DEVKOTA, PRATIK. M.S. NEural Models for Ontology annotations - NEMO. (2022)
Directed by Dr. Somya D. Mohanty and Dr. Prashanti Manda. 71 pp.

The rapid progression of technology has allowed a significant increase in the pace of modern, novel scientific experimentations. Important results from these experiments are often buried in rather comprehensive documents and thus information retrieval is difficult. To facilitate retrieval and knowledge discovery, domain experts have been using ontologies (a formal way to represent knowledge within a domain) to annotate important entities. These annotations are generally curated manually which is a slow and laborious process and hence unscalable. As a solution for scalable ontology annotations, Named Entity Recognition (NER) is critical. NER is the task of recognizing ontology concepts from the text. Traditionally, entity recognition was achieved using syntactic analysis, lexical approaches, and traditional machine learning. In recent years, deep learning has shown improved results in terms of concept recognition.

This research explores different approaches to improve the state-of-the-art deep learning models for automated ontology annotations. Here, CRAFT (a manually curated biomedical corpus for ontologies) is used as a gold standard corpus for training and evaluating the performance of different deep learning architectures. We augment the information from CRAFT with several existing knowledge bases. This study demonstrates that we can improve the prediction accuracy of existing deep learning models by including additional information as input pipelines to existing architectures. Additionally, ontologies are hierarchical and have semantic relations between concepts. While deep learning models generally fail to take this hierarchy into account, our work also explores the possibility of making the models ontology-aware and shows improvement over baseline models. Furthermore, we implement a novel concept called Ontology Boosting to boost the prediction accuracy of pre-trained models through post-processing steps.

NEURAL MODELS FOR ONTOLOGY ANNOTATIONS - NEMO

by

Pratik Devkota

A Thesis Submitted to
the Faculty of The Graduate School at
The University of North Carolina at Greensboro
in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Greensboro
2022

Approved by

Dr. Somya D. Mohanty

Dr. Prashanti Manda

APPROVAL PAGE

This thesis written by Pratik Devkota has been approved by the following committee of the Faculty of The Graduate School at The University of North Carolina at Greensboro.

Committee Co-Chair _____
Somya D. Mohanty

Committee Co-Chair _____
Prashanti Manda

Committee Members _____
Yingcheng Sun

_____
Chunjiang Zhu

_____
Date of Acceptance by Committee

_____
Date of Final Oral Examination

ACKNOWLEDGMENTS

Words cannot express my gratitude to my advisors Dr. Somya D. Mohanty and Dr. Prashanti Manda for their invaluable supervision and guidance. I am extremely grateful to them for their continuous support, constructive suggestions, and encouragement. Dr. Mohanty's expertise in Data Science and Deep Learning, and Dr. Manda's expertise in Natural Language Processing and Bioinformatics were instrumental in bringing this study to its current form. I am fortunate to have them as my mentors who inspired me along my journey at the University of North Carolina at Greensboro (UNCG).

I would also like to extend my gratitude to my thesis committee members Dr. Yingcheng Sun and Dr. Chunjiang Zhu for their enthusiasm toward my work and their involvement during the defense. I am equally indebted to the Department of Computer Science at UNCG for providing the resources to run my resource-exhaustive experiments and for providing access to the lab.

I would be remiss in not mentioning my family, especially my parents and my sisters. Their emotional support and belief in me kept my spirits high during my master's degree. I also want to thank my uncle, aunt, and cousins for their advice and inspiration. Last, but by no means least, I would like to thank my long-suffering girlfriend Durga Tandon for her forbearance and motivation throughout this endeavor.

This work is funded by a CAREER grant from the Division of Biological Infrastructure at the National Science Foundation of the United States of America (#1942727).

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Named Entity Recognition (NER) or sometimes also referred to as entity identification/extraction or annotation is the task of recognizing and associating important information from a text to some predefined categories. It is widely used in the field of Natural Language Processing (NLP), for different purposes such as sentiment analysis, trend analysis, topic modeling, decision support, information retrieval, and much more. Text/documents for these tasks can be a sentence, paragraphs, dialogue, literature reviews, or even scientific articles.

Documents like scientific articles, journal publications, and literature reviews often contain novel experiments and findings which can be used for further analysis or to use these findings in real-world applications. However, these documents are generally detailed, and extracting information is often difficult. Traditionally, the process of knowledge extraction was achieved through manual annotation, syntactic and lexical analysis, rule-based analysis, dictionary lookup, and machine learning [1]. Manual annotation is a slow and tedious process that requires domain expertise and the use of specialized software [2]. Lexical approaches use lexical and semantic similarities between a piece of text and an ontology concept to annotate the text with the concept [3]. A rule-based NER system consists of a lexicon and grammar. A lexicon is a collection of named entities that are known and categorized into classes beforehand. Grammar is used to recognize and classify entities not present in the lexicon and decide the class in case of ambiguous entities [4]. These approaches generally have lower accuracy and lower throughput. However, the recent advancement in deep learning algorithms have shown promising avenues in NLP because of their increased accuracy and higher throughput [5–9].

Accurate recognition of biological/biomedical entities has made it possible to integrate, query, and run large-scale analyses on biological data. Computational analyses such as hypothesizing the genetic bases of evolutionary transitions to predicting gene functions to understanding rare human diseases can be performed with higher accuracy [10, 11]. With new deep learning architectures, entities described in the literature are recognized with higher confidence and the rate of missing entity recognition is reduced. This can help improve the quality of clinical documentation, clinical decision support, clinical trial matching, computational phenotyping, and much more.

Nonetheless, applying deep learning for recognizing concepts from biomedical literature possesses significant challenges. Many biomedical concepts (1) are long and descriptive, (2) have a large number of synonyms, (3) are commonly represented using abbreviations and mixtures of letters, symbols, and punctuation, and (4) are even presented differently by different authors [12]. Additionally, biomedical annotations require experts' knowledge and specific, detailed guidelines to create a dataset to feed into deep learning models for training and evaluation purposes. Automation of these annotations is essential to take advantage of the rapid pace of scientific publishing. The accuracy of these annotations is crucial to better the quality of large-scale computational analyses.

As an effort to improve the accuracy of automated annotation using deep learning models, we test different methods and deep learning architectures and analyze their performance. For all our experiments, we use the Colorado Richly Annotated Full Text Corpus (CRAFT) (v4.0.1) as a gold standard corpus for training and evaluation. A gold standard corpus is a set of textual documents that are manually annotated across multiple categories of entities. CRAFT is an open-source corpus, semantically and syntactically annotated to serve as a research resource for NLP tasks. Version 4.0.1 provides a collection of 97 full-length, open-access biomedical journal articles from PubMed Central Open Access Subset. The collection contains ontologies with different classes organized across 10 different modules.

An ontology is a formal representation of knowledge within a domain described by a set of concepts and their relationship to each other. Ontologies not only aim to standardize concepts used in the domain but also help in the classification, inference,

and reasoning. Directed Acyclic Graphs (DAG) are generally used to represent ontologies where concepts are denoted by nodes and semantic relationships between concepts by the edge connecting the nodes. Hierarchy exists between concepts and the directed graph represents both generalization (represents parent) and specialization (represents children).

For any deep learning model training, the quality of dataset is crucial to achieving good performance. In our work, we spend a considerable amount of time preprocessing the CRAFT corpus, making sure that the dataset is sound and complete. CRAFT v4.0.1 introduces some sophisticated annotation formats such as overlapping annotations and discontinuous annotations. Even though the occurrences of these annotations in the corpus are relatively low, all of these occurrences are properly addressed and included in our training and validation dataset. CRAFT has a separate plain-text version of each of the 97 articles. For each of them, a corresponding XML file is provided that contains the concept annotations mapped to specific ontologies. We pre-process the plain text along with the xml file to split the entire document into sentences. To avoid data leakage between the training and validation dataset, the sentences are divided into 80-20 splits. Once separated, all preprocessing steps are applied to both the training and validation datasets. The sentences in either dataset are further split into tokens and each token is normalized either to a concept or not-a-concept. If for a given token, the xml contains the annotation, the token is normalized to that particular annotation. Else, the token is represented as not-a-concept. Additionally, to correctly represent long words or phrases corresponding to a concept, we use the IOB tagging format.

IOB tagging, also known as BIO tagging is a common format to tag tokens in tasks like Named Entity Recognition. The first token belonging to a particular concept is represented as a B-concept (representing the beginning of a phrase), the rest of the tokens belonging to the same concept are represented as an I-concept (representing the inside of a phrase), and any token not belonging to a concept is represented as an O (representing outside of a phrase). Furthermore, we tag each token of a sentence with Parts-of-Speech information. To help the model understand the character level detail of a token, we use character encoding with upper-case characters represented by ‘C’, lower-case characters by ‘c’, numerals by ‘N’, and punctuation kept as is.

In addition, we use Bio-Thesaurus - a web based thesaurus of protein and gene names [13] and Unified Medical Language System (UMLS) [14] metathesaurus for dictionary lookup. The presence of a token in these databases would suggest that the token is more likely to be a concept, thus helping augment the information from CRAFT. Each article, once preprocessed has a corresponding JavaScript Object Notation (JSON) file that contains a list where each item in the list is a list of tokens with augmented information and IOB tags.

All augmented information from preprocessing is used as separate inputs to deep learning models to explore the effect of these inputs on model performance. These deep learning models aim to accurately recognize and normalize concepts present in our corpus. Ontology annotation is generally considered a two-step process. The first step is Named Entity Recognition (NER), to recognize whether an input token belongs to a concept or not. Based on the prediction, the token is assigned one of the IOB tags i.e. the token is either predicted to be a `'B'` tag, `'I'` tag, or an `'O'` tag. The output/prediction of this step is used for the second step where the task is to normalize a concept. Named Entity Normalization (NEN) or Concept Normalization is the process of assigning a concept name or ID to output if a token is predicted as a concept. After combining the output from both steps, each token is predicted to be either a `'B-X'` tag, `'I-X'` tag or `'O'` tag, `'X'` representing concept name/ID predicted from the second step.

Many treat these two steps separately and either develop two different deep learning models, one for each step, or develop one deep learning model to predict two separate outputs (recognition and normalization). We, on the other hand, treat these two steps as one and develop a deep-learning model to predict the combined output of the above described steps. We develop different Gated Recurrent Unit (GRU) models [15] whose architecture differs in the input pipelines added to them. It is concluded from prior works [9, 16] that GRU-based architecture performs the best on CRAFT corpus among vanilla RNN, LSTM [17] and GRU models. Over 100 different experiments are carried out on GRU–based architectures, testing different hyperparameters and supervised embeddings - dense latent space representations of higher dimensional inputs. After hyperparameter tuning, a total of 28 different models are recorded and their performance is evaluated using F1 and Jaccard semantic similarity metrics.

Jaccard similarity is a set-based semantic similarity measure specifically designed to calculate semantic closeness between ontology concepts [18]. The notion of augmenting biological information with that in the gold standard corpus is explored with different combinations of input pipelines, hyperparameters, and model architectures.

In all these variants of GRU-based deep learning models, the ontologies are treated as independent concepts, thus assuming no association between them. However, in reality, these ontologies can have some semantic association and hence can not be treated as mutually exclusive concepts. In another study, we introduce a novel approach to incorporate the ontology hierarchy for training GRU-based models. This is achieved by creating a target vector as a label encoded vector or sometimes also known as a multi-hot encoded vector as opposed to a one-hot encoded vector in previous models.

