily limited by data quality and problem formulation in both tasks. More work needs to be done on better parsing the dataset. For example, our method for extracting definition sentences is not always accurate and we need to handle variants of key terms such as plurals. Additionally, it is clear that there are limitations with the current problem formulation as glossaries are incomplete. This leads to both models getting punished for many false positives that are actually true positives. Furthermore it creates a bigger issue with the training as many true positives get labeled as false examples in the training data as a glossary term in one subject is not necessarily a glossary term in another subject.

Future work could look to combine definition extraction and term extraction models. This could potentially alleviate some of the problems listed above because a reasonable assumption is that most key terms will get clearly defined in the text. Thus, a term extraction model that could explicitly exploit definition sentence structure could be able to better select for more important terms that were clearly defined. Indeed, this is one of the reasons we pursued both of these sub-problems together though we did not have time to combine them for the work presented here.

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] Luis Espinosa Anke and Steven Schockaert. Syntactically aware neural architectures for definition extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 378–385, 2018.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30, 2016.

[11] Anita Kulkarni and Rachel Smith. Automated glossary construction of a biology textbook. *Stanford CS229: Machine Learning*, Fall 2018.

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

[13] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*, 2016.

[14] Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. Definition modeling: Learning to define word embeddings in natural language. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[15] 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.

[16] 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.

[17] Jeffrey Pennington, Richard Socher, and Christopher 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.

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

[19] 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.

[20] Juana María Ruiz-Martínez Roberto Navigli, Paola Velardi. An annotated dataset for extracting definitions and hypernyms from the web. In *LREC 2010, Valletta, Malta, May 19-21*, pages 3716–3722, 2010.

[21] 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.

[22] 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.

[23] Paola Velardi, Roberto Navigli, and D Pierluigi. Mining the web to create specialized glossaries. *IEEE Intelligent Systems*, (5):18–25, 2008.

[24] 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.

[25] Jun-Ping Ng Yiping Jin, Min-Yen Kan and Xiangnan He. Mining scientific terms and their defini- tions: A study of the acl anthology. In *In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Seattle, Washington, USA*, page 780–790, 2013.