In a one-hot encoded vector, each token corresponds to a row vector of k length, k representing the total number of ontology concepts in the corpus. Here, the value in the index of the ground truth concept is set to 1 and all other indexes have 0 value, therefore signifying mutual exclusiveness. In label encoded vector, however, the value in the index of the ground truth concept is set to 1 and all other indexes have values equal to the Jaccard similarity between the ground truth concept and the rest of the concepts in the corpus. Providing these vectors as target/output allows the model to learn the ontology hierarchy, therefore, making the model capable of predicting a semantically close ontology concept (such as a parent or a super-class) when it fails to make an exact prediction.

To further improve the accuracy of the ontology-aware model, we present another approach called Ontology-powered Boosting (OB) by using information about the ontology hierarchy to post-process the model's prediction. We "boost" all predictions where the model is relatively less confident about the predictions. Using the entropy and probability distribution information, we identify low-confidence predictions as potential candidates for boosting, and boost the prediction probability with the probabilities of semantically similar concepts. The goal of OB is to combine the model's preliminary predictions with knowledge of the ontology hierarchy to selectively increase the confidence of certain predictions to improve overall prediction accuracy. The method relies on a computationally inexpensive calculation and avoids bloated machine learning

models that cannot be trained or deployed without requiring enormous resources.

Chapter 2: Related Works

Substantial work has been conducted in the area of employing automated methods for identifying ontology annotations. The majority of this work is geared toward identifying Gene Ontology (GO) annotations since GO is the most widely used biological ontology. Some of the preliminary work in this space was aimed to assign GO terms to protein sequences and not to free text in literature.

Similarity-based approaches identify GO annotations based on the similarity between protein sequences [19–21]. When a sequence database is searched for a protein sequence, GO terms associated with similar sequences retrieved from the search are assigned to the query sequence. Probabilistic methods assume that the probability of shared GO functions is higher between proteins in close proximity on a protein interaction graph [22–26]. Markov Random Fields and Bayesian frameworks were used to determine the probability of shared GO functions in these approaches. Later, machine learning approaches such as Support Vector Machines were used to identify hidden relationships between protein features such as sequences, structure, etc. to annotate new proteins [27–30]. The latest developments in this area employ deep learning models for the task of automatically annotating proteins with GO terms. Various supervised deep learning architectures like Long Short Term Memory (LSTM), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Gated Recurrent Units (GRU), and Bidirectional RNNs have been shown to perform well at this task.

The early use of automated concept annotation set the stage for more sophisticated problems such as associating ontology concepts to pieces of text from the scientific literature. The task of automatically annotating scientific literature with ontology concepts is the task of focus in our study. Preliminary studies in this area employed

the use of lexical, syntactical, and traditional machine learning [1]. In prior work, we presented a review of these approaches and conducted a performance comparison using a gold standard dataset [1]. Three concept recognition tools (MetaMap [31], NCBO Annotator [32], Textpresso [33] were compared [1].These methods can form generalizable associations between text and ontology concepts leading to improved accuracy. However, in more recent years, the state of the art has evolved to leverage deep learning models due to the promise of increased accuracy and speed [5–8]. Deep learning models use vector representations that enable them to capture dependencies and relationships between words using enriched representations of character and word embeddings from training data [34].

In early uses of deep learning for ontology annotation of text, CNNs combined with LSTMs were used [35]. The work provided a proof-of-concept for the use of deep learning for ontology annotation and showed improved performance over traditional, machine learning methods. Other studies conducted performance comparisons among deep learning models and found that CNNs with enhanced inputs such as character embeddings were particularly effective for biomedical named entity recognition [36].

In a previous study [16], we compared Gated Recurrent Units (GRUs), Long Short Term Memory (LSTM), Recurrent Neural Networks (RNNs), and Multi Layer Perceptrons (MLPs) and evaluated their performance on the CRAFT gold standard dataset. We also introduced a deep learning architecture that used multiple GRUs with a character+word based input. The model was compared to seven models from existing work using the CRAFT corpus as a gold standard. We used data from five ontologies in the CRAFT corpus as a Gold Standard to evaluate our model's performance. Results showed that our GRU-based model outperformed prior models across all five ontologies. These findings indicated that deep learning algorithms are a promising avenue to be explored for automated ontology-based curation of data. This study also served as a formal comparison and guideline for building and selecting deep learning models and architectures for ontology-based curation.

This work was limited to predicting unigram annotations and did not take into account the rich semantic information in ontology hierarchies. In 2020, we presented new architectures based on GRUs and LSTM combined with different input encoding

formats for automated Named Entity Recognition (NER) of ontology concepts from text. We found that GRU-based models outperform LSTM models across all evaluation metrics. We also created multi-level deep learning models designed to incorporate ontology hierarchy into the prediction. Surprisingly, the inclusion of ontology semantics via subsumption reasoning yielded modest performance improvement [9]. This result indicated that more sophisticated approaches to take advantage of the ontology hierarchy are needed.

Most recent publications in this area have separated the ontology annotation task into two sub-tasks - 1) span detection: detecting the part of the text that corresponds to an ontology concept, and 2) concept normalization: identifying the ontology concept most appropriate for the identified piece of text [37, 38]. Using the CRAFT corpus as a training set, the study reports that Bidirectional encoder representations from transformers for biomedical text mining (BioBERT) resulted in the best performance (0.81 F1) for the span detection sub-task. The Open-source toolkit for Neural Machine Translation (OpenNMT) yielded the best performance for concept normalization. Overall, their results suggest that their approach using BioBERT for span detection and OpenNMT for concept normalization achieved state-of-the-art performance for most ontologies in the CRAFT corpus while using substantially fewer computational resources.

Treating the ontology annotation task as a sequence-to-sequence problem, another study [39] compared the performance of an LSTM model with BERT. This study divided the ontology annotation task into span detection and named entity normalization (NEN). However, instead of treating the steps like a pipeline where the output for the first step feeds into the next, these steps are carried out independently and agreement between the predictions is examined. The work uses ontology pretraining using names and synonyms of concepts found in the ontology. This step enables the models to predict concepts that might not be seen in the training data. The pretraining is further combined with a rule-based dictionary-lookup system that directly queries concept names from the ontology. Results show that the pretraining and lookup systems improve performance. The study reports an F1 score of 0.84 using a bidirectional LSTM-based architecture. Note that this system currently cannot handle sophisticated annotation formats such as discontinuous and overlapping annotations as represented

in the CRAFT corpus.

The application of deep learning architectures such as RNNs and LSTMs has also been explored for the task of relationship extraction between ontology concepts [40]. Sousa et. al. discuss neural network models to perform text mining tasks on data structures such as ontologies. Biological annotation tools such as Textpresso [33] conduct automatic curation by automatically extracting ontological entities and the relations between them. Similarly, other studies have facilitated relation extraction by creating gold standard datasets such as BioRel [41]. BioRel is a large-scale dataset designed specifically for relation extraction problems using the Unified Medical Language System as the source. Self attentive networks have also been found to be promising and have been applied for identifying drug-drug interactions, protein-protein interactions, as well as for identifying relations between medical concepts [42].

Continuing the work in [9], a 2022 study [43] presented state-of-the-art deep learning architectures based on GRUs for annotating text with ontology concepts. We augmented the models with additional information sources including NCBI's BioThesauraus and Unified Medical Language System (UMLS) to augment information from CRAFT for increasing prediction accuracy. We demonstrated that augmenting the model with additional input pipelines can substantially enhance prediction performance.

Considering that our previous attempt at creating intelligent prediction systems that use the ontology hierarchy was not successful, we developed a different approach to providing the ontology as input to the deep learning model [9]. In 2022, we presented an intelligent annotation system [44] that uses the ontology hierarchy for training and predicting ontology concepts for pieces of text. Here, we used a vector of semantic similarity scores to the ground truth and all ancestors in the ontology to train the model. This representation allowed the model to identify the target GO term followed by "similar" GO terms that are partially accurate predictions. This output label representation also helped the model optimize the weights to target more than one prediction label. We showed that our ontology-aware models can result in 2% - 10% (depending upon the choice of embedding) improvements over a baseline model that doesn't use ontology hierarchies.

Chapter 3: Background, Data and Preprocessing [43]

3.1 Background

3.1.1 Ontology

Ontologies are formal ways or techniques to represent and share knowledge about a particular domain by modeling objects in the domain and their relationships [45]. Agreement on a particular ontological representation allows domain experts to use common vocabulary to describe, store and analyze data. [45]. They also facilitate in easier handling of knowledge computationally. Therefore, we can see an increasing trend in the adoption of ontologies in several domains, including biology, medicine, and bioinformatics. Bio-ontologies are ontologies defined to describe entities in the biomedical domain. Currently, there are an estimated 958 bio-ontologies containing more than 55 million annotations (ontological terms together with their descriptions and synonyms) (as of 1-20-22 from `https://bioportal.bioontology.org/`).

3.1.2 Gene Ontology

Among the different ontologies, Gene Ontology (GO) is the most comprehensive and widely used ontology concerning the function of genes [46]. The knowledge base is both human-readable and machine-readable and is a foundation for computational analysis of large-scale molecular biology and genetics experiments in biomedical research. Additionally, the corpus that we use for training and evaluating our model includes

the largest number of annotations that are made using the GO.

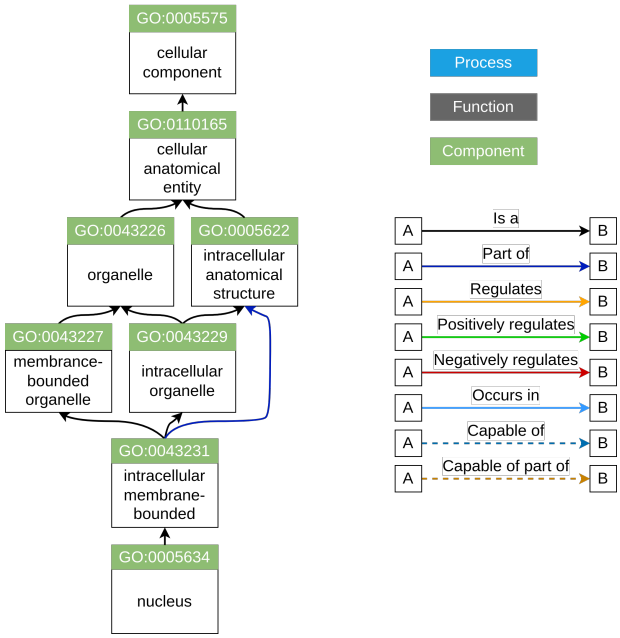


Figure 3.1. Example of a GO concept with the hierarchy `https://www.ebi.ac.uk/QuickGO/term/GO:0005634`

Figure 3.1 is an example of a GO concept - `GO:0005634` with its parents/ancestors. GO describes the knowledge of the biological domain with respect to three aspects: `Biological Process (BP)`, `Molecular Function (MF)`, and `Cellular Component (CC)`. The color of the concept represents its sub-domain within the GO ontology. The type and color of the directed edge from one ontology to another represent their relationship. GO is loosely hierarchical, meaning the child concept is more specialized than its parent concept or the parent concept is more generalized than its children. Additionally, a child can have more than one parent concept. All child concepts (leaf nodes) lead to one and exactly one of the three parent concepts (root node) (BP, MF, or CC), and the three root nodes are completely unrelated.

3.2 Dataset

This study uses version v4.0.1 (`https://github.com/UCDenver-ccp/CRAFT/releases/tag/v4.0.1`) of The Colorado Richly Annotated Full Text Corpus (CRAFT) [47], a

manually annotated corpus containing 97 articles each of which is annotated to 10 ontologies. All of the articles in the CRAFT corpus are part of the PubMed Central Open Access Subset. We selected GO annotations from the CRAFT corpus as our training and testing set.

3.3 Data Preprocessing

Each of the 97 articles in the CRAFT corpus has a corresponding `xml` annotation file which describes annotations within the sentences using character indexes of the article. The first step is to preprocess each annotation into a format that can be used by the deep learning models. All 97 articles are read as `UTF-8` encoded strings and the corresponding `xml` file for each article is parsed. The following preprocessing steps are performed to translate annotations from the CRAFT corpus to the desired input formats for the deep learning models.

3.3.1 Sentence segmentation and Tokenization

As mentioned earlier, annotations for each CRAFT article are recorded in the corresponding `xml` annotations file via character index spans. The following is an example of a sentence and its corresponding annotation:

Sentence: *"We observed a severe autosomal recessive movement disorder in mice used within our laboratory."*

Annotation:

```
<annotation>
    <mention id="GO_CC_2016_02_16_test_Instance_22573"/>
    <annotator id="GO_CC_2016_02_16_test_Instance_10000">
    Mike Bada, University of Colorado Anschutz Medical Campus
    </annotator>
    <span start="115" end="124"/>
    <spannedText>autosomal</spannedText>
</annotation>

<classMention id="GO_CC_2016_02_16_test_Instance_22573">
    <mentionClass id="GO:0030849">autosome</mentionClass>
```

```
</classMention>}
```

Here, the word *"autosomal"* with a character span of 115 - 124 is tagged to GO term "GO:0030849". To obtain annotations per word, we utilize a sentence segmentation library called `SpaCy` (`https://spacy.io/`). First, the segmenter splits the text into sentences by accounting for sentence end marks (such as periods, exclamation, question marks, etc.) and then uses a tokenizer to split the sentences into individual words (or tokens) by accounting for word boundaries (such as space, hyphen, tab, etc.). For example, the above sentence is split into individual tokens as follows:

Sentence: *"We observed a severe autosomal recessive movement disorder in mice used within our laboratory."*

Tokens: [*'We', 'observed', 'a', 'severe', 'autosomal', 'recessive', 'movement', 'disorder', 'in', 'mice', 'used', 'within', 'our', 'laboratory', '.'*]

Annotation: *{'start': 115, 'end': 124, 'spanned_text': 'autosomal', 'go_term': 'GO:0030849' }*

3.3.2 IOB Tagging

The deep learning models need to know if each word/token corresponds to a GO term. Each extracted word/token is mapped to a GO term or an *out-of-concept* annotation. Here we use the range specified in the `xml` to map the token to one of three tags: 1) `GO` to indicate an annotation, `'O'` for a non-annotation (out-of-concept), and `'EOS'` to indicate the end of a sentence. For example, the following sentence would be tagged as below:

Sentence: *"We observed a severe autosomal recessive movement disorder in mice used within our laboratory."*

Tokens: [*'We', 'observed', 'a', 'severe', 'autosomal', 'recessive', 'movement', 'disorder', 'in', 'mice', 'used', 'within', 'our', 'laboratory', '.'*]

IOB Tags: `['O', 'O', 'O', 'O', 'GO:0030849', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'EOS']`

The above example shows a simple case where a single word is annotated to a GO concept. In other cases, a sequence of words/tokens is annotated to a GO term. We utilize the IOB (Inside, Outside, Beginning) [48] standard for annotating multi-span tokens to account for such annotations. The IOB format uses three prefixes to tag tokens in a sentence: 1) `'B-GO'` is used to specify the beginning of the annotation, 2) `'I-GO'` is used to map the tokens following the beginning of annotation till the end, and 3) `'O'` is used to map tokens that don't correspond to a GO term. The following sentence shows an example of IOB formatting:

Sentence: *"The phosphatidylserine receptor primarily functions in apoptotic cell clearance."*

Annotation: *{'start': 1862, 'end': 1886, 'spanned_text': 'apoptotic cell clearance', 'go_term': 'GO:0043277'}]*

Tokens: [*'The', 'phosphatidylserine', 'receptor', 'primarily', 'functions', 'in', 'apoptotic', 'cell', 'clearance', '.'*]

IOB Tags: `['O', 'O', 'O', 'O', 'O', 'O', 'B-GO:0043277', 'I-GO:0043277', 'I-GO:0043277', 'EOS']`

In the above example, the phrase *"apoptotic cell clearance"* is annotated to `GO:0043277`. We tag the token 'apoptotic' with `B-GO:0043277` indicating the beginning of the annotation. The tokens 'cell' and 'clearance' are tagged with `I-GO:0043277` indicating the continuation of the annotation. `O` is used to map the rest of the tokens which do not correspond to any annotations and `EOS` is used to map *'.'* signifying the end of the sentence.

3.3.3 Annotation Formats

Sentences in the CRAFT corpus are annotated following a set of annotation formats and guidelines as detailed in `https://github.com/UCDenver-ccp/CRAFT/tree/master/concept-annotation`. Below, we describe how sentences that contain annotations in different formats are represented in the IOB format.

- **No annotations**: Some sentences in an article might not contain any anno-

tations. In this case, all tokens are represented by `'O'` tags except the ending character which is represented by `'EOS'` tag.

- **Disjoint annotations:** A sentence might contain one or more annotations that don't overlap in terms of annotation span. In this case, all tokens not corresponding to an annotation are tagged with **O** tags. The end of the sentence character is represented by **EOS** tag. Tokens that mark the beginning of an annotation are marked with a **B-GO:term** followed by **I-GO:term** to represent subsequent tokens corresponding to an annotated phrase.

- **Overlapping annotations:** A sentence might contain a phrase (sequence of words/tokens) that is annotated to a GO concept, and a word or a sub-phrase within the original phrase that is annotated to a different GO concept.

  In these instances, we make n copies of the sentence where n is the number of different annotations. Each copy contains a modified sentence that represents the text needed to convey one of the annotations.

  If a sentence contains a case of overlapping annotations and other disjoint annotations (non-overlapping annotations), we create sentences that capture the different variations of the overlapping annotations while keeping the disjoint annotations common.

- **Multiple overlapping annotations**: Sentences can also have more than one phrase with sub-annotations. In such a case, where there exist m phrases with $n_1, n_2, \cdots, n_m$ overlapping subphases, there will $n_1 \times n_2 \times ... \times n_m$ copies with all possible combinations of sub-phrase mappings.

Below we present examples of each of these annotation types and describe how they are processed for training the models:

- **No annotations**:

  Sentence: *"Rescue of Progeria in Trichothiodystrophy by Homozygous Lethal Xpd Alleles"*

  Annotations: {None}

IOB Tags: [‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘EOS’]

- **Disjoint annotations:**

  Sentence: *“A cell progressing from anaphase to cytokinesis (pink arrowheads).”*

  Annotations: {*‘anaphase’* — GO:0051322; ‘cytokinesis’ — GO:0000910}

  IOB Tags: [‘O’, ‘O’, ‘O’, ‘O’, ‘B-GO:0051322’, ‘O’, ‘B-GO:0000910’, ‘O’, ‘O’, ‘O’, ‘O’, ‘EOS’]

- **Overlapping annotations:**

  Sentence: *“Having excluded a direct role in **vesicle formation** and membrane fusion, annexin A7 might act by its property as Ca2+-binding protein”*

  Annotations: {*‘vesicle’* — GO:0031982; ‘vesicle formation’ — GO:0006900}

  The above example is represented as two sentences with each sentence representing one of the two annotations.

  Sentence 1: *“Having excluded a direct role in **vesicle** and membrane fusion, annexin A7 might act by its property as Ca2+-binding protein"*

  Annotations: {*‘vesicle’* — GO:0031982}

  IOB Tags: [‘O’, ‘O’, ‘O’, ‘O’,‘O’,‘O’, ‘B-GO:0031982’ ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘EOS’]

  Sentence 2: *“Having excluded a direct role in **vesicle formation** and membrane fusion, annexin A7 might act by its property as Ca2+-binding protein”*

  Annotations: {*‘vesicle formation’* – GO:0006900}

  IOB Tags: [‘O’, ‘O’, ‘O’, ‘O’,‘O’,‘O’, ‘B-GO:0006900’, ‘I-GO:0006900’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘O’, ‘EOS’]

- **Multiple overlapping annotations**:

Sentence: *"Having excluded a direct role in* ***vesicle formation*** *and* ***membrane fusion****, annexin A7 might act by its property as Ca2+-binding protein."*

Annotations: {*'vesicle'* — `GO:0031982`; *'vesicle formation'* — `GO:0006900`; *'membrane'* — `GO:0016020`; *'membrane fusion'* — `GO:0061025`}

In this example, we have two instances of overlapping annotations with two sub-phrase annotations each. This sentence would be transformed into four sentences that each represent a unique combination of annotations.

Sentence 1: *"Having excluded a direct role in* ***vesicle*** *and* ***membrane****, annexin A7 might act by its property as Ca2+-binding protein."*

Annotations: {*'vesicle'* — `GO:0031982`; *'membrane'* — `GO:0016020`}

IOB Tags: `['O', 'O', 'O', 'O','O','O', 'B-GO:0031982' 'O', 'B-GO:0016020','O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'EOS']`

Sentence 2: *"Having excluded a direct role in* ***vesicle formation*** *and* ***membrane****, annexin A7 might act by its property as Ca2+-binding protein."*

Annotations: {*'vesicle formation'* — `GO:0006900`; *'membrane'* — `GO:0016020`}

IOB Tags: `['O', 'O', 'O', 'O','O','O', 'B-GO:0006900', 'I-GO:0006900', 'O','B-GO:0016020', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O','O', 'EOS' ]`

Sentence 3: *"Having excluded a direct role in* ***vesicle*** *and* ***membrane fusion****, annexin A7 might act by its property as Ca2+-binding protein."*

Annotations: {*'vesicle'* — `GO:0031982`; *'membrane fusion'* — `GO:0061025`}

IOB Tags: `['O', 'O', 'O', 'O', 'O', 'O', 'B-GO:0031982', 'O', 'B-GO:0016025', 'I-GO:0016025', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'EOS']`

Sentence 4: *"Having excluded a direct role in **vesicle formation** and **membrane fusion**, annexin A7 might act by its property as Ca2+-binding protein."*

Annotations: {*'vesicle formation'* — `GO:0006900`;
'membrane fusion' — `GO:0061025`}

IOB Tags: `['O', 'O', 'O', 'O', 'O', 'O', 'B-GO:0006900', 'I-GO:0006900', 'B-GO:0016025','I-GO:0016025', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'EOS']`

- **Discontinuous annotations**:

  Sentence: *"Because the F7 is the most severely affected allele, it is possible that the difference between the heart and kidney levels is due to a developmental delay in **v/p formation**."*

  Annotations: *"v formation"* — `GO:0097084`

  Here we see *"v formation"* is annotated to `GO:0097084`, whereas *"/p"* is not. In such a case we represent the sentence by removing the tokens/words which were not annotated (*"/p"*). This is done to represent the continuous span of the phrase to GO term mapping.

  Transformed Sentence: *"Because the F7 is the most severely affected allele, it is possible that the difference between the heart and kidney levels is due to a developmental delay in **v formation**."*

  IOB tags: `['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-GO:0097084', 'I-GO:0097084', 'EOS']`

We acknowledge the representation of discontinuous annotations is not ideal. However given that the majority of annotations in CRAFT are continuous, we prioritized the data to follow the same pattern. Some sentences might have a combination of disjoint, overlapping, and/or discontinuous annotations. These sentences are broken down into smaller cases with precedence in the order of - overlapping, discontinuous, and disjoint annotations. If there are overlapping annotations, they are treated first i.e., multiple copies of the sentence are created and mapped for their annotations. Then for each

copy, the discontinuous annotations are handled while keeping the disjoint annotations common between the representations.

While creating multiple copies of the sentences can lead to over-sampling of such cases, the overall number of such sentences was very low in comparison to the total number of sentences present in the training data. Furthermore, this is only done in the training dataset, where the validation data is preprocessed separately leading to more robust metrics presented in the results.

3.3.4 POS Tagging and Token Encoding

Following the tokenization and IOB tagging, we enrich training data with parts-of-speech (POS) information and a compressed character representation. POS tagging looks at the contextual information of the word based on the words surrounding it in a sentence or a phrase. Here we used the `SpaCy` POS tagger to evaluate and tag the tokens of sentences with 15 parts of speech tags — adjective, adposition (such as - in, to, during), adverb, auxiliary (such as - is, has done, will do, should do), conjunction, coordinating conjunction, determiner, interjection, noun, numeral, particle, pronoun, proper noun, punctuation, subordinating conjunction, symbol, verb, other (not annotated to any of the others), space.

While POS tagging looks at the word-level representation of the context, we also represent character-level nuances of a token using character encodings. These encodings represent upper-case and lower-case characters with 'C' and 'c' respectively. Numbers are represented using an 'N' and punctuation (such as commas, and periods) is retained in the encoding. Character encodings enable a succinct representation of a token's unique characters which can indicate named entities and aid in the model's learning.

Here we show an example of a sentence tagged with POS and character representations.

Sentence: *"Smith-Lemli-Opitz syndrome (SLOS, MIM 270400), a relative common dysmorphology disorder, is caused by mutations in DHCR7 [2-5], which encodes for 7-dehydrocholesterol ▽7-reductase and catalyzes a final step of cholesterol biosynthesis."*

Character Representation: `['Ccc-Ccc-Ccc', 'ccc', '(', 'CCC', ',', 'CCC', 'N', ')', ',', 'c', 'ccc', 'ccc', 'ccc', 'ccc', ',', 'cc', 'ccc', 'cc',`

‘ccc’, ‘cc’, ‘CCCN’, ‘[’, ‘N-N’, ‘]’, ‘,’, ‘ccc’, ‘ccc’, ‘ccc’, ‘N-ccc’, ‘U’, ‘ccc’, ‘ccc’, ‘c’, ‘ccc’, ‘ccc’, ‘cc’, ‘ccc’, ‘ccc’, ‘.’]

Parts-of-Speech: [‘NNP’, ‘NN’, ‘-LRB-’, ‘NNP’, ‘,’, ‘NNP’, ‘CD’, ‘,’, ‘,’, ‘DT’, ‘JJ’, ‘JJ’, ‘NN’, ‘NN’, ‘,’, ‘VBZ’, ‘VBN’, ‘IN’, ‘NNS’, ‘IN’, ‘NNP’, ‘XX’, ‘CD’, ‘,’, ‘,’, ‘WDT’, ‘VBZ’, ‘IN’, ‘NN’, ‘NN’, ‘CC’, ‘VBZ’, ‘DT’, ‘JJ’, ‘NN’, ‘IN’, ‘NN’, ‘NN’, ‘.’]

3.3.5 BioThesaurus Encoding

In addition to POS and token encoding, which capture sentence and token level context present in the data, we also include information from existing large-scale knowledge bases. The first data source we use is BioThesaurus [13], which is a database of protein and gene names mapped to the UniProt Knowledgebase. The database contains over 2.8 million names/tokens from separate data sources and is well regarded as a comprehensive thesaurus mapping words to their molecular/biological entities. We query BioThesaurus for each of the tokens extracted from the articles. First, we map if a token is present (`1`) or absent (`0`) in the database. If a token is present, we map if it identifies as a protein name, biomedical terms, chemical terms, and/or macromolecule. Sometimes, a token can be identified in multiple categories. In the following example we show the mapping of a token as queried from the BioThesaurus:

Sentence: *“Hematopoiesis is precisely orchestrated by lineage-specific DNA-binding proteins that regulate transcription in concert with coactivators and corepressors.”*

Tokens: [‘Hematopoiesis’, ‘is’, ‘precisely’, ‘orchestrated’, ‘by’, ‘lineage-specific’, ‘DNA-binding’, ‘proteins’, ‘that’, ‘regulate’, ‘transcription’, ‘in’, ‘concert’, ‘with’, ‘coactivators’, ‘and’, ‘corepressors’, ‘.’]

Protein: `[0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0]`

Biomedical: `[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]`

Chemical: `[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`

Macromolecule: `[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`

3.3.6 Unified Medical Language System (UMLS) Encoding

Continuing with the information augmentation, we also query the UMLS [14] database for tokens extracted from the articles. UMLS is another comprehensive database of over 2 million names representing medical and bio-medical terms aggregated from several databases such as NCBI taxonomy, Gene Ontology, the Medical Subject Headings (MeSH), OMIM, ICD-10-CM, SNOMED CT, and the Digital Anatomist Symbolic Knowledge Base.

Here we query the metathesaurus component of the database for the extracted tokens. Words/tokens associated with a UMLS term are encoded as `1` or `0` otherwise. If a phrase (sequence of tokens) is found in UMLS, all tokens from the phrase are encoded as `1` Below we show an example of the mapping:

Sentence: *"Hematopoiesis is precisely orchestrated by lineage-specific DNA-binding proteins that regulate transcription in concert with coactivators and corepressors."*

Tokens: ['Hematopoiesis', 'is', 'precisely', 'orchestrated', 'by', 'lineage-specific', 'DNA-binding', 'proteins', 'that', 'regulate', 'transcription', 'in', 'concert', 'with', 'coactivators', 'and', 'corepressors', '.']

UMLS: `[1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0]`

Before preprocessing, CRAFT articles are divided using an 80-20 split to create training and testing data. Training and testing data are then processed into sentences, tokenized, translated into different annotation formats, and encoded using BioThesaurus and UMLS. The training data is used for the development of the deep learning models (described in the following section). Testing data is used to evaluate model performance.

Here is an example of a sentence that is preprocessed using the steps described above. It is extracted from a `json` file corresponding to an article. The sentence extracted from the article is positioned between index 27731 and 27769. Each of the information pertaining to the sentence is stored as key-value pair where the key stores the type of information and the value stores the values corresponding to the key.

```
{
  "annot":[
      { "start":  27750, "end":  27760, "spanned_text":  "expression",
        "go_term":  "GO:0010467"},
      { "start":  27764, "end":  27768, "spanned_text":  "RNAi",
        "go_term":  "GO:0016246"},
  ],
  "text_span":[27731, 27769 ],
  "main":  "Knockdown of Alms1 expression by RNAi.",
  "tokens":["Knockdown", "of", "Alms1", "expression", "by", "RNAi",
            "."],
  "iob_tags":["O", "O", "O", "B-GO:0010467", "O", "B-GO:0016246",
              "EOS"],
  "pos_tags":["NN", "IN", "NNP", "NN", "IN", "NNP", "."],
  "encodings":["Ccc", "cc", "CccN", "ccc", "cc", "CCc", "."],
  "protein_terms":[0, 0, 1, 0, 1, 0, 0],
  "biomedical_terms":[0, 0, 0, 1, 0, 0, 0],
  "chemical_terms":[0, 0, 0, 0, 0, 0, 0],
  "macromolecule_terms":[0, 0, 0, 0, 0, 0, 0],
  "umls_terms":[0, 0, 0, 0, 0, 1, 0],
}
```

Chapter 4: GRU based architectures for concept recognition [43]

Ontologies have become the de-facto mode of representing biological knowledge since the development of the Gene Ontology (GO) [49]. Following the widespread adoption of the GO, other bio-ontologies representing knowledge in disparate aspects of biology and biomedicine have been created. Today, an estimated 958 bio-ontologies are in use spanning over 55 million annotations (as of 1-20-22 from `https://bioportal.bioontology.org/`). While the use of bio-ontologies and the number of annotations created using these ontologies have grown exponentially, the methods used to create these annotations haven't changed at a comparable pace. The majority of ontology annotations are still created via manual curation - the process where a human curator reads scientific literature and manually selects appropriate ontology concepts to describe phrases/words in the text. The process of manual creation is slow, tedious, and unscalable to the rapid pace of scientific publishing [2].

Over the past decade, text mining approaches have been developed to conduct ontology annotation in an automated manner. Preliminary solutions include syntactic, lexical approaches followed by traditional machine learning applications [1]. Lexical solutions for automated ontology annotation rely on similarities between a piece of text and the name of an ontology concept or their synonym to assign annotations [3]. This approach can be challenging when the text does not match the names of ontology concepts. Also, some ontology concept names contain a large number of words which makes text matching difficult [3].

Text mining tools that use machine learning-based methods employed supervised

learning techniques using gold standard corpora [1]. These methods can form generalizable associations between text and ontology concepts leading to improved accuracy. The rise of deep learning in the areas of image and speech recognition has translated into text-based problems as well. Preliminary research has shown that deep learning methods result in greater accuracy for text-based tasks including identifying ontology concepts in text [5–9]. Deep learning methods use vector representations that enable them to capture dependencies and relationships between words using enriched representations of character and word embeddings from training data [34].

The semantic complexities of identifying the appropriate ontology concept for a word/phrase are quite challenging. In the simplest case, the name of the ontology concept is an exact match to the piece of text. For example, the phrase "brain development" in the sentence "HOMER proteins have also been implicated in axon guidance during brain development" is annotated to the GO term "brain development (GO:0007420)". Sometimes, a match can also be made by comparing the text to the names of known synonyms of concepts in the ontology. In most cases, there aren't clear matches between the words being annotated to the names of the ontology concepts. For example, the word "olfactory" in the sentence "Class I olfactory receptors are bracketed, and the remaining olfactory receptors are class II." is annotated to the GO term "sensory perception of smell (GO:0007608)". 80% of the annotations made in the latest version of the CRAFT corpus has no clear match between the text and the name of the ontology concept used for annotation. This is a clear indication of the complexity of the problem at hand, one that cannot be solved just by syntactic methods or by text matching. These are the cases where effective training can make a substantial difference.

Training deep learning models requires good quality training datasets. The Colorado Richly Annotated Full Text Corpus (CRAFT) [47] is a widely used training resource for automated annotation approaches. The current version of the CRAFT corpus (v4.0.1) provides annotations for 97 biological/biomedical articles with concepts from 7 ontologies including the GO. CRAFT uses several formats with different levels of complexity to represent annotations.

One of the challenges in creating effective deep learning models is translating all

of CRAFT's annotations to formats that can be leveraged by the models. This process involves a substantial amount of preprocessing that's designed specifically for each annotation format to ensure that each annotation is represented soundly and completely in the training data. Another challenge when using machine learning solutions - including deep learning models is the availability and abundance of training data. Not all concepts in the ontology are represented in the gold standard corpus hindering the ability of the trained models to recognize those unseen concepts. Among the concepts that are present in the training data, some of them occur frequently while others are sparse. It might be necessary to augment the primary training corpus with information from other sources to improve prediction accuracy.

The choice of deep learning model and architecture also impacts prediction performance. We have conducted comparisons of models such as CNNs, GRUs, LSTMs, and RNNs in previous work [9,16] whose findings enable us to make informed choices in this study. Here, we present a deep learning architecture that leverages inputs from multiple sources and in different formats (characters, words, etc.) to improve on the state-of-art in terms of prediction performance.

We make two contributions in this study - 1) publicly available preprocessed annotations from the CRAFT corpus for training deep learning models and 2) deep learning architectures for identifying ontology concepts.

4.1 Methods

4.1.1 Deep Learning Architecture

After all the preprocessing steps described above are complete, we develop multidimensional vectors for each sentence of the articles. Our deep learning architecture (Figure 4.1) consists of three key components — 1) Input Pipelines; 2) Embedding/Latent Representations; and 3) Sequence Modeler. Below we describe each of the components:

Input Pipelines

The recurrent neural architecture used in our approach requires fixed-size inputs. Accordingly, we restrict each sentence to contain a maximum of 71 words/tokens. This

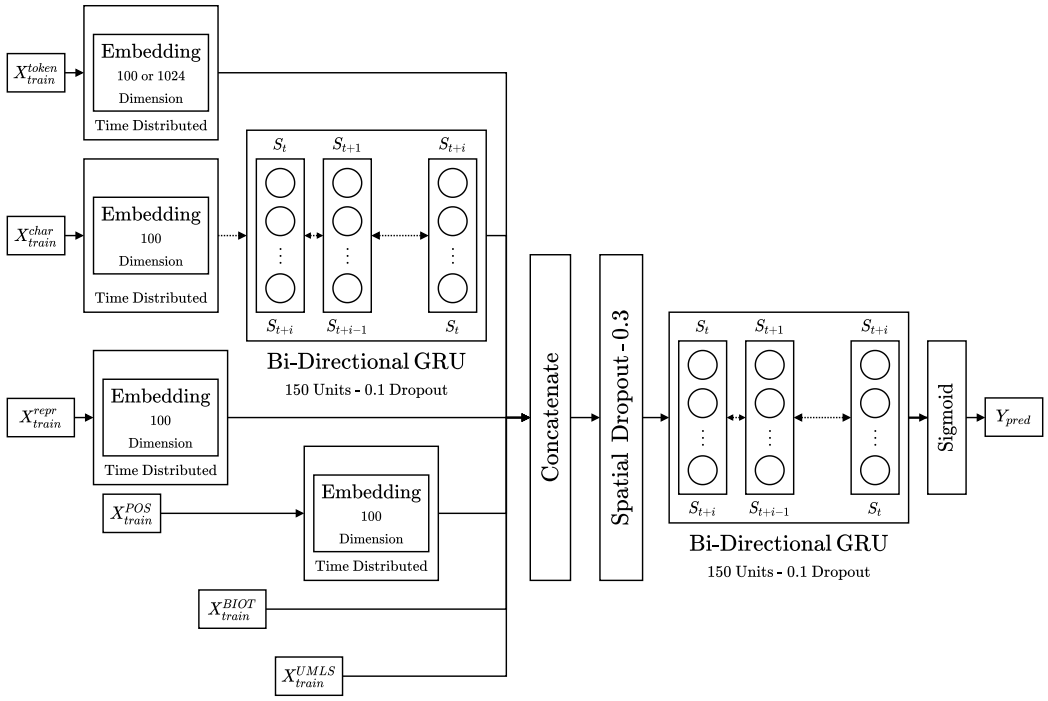


Figure 4.1. Architecture of a GRU model using multiple input pipelines

is based on the third standard deviation of the distribution of the frequency of words present in sentences. Sentences with a lower number of words are padded with the token `<PAD>` and ones with a higher number of tokens are truncated to a length of 71. All corresponding input vectors are also adjusted accordingly to reflect the maximum sequence length representation of a sentence.

Each sentence and each token has six different components that are provided as input — 1) token (X^{token}_{train}), 2) character sequence (X^{char}_{train}), 3) token-character representation (X^{repr}_{train}), 4) parts-of-speech (X^{POS}_{train}), 5) BioThesaurus (X^{BIOT}_{train}), and 6) UMLS (X^{UMLS}_{train}).

The token (X^{token}_{train}) input, is a sequential tensor consisting of 71 tokens, where each token is represented with a high dimensional one hot encoded vector (for 34,164 unique words/tokens present within our corpus vocabulary). Apart from the `<PAD>` token, we also use `<UNK>` token to represent unknown tokens. This is done to generalize the model for words that were not available in the training data but can be present in the testing dataset. Similarly, the character sequence (X^{char}_{train}) is also a sequential

tensor consisting of character sequences present in a word/token. Here, we limit the maximum character length to 15 based on the third standard deviation of the character distribution. Tokens with longer sequences are truncated and tokens with shorter are padded with a `<PAD>` character identifier. A single input sentence tensor for X_{train}^{char} has a shape of (1,71,15), for 71 tokens and 15 characters.

Next we provide character representations (X_{train}^{repr}) and POS tags (X_{train}^{POS}). Both of these are based on words/tokens in sentences and are given as an input of 71 vectors. Biothesaurus encodings (X_{train}^{BIOT}) contain a four-dimensional vector sequence where each token is one hot encoded for its association with protein, biomedical, chemical, and macromolecule categories. UMLS encodings (X_{train}^{UMLS}) are also provided as one hot encoded vector sequence, where `1` indicates a token's presence and `0` indicates absence in UMLS.

Embedding/Latent Representations

Our architecture utilizes embeddings to provide a compressed latent space representation for very high dimensional input components. For example, the one hot vectorization of an individual word has a dimensionality of 34,166. To represent them succinctly and with contextual representation, we evaluated three different approaches for embeddings — 1) supervised embedding layer, 2) GloVe layer, and 3) ELMo layer.

The supervised embedding is a bottleneck layer that learns to map the one hot encoded input into a smaller dimensional representation. The weights of this layer are learned from the backpropagation of losses based on the final output of the model. The resulting embedding learns the mapping of the IOB tags to the tokens of the sentences. The layer is used with token inputs (X_{train}^{token}), character sequences (X_{train}^{char}), and character representation (X_{train}^{repr}), each of which has very high dimensionality in their original vectors. We utilize a 100-dimensional output representation for each of the aforementioned outputs, where weights are uniformly initialized at the start of the model training.

We also evaluate GloVe [50] and ELMo [51] pretrained embeddings for the X_{train}^{token} input. Both are unsupervised approaches towards learning contextual representation of words from large-scale corpora. GloVe uses word co-occurrence statistics to learn

the embeddings. Pretrained data from cased Common Crawl with 840B tokens, 2.2M vocabulary, and a 300-dimensional output embedding vector is used for this. In comparison, the embeddings in ELMo are learned via a bidirectional language model where the sequence of the words is also taken into account. We use the pretrained model on 1 Billion Word Benchmark, which consists of approximately 800M tokens of news crawl data and has an embedding of 1024 dimensional output embedding vector. While the embeddings are initialized from pretrained models, we allow for updates/retraining to the embedding models during the training of our larger model.

Sequence Modeler

To model the input sequences we utilize a deep bi-directional gated recurrent model (Bi-GRU). Bi-GRU was first proposed by Cho et. al. [15] as a more efficient approach than Long -Short Term Memory (LSTM) [17] while being able to tackle the vanishing gradient problem of vanilla Recurrent Neural Networks (RNN). The approach uses a gated mechanism to decide what information needs to be transmitted to the output of a single unit.

In our prior work [9, 16], we evaluated multiple models based on RNN, LSTM, and GRU, and concluded that the GRU-based architecture performed the best on CRAFT v2 annotation data. Building on that result, we employ the Bi-GRU as the base of our architecture in this work. As shown in Figure 4.1, we utilize Bi-GRUs in two locations in the architecture, first to model the sequence of characters present in each token and second as the main Bi-GRU model to concatenate input pipelines together. After the embedding of the characters, they are passed via the first Bi-GRU (consisting of 150 units) resulting in a sequence representation of the characters in a sentence. 10% dropout is used in this pipeline to regularize the output to prevent overfitting.

The character sequence representation is then concatenated with other embeddings, i.e. token (supervised/GloVe/ELMo), character representation, parts of speech, and input tensors from Bio-Thesaurus and UMLS. This concatenated feature map representing each sentence is then passed to a spatial dropout, which removes 30% of the 1-D sequence features from the input to the main Bi-GRU. The main Bi-GRU processes the feature maps (with 10% dropout), and outputs to a single time-distributed dense

layer of 1774 nodes (representing each of the output tags). A sigmoid activation is used in the last layer, where the final prediction is based on the highest probability value of the tags. There are 6 hidden layers and thus a total of 8 layers (including input and output layers) in our models. However, not all the input pipelines go through all the layers. For instance, UMLS and PROT terms do not pass through the embedding layer.

Figure 4.2 shows a snapshot of the model architecture in the context of training and inference of a sample set of tokens. Here we show the training/inference on a sequence of tokens "vesicle", "formation", and "in" (which are parts of a sentence) as it is evaluated by the network. Each token is preprocessed to obtain the representative tensors – X^{token}_{train}, X^{char}_{train}, X^{repr}_{train}, X^{POS}_{train}, X^{BIOT}_{train}, and X^{UMLS}_{train}. X^{token}_{train}, X^{char}_{train}, X^{repr}_{train}, and X^{POS}_{train} which are passed through embedding layers, where the embedding of X^{token}_{train} can be a complete pretrained architecture such as GloVe or ELMo.

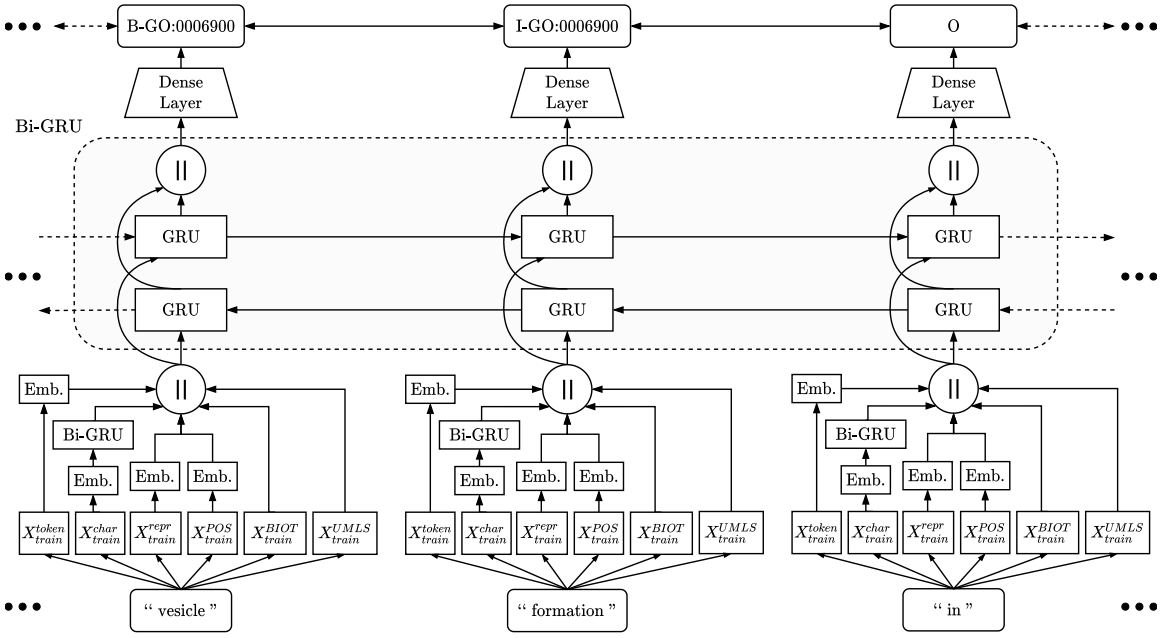


Figure 4.2. Workings of a GRU model with an example input sequence

The embedding of X^{char}_{train} is also passed via a Bi-directional GRU (Bi-GRU) layer. All of the resulting values are concatenated to be processed via the main Bi-GRU layer. Here we show each direction of the GRU layers as they process the input sequence. The

first layer processes the sequence in its left to right ordering, i.e. “vesicle”, “formation”, and “in”, whereas the second layer processes the reverse, i.e. “in”, “formation”, and “vesicle”. The bi-directionality allows the architecture to learn the preceding and succeeding sequence patterns within the sequence tokens in a sentence. The states of both the GRU layers are then concatenated to deliver to the final dense layer, which is the sigmoid classifier of the architecture that predicts the associated IOB tags for the input tokens. Here we select the tag with the highest probability for each of the tokens.

We evaluated the impact of including each pipeline and token embedding approach to create nine different models that differ in the inputs pipelines provided to them. We evaluated three embeddings (CRAFT, GloVe, ELMo) in conjunction with these models to result in a total of 28 experiments. Architecture hyper-parameters, which include — supervised embedding shape ({20, 50, 100, 150, 200}), dropout ({01, .2, .3, .5, .7}), number of epochs ({50, 100, 200, 300}), and class weighting, were evaluated using a grid search approach. We used Adam [52] as our optimizer for all of the experiments with a default learning rate of 0.0001. Learning rate reduction on a plateau of loss was used, which reduced the rate by a factor of 0.1 if the loss stayed constant for 4 epochs. A batch size of 16 was used in all of our experiments.

Bidirectional Encoder Representations from Transformers (BERT) [53] is a popular attention model developed by Google. BERT has rapidly become the state of the art in several applications, especially those involving text processing. Instead of looking at a text sequence in one direction, BERT uses bidirectional training which allows it to build better representations and context of textual inputs. The classic version of BERT was pretrained on a large corpus of English data. SciBERT, a variant of BERT, is trained on a large multi-domain corpus of scientific literature to improve performance on the prediction of scientific entities. We compared the best model from our experiments with both versions of BERT.

4.1.2 Performance Evaluation Metrics

The performance of each experiment is evaluated using a modified F1 score. The model is tasked with predicting non-annotations (indicated by an `'O'` tag) or annotations

(indicated by a 'GO' tag). Since the majority of tags in the training corpus are non-annotations, the model predicts them with great accuracy. To avoid biasing the F1 score, we omit accurate predictions of 'O' tags from the calculation to report a relatively conservative F1 score.

F1 quantifies whether the model's prediction matches the actual annotation exactly. However, ontology-based prediction systems need to be evaluated while accommodating partially accurate predictions. For example, a model might not retrieve the exact ontology concept as the gold standard but a related concept (sub-class or super-class) achieving partial accuracy. Semantic similarity metrics [18] designed to measure different degrees of similarity between ontology concepts can be leveraged to measure the similarity between the predicted concept and the actual annotation to quantify the partial prediction accuracy. Here, we use Jaccard similarity [18] that measures the ontological distance between two concepts to assess partial similarity.

4.2 Results and Discussion

The CRAFT v4.0.1 dataset contains 18689 annotations pertaining to 974 concepts from the three GO sub-ontologies across 97 articles. Table 4.1 provides further information on the coverage of GO terms in CRAFT.

Table 4.1. Coverage of GO ontology concepts and annotations in the CRAFT corpus

GO sub-ontology	Concepts in ontology	Total annotations in CRAFT	Unique occurences in CRAFT
Biological Process (BP)	30490	18392	710
Cellular Component (CC)	4463	6976	241
Molecular Function (MF)	12257	464	5

Table 4.2 shows the performance scores for Models 1 through 9 (M_1 - M_9) which differ in the inputs provided to them. M_1 is built with only tokens and no other inputs. Gradually, we add characters, character representation, parts of speech, and other inputs in each subsequent model. Each model is tested with three embeddings (CRAFT, GloVe, and ELMo). F1 and Jaccard semantic similarity are used to evaluate the models.

The base model with only tokens as input results in an F1 score in the range of 0.78 (for the CRAFT embedding) to 0.81 (GloVe and ELMo) and a semantic similarity of 0.81 to 0.82. The higher semantic similarity indicates that there are instances where the model misses the exact annotation in the gold standard yet predicts a partially related concept. These instances are captured and accounted for in the semantic similarity metric via partial credit whereas they receive a score of 0 in the F1 calculation.

Table 4.2. Performance comparison of nine GRU models with different input pipelines. Models are evaluated using F1 and semantic similarity. Each model includes certain inputs (listed as column headers). When a particular input type is included, there is a ✓in the corresponding cell.

Model	Input Pipelines									Embeddings					
	X_{test}^{token}	X_{test}^{char}	X_{test}^{repr}	X_{test}^{POS}	X_{test}^{BIOT}				X_{test}^{UMLS}	CRAFT		GloVe		ELMo	
					Prot.	Biom.	Chem.	Macr.		F1	Sem.	F1	Sem.	F1	Sem.
M_1	✓									0.78	0.79	0.82	0.83	0.81	0.81
M_2	✓	✓								0.79	0.80	0.82	0.83	0.82	0.83
M_3	✓	✓	✓							0.80	0.81	0.82	0.83	0.81	0.81
M_4	✓	✓	✓	✓						0.79	0.80	0.82	0.83	0.82	0.83
M_5	✓	✓	✓	✓	✓					0.81	0.82	0.81	0.82	0.84	0.84
M_6	✓	✓	✓	✓	✓	✓				0.79	0.80	0.82	0.83	0.83	0.83
M_7	✓	✓	✓	✓	✓	✓	✓			0.81	0.82	0.82	0.84	0.84	0.84
M_8	✓	✓	✓	✓	✓	✓	✓	✓		0.80	0.81	0.82	0.83	0.83	0.84
M_9	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.80	0.81	0.82	0.83	0.84	0.84

Adding character sequences (M_2) improves F1 and semantic similarity scores across almost all embeddings. Adding token-character representation (M_3) yields mixed results. We see an improvement in F1 and Semantic similarity for the CRAFT embedding. However, both scores stay unchanged with GloVe and decrease with ELMo. The inclusion of parts of speech (M_4) causes a decrease in scores with CRAFT and ELMo. Both scores remain unchanged with GloVe. Providing protein names from BioThesaurus (M_5) improves both scores for CRAFT and ELMo while we observe a decrease in Glove. Here, we observe the highest F1 (0.84) and semantic similarity (0.84) across all models tested so far. M_6 - M_9 yield comparable results but do not result in further improvements over M_5. In summary, our best model resulted in an F1 score and semantic similarity score of 0.84 with the ELMo embedding.

Table 4.3. Modified F1 scores from model M_9 broken down by GO sub-ontology

Model	GO_BP	GO_MF	GO_CC
M_9	0.82	0.96	0.85

Table 4.4. Confusion matrix for predictions by GO sub-ontology (Biological Process (GO_BP), Cellular Component (GO_CC), and Molecular Function (GO_MF). Note that this matrix does not include accurately predicted 'O' and 'EOS' tags since these instances are omitted during the calculation of the modified F1 score. Results are from our best model M_9.

	Predicted Class					
True Class		GO_BP	GO_MF	GO_CC	'O'	'EOS'
	GO_BP	3419	1	25	393	0
	GO_MF	0	96	1	2	0
	GO_CC	7	0	1288	116	0
	'O'	68	0	34	N/A	4
	'EOS'	0	0	0	0	N/A

We further analyzed our best model to gather insights into the model's performance. First, we show the modified F1 score for the three GO sub-ontologies (Table 4.3). The model shows similar performance for biological processes and cellular components and registers a substantially higher score for molecular function. This might be because the total number of annotations from the molecular function sub-ontology in the CRAFT corpus is far lower than the other two ontologies (see Table 4.1). Table 4.4

shows the false positives, false negatives, true positives, and true negatives among our predictions broken down by the three GO sub-ontologies. Next, we explored if the occurrence frequency of a concept in the training corpus impacts the model's prediction performance on that concept. Figure 4.3 breaks down the F1 score into five bins based on the GO terms' frequency of occurrence in the corpus. We see that GO terms with a frequency of co-occurrence between 1-10 have substantial variability in their F1 scores. Most of the GO terms with 10-20 occurrences show F1 scores between 0.6 and 1. We see some outliers in this bin where the F1 scores are lower than 0.6. All bins with occurrences of 20 and higher show high F1 scores (> 0.8) and low variability. This figure clearly shows that the model makes incorrect predictions for GO terms with low occurrences (< 20) in the corpus. We did not observe evident differences in prediction performance when the 1-10 occurrence bin was further subdivided into smaller intervals (Figure 4.4).

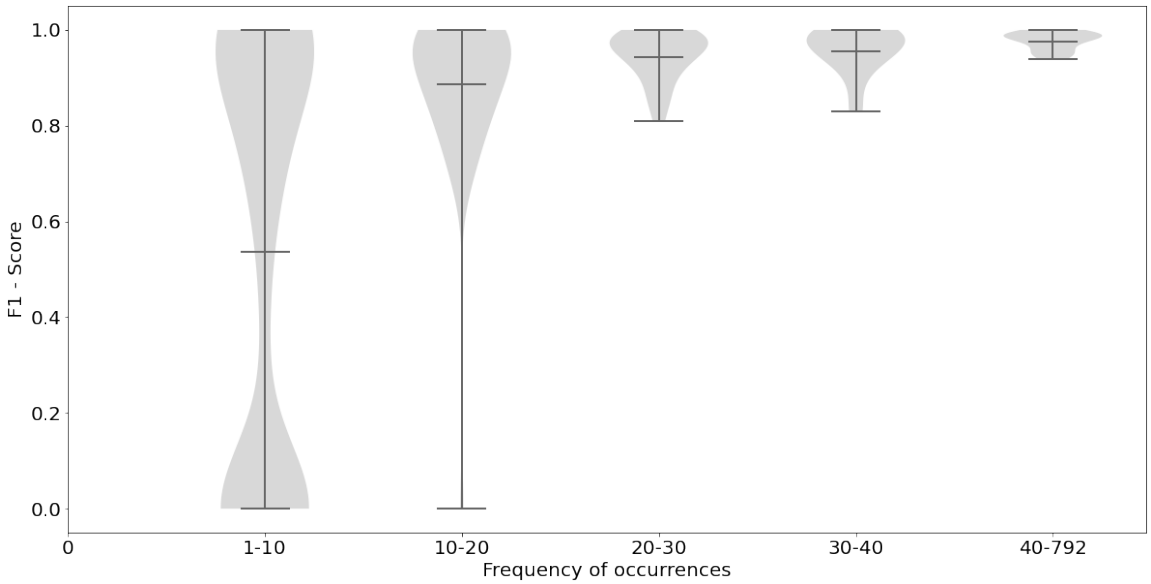


Figure 4.3. Distribution of F1 scores by occurrence frequency of GO terms in CRAFT

We compared our best model with classic BERT as well as SciBERT (Table 4.5). We find that SciBERT performs better than BERT by 3 points in F1 and 2 points in semantic similarity. Our model improves SciBERT's F1 by 4 points and semantic similarity by 2 points.

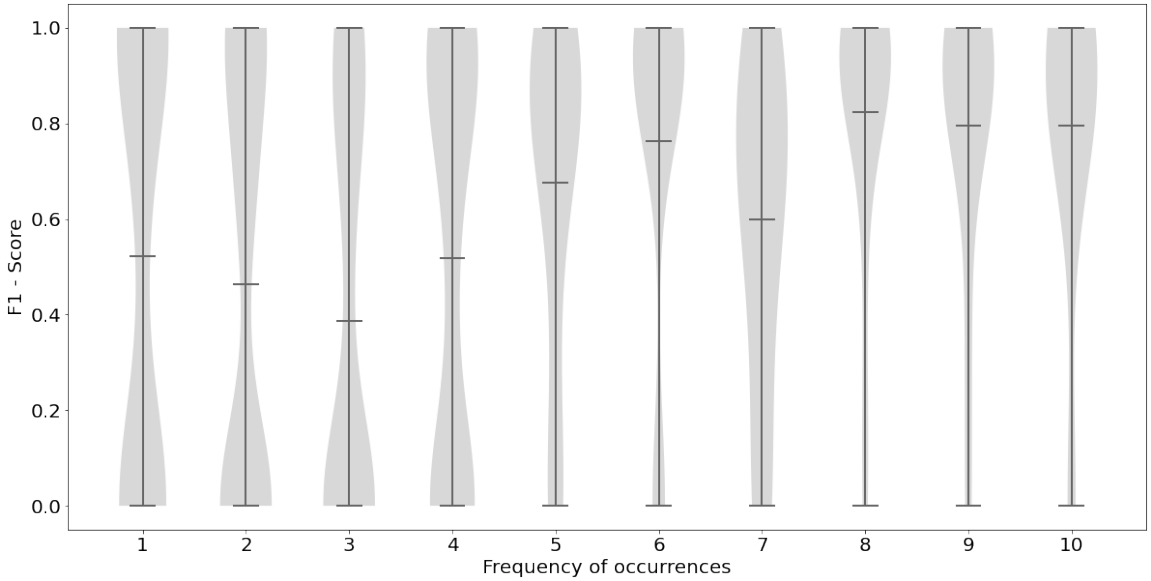


Figure 4.4. A closer look at the distribution of F1 scores for GO terms with 10 or fewer occurrences in CRAFT

Table 4.5. Performance comparison between our best model and two variants of BERT

Model	F1	Semantic Similarity
BERT	0.77	0.80
SciBERT	0.80	0.82
M_9	0.84	0.84

The model predicted 83.61% of annotations in the test set accurately. 9.34% were prediction errors where the model miss-classified GO annotations as non-annotations (`'O'` tags). 1.72% were prediction errors where the model miss-classified `'O'` tags as GO terms. Finally, in 5.32% of cases, the model predicted a different GO term than the GO term in the test corpus.

For each word in a sentence, the model outputs a tensor of the sigmoid ($\frac{1}{(1+\exp(-x_i))}$) activation outputs. These outputs are then converted to probabilities using a softmax function ($\frac{\exp(x_i)}{\sum_j \exp(x_j)}$). We can calculate the entropy ($H(X)$) over the tensor of probabilities to observe the level of "information" within the probabilities. For example, if there is uniformity in the probabilities for the predicted annotations, entropy is maximized, and vice versa. We visualized the interactions between entropy, predicted

probability, and the frequency of annotations, in Figure 4.5. Here, the dots represent the predicted annotations (annotations with the highest sigmoid activation) by the model. Incorrect predictions are shown in red and correct predictions are in blue.

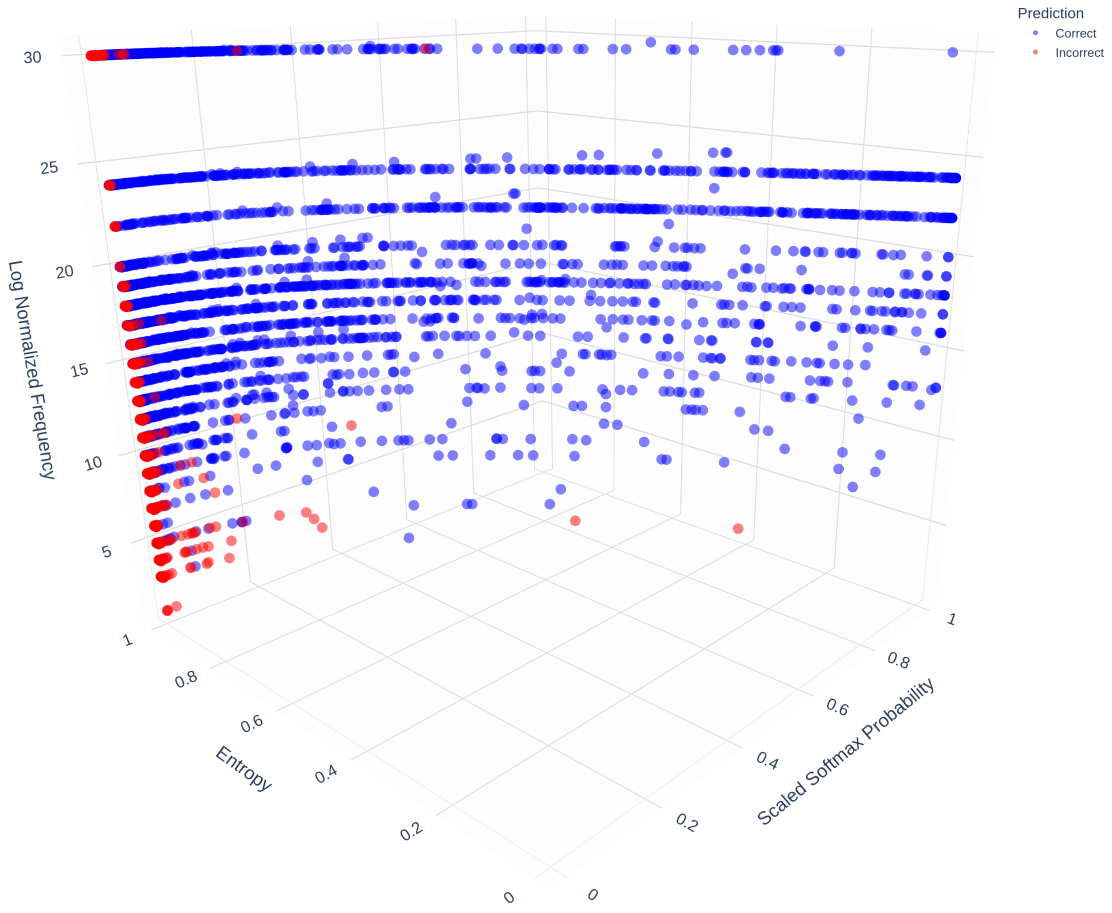


Figure 4.5. Distribution of incorrect and correct predictions with respect to entropy, probability, and frequency of occurrences.

We observe that as the probability score increases (for the top prediction) and the entropy reduces (across prediction tensor) the model predictions are more accurate. The high probability of the top prediction indicates the model's confidence and low entropy indicates that the model assigned low probabilities to the other potential predictions thereby offering a clear discrimination between the top prediction and the rest.

In comparison, incorrect predictions (Figure 4.6) are concentrated in a small area demarcated by low probability, high entropy, and low frequency. These incorrect predictions happen overwhelmingly at frequencies under 10 and probability values lower than 0.1. The entropy values of the majority of these predictions are close to 1 indicating that the model assigned near-uniform probabilities to the potential predictions. This combined with the low probability indicates that the model was not confident of any of the predictions it made.

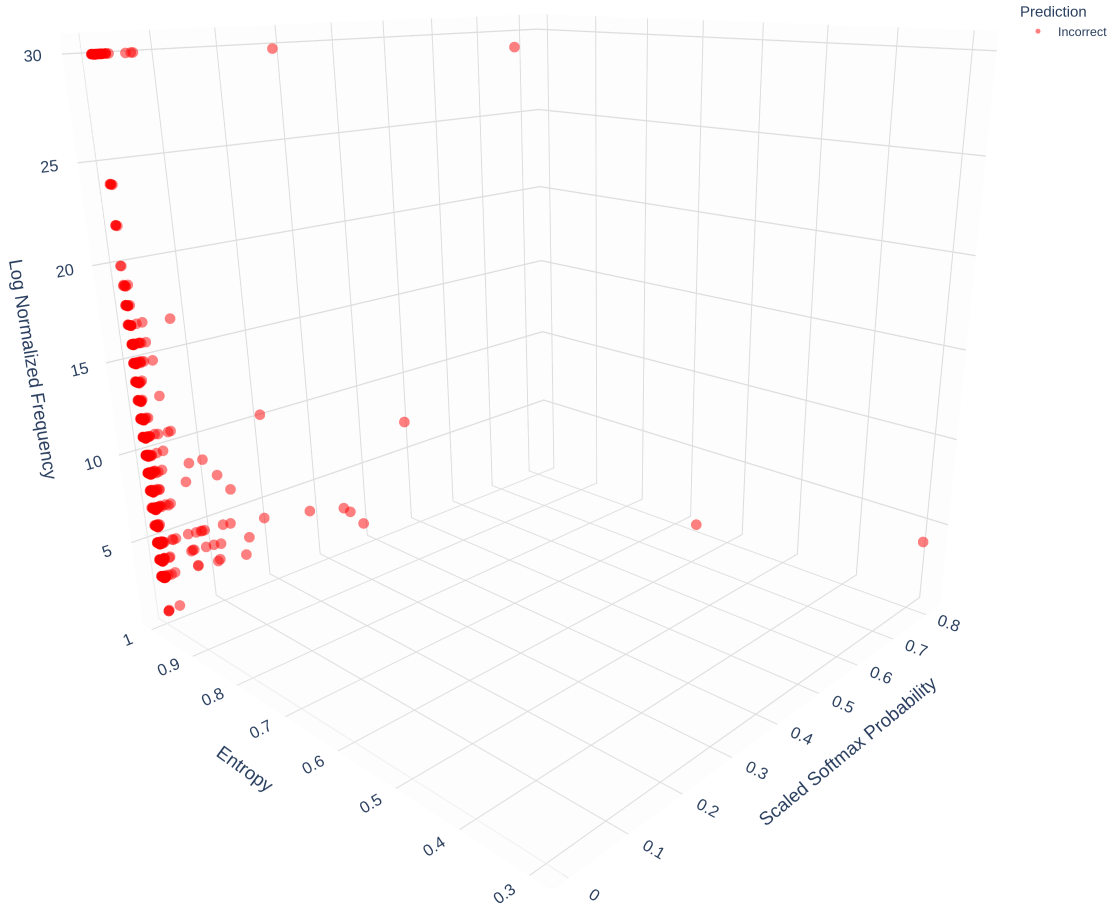


Figure 4.6. Distribution of incorrect predictions with respect to entropy, probability, and frequency of occurrences.

We tested if there are differences in the entropy, frequency, and probability distributions between correct vs. incorrect predictions using two-sided independent T-tests. We found statistically significant differences at the Bonferroni-corrected threshold of

$\alpha = 0.01$ between correct vs. incorrect predictions for entropy (p = 1.5e-221), frequency of occurrence (p = 2.9e-20), and probability of highest prediction (p=0.0).

We cannot use our results as a direct benchmark against other studies since the CRAFT corpus version used here might vary but we can remark on a general comparison. Lenz et al [39] developed a concept recognition system based on LSTMS which resulted in a 0.81 F1 (averaged across the three GO ontologies) as compared to our best model which results in a 0.84 F1 score. In a 2017 paper, OGER (OntoGene's Entity Recognizer)'s method for concept recognition uses dictionary lookup and flexible matching reporting an F1 score of 0.70 [54]. A systematic evaluation [55] found that ConceptMapper [56] a dictionary-lookup system performed the best among other tools resulting in an F1 of 0.83 (using earlier versions of CRAFT).

Chapter 5: Knowledge of the Ancestors [44]

In ontology-based NLP tasks, deep learning models have been applied to NER, NEN, and relation extraction [41, 42, 57]. NER focuses on identifying entities from unstructured text such as scientific literature while NEN focuses on linking the entities to unique concepts such as terms in an ontology. Relation extraction focuses on identifying relations between the identified ontology concepts and enables the automated creation of structured hierarchies. While our GRU-based deep learning models presented are successful at performing NER and NEN from biological literature, they lack awareness of the underlying ontology hierarchy. It is critical for ontology-based information retrieval systems to know the ontology structure and relationships. It enables the model to make intelligent predictions for concepts that take into account patterns learned from the training data as well as semantics embedded in the ontology.

The key difference between traditional information retrieval vs. ontology-based information retrieval is the possibility of partial success. Traditional information retrieval systems are evaluated based on whether the target information is retrieved (success) or not (failure). In contrast, ontology-based information retrieval systems are evaluated based on three possibilities: accurate retrieval (success), inaccurate retrieval (failure), or partially accurate retrieval (partial success).

If a model accurately predicts the ontology concept from the gold standard data, it is counted as a success and is scored a 1. If the model fails to predict any ontology concept, it is counted as a failure and is scored a 0. If the model retrieves an alternative ontology concept from the one in the gold standard (partial success), the model is

scored depending on how semantically similar the retrieved concept is to the true concept. Intuitively, the most semantically similar concept to the true concept would lie in the near vicinity (such as a sibling, or a parent). Thus, it is important for the model to be aware of the ontology hierarchy to be able to make intelligent predictions in cases when it misses the actual concept. The goal of intelligent concept annotation presented in this study is to maximize accurate retrieval rates and subsequently maximize partial accuracy in cases where complete accuracy is not achieved thereby improving overall accuracy.

This nuance of ontology-based concept retrieval means that the first goal of these models is to predict the true concept. If the first goal is not achieved, the model should aim to predict the most semantically similar concept to the truth so as to maximize the partial accuracy score. The ontology hierarchy contains valuable information regarding the semantics embedded in the ontology. This crucial information has been ignored in past work resulting in ontology recognition models that are not fully aware of the ontology semantics. Here, we present intelligent deep learning architectures that are ontology-aware and use the hierarchies embedded in the ontology to improve concept prediction accuracy.

5.1 Methods

5.1.1 Deep Learning Architecture

Our deep learning architecture (Figure 5.1) consists of three key components — 1) input pipelines; 2) embedding/latent representations; and 3) a deep learning model.

Input Pipelines

The neural architectures in this study are designed to use fixed-size inputs. As established in section 4.1.1, each sentence is transformed to a size of 71 words. Sentences with a lower number of words are padded with the token `<PAD>` and ones with a higher number of tokens are truncated to a length of 71.

We provide three inputs for each word in a sentence - 1) token (X_{train}^{token}), 2) character sequence (X_{train}^{char}), 3) parts-of-speech (X_{train}^{POS}).

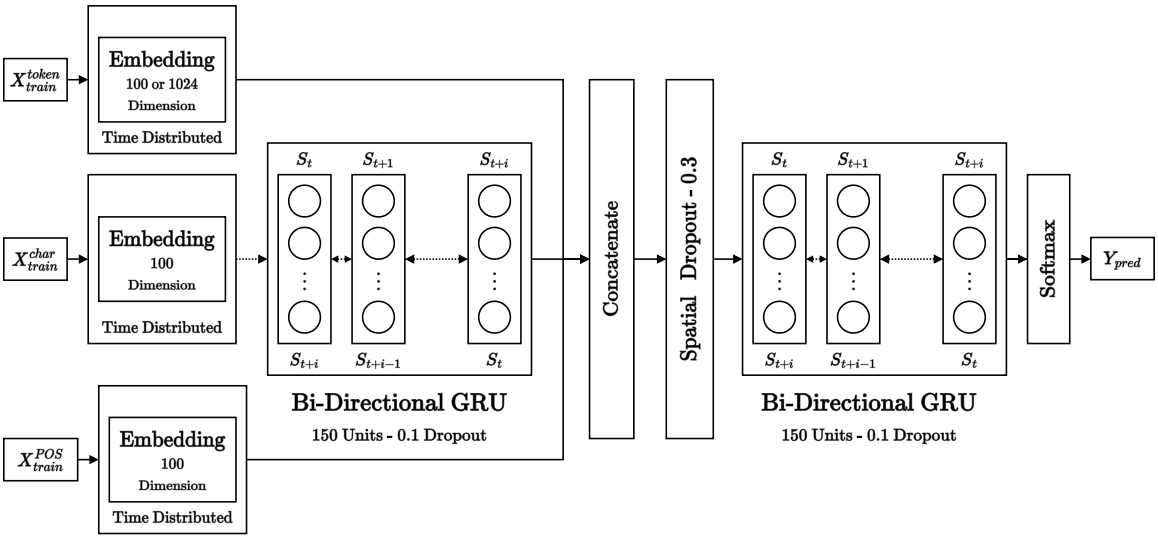


Figure 5.1. Architecture of GRU model with three input pipelines

The token (X^{token}_{train}) input, is a sequential tensor consisting of 71 tokens, where each token is represented with a high dimensional one hot encoded vector (for 34,164 unique words/tokens present within our corpus vocabulary). Similarly, the character sequence (X^{char}_{train}) is also a sequential tensor consisting of character sequences present in a word/token. Each token is transformed into a 15 character sequence; the value 15 is derived from the frequency distribution of characters as described in section 4.1.1. Words with more than 15 characters are truncated to a sequence of 15 and ones with lower than 15 characters are padded with padding characters to make the final sequence of 15 characters. Any unique character not present in the corpus can be used for padding. The choice of character does not really matter as long as the chosen padding character does not match with the ones present in the corpus since each character needs to be consistently converted to numbers in the final stage of model input. Next, we provide POS tags that indicate the type of words in a sentence (X^{POS}_{train}).

Embedding/Latent Representations

Our architecture utilizes embeddings to provide a compressed latent space representation for very high dimensional input components. We evaluated three different approaches for embeddings (shown as Emb. in Figure 5.2) - 1) CRAFT 2) Global

Vectors for Word Representation(GloVe), 3) Embeddings from Language Models (ELMo).

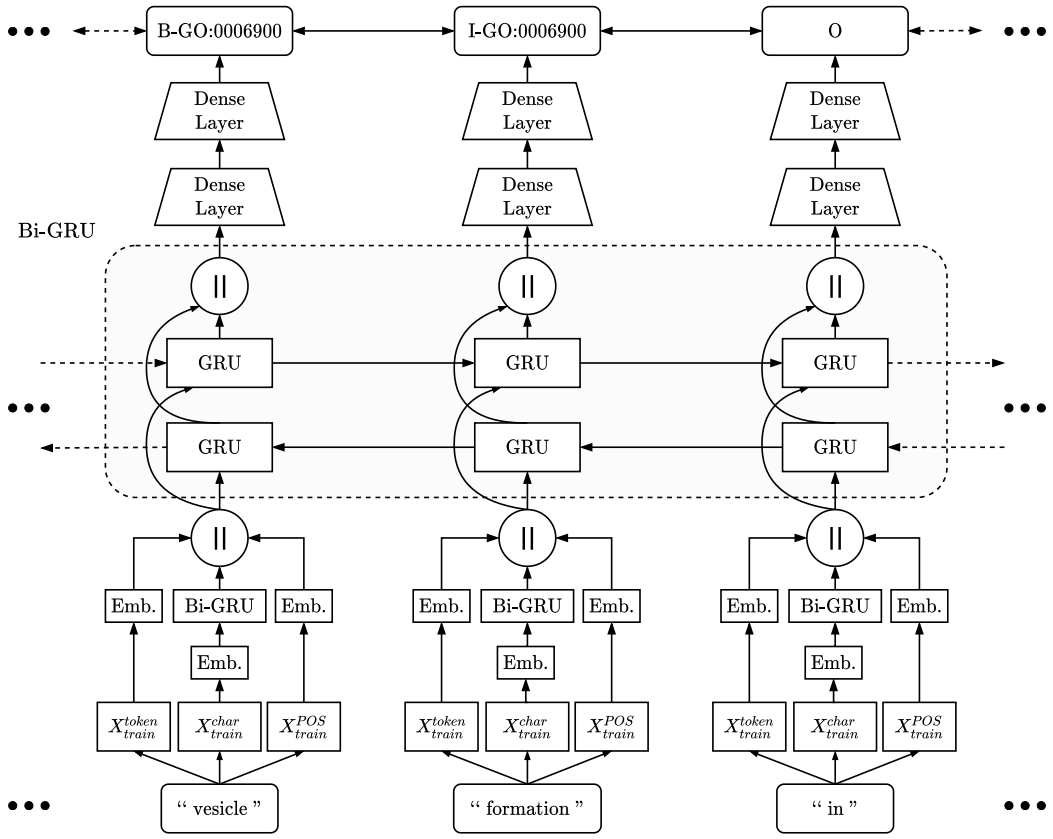


Figure 5.2. Snapshot of GRU model with example input sequence

The supervised embedding layer learns to map the one hot encoded input into a smaller dimensional representation. The resulting embedding learns the mapping of the IOB tags to the tokens of the sentences. The layer is used with token inputs (X^{token}_{train}), character sequences (X^{char}_{train}), and character representation (X^{POS}_{train}), each of which has very high dimensionality in their original vectors.

We also evaluate GloVe and ELMo embeddings for the X^{token}_{train} input. GloVe with a 300-dimensional output embedding vector and ELMo with 1024 dimensional output embedding vector are used in all experiments. While ELMo and GloVe are pretrained embedding models, we also evaluate the model's performance using the CRAFT embedding. Here we use an embedding layer that is co-trained during the model

training, where each word is mapped to an embedding vector which is a continuous vector of 100 dimensions. Initially, the embedding values of the layer are randomly initialized but are then optimized for word lookup during the model training for the objective of ontology annotation.

Sequence Modeler

Figure 5.2 shows a snapshot of the model architecture in the context of training and inference of a sample set of tokens. Here we show the training/inference on a sequence of tokens "vesicle", "formation", and "in" (which are parts of a sentence) as it is evaluated by the network. Each token is preprocessed to obtain the representative tensors – X_{train}^{token}, X_{train}^{char}, X_{train}^{POS} which are passed through embedding layers, where the embedding of X_{train}^{token} can be a complete pretrained architecture such as GloVe or ELMo. The embedding of X_{train}^{char} is also passed via a Bi-directional GRU (Bi-GRU) layer. All of the resulting values are concatenated to be processed via the main Bi-GRU layer. The bi-directionality allows the architecture to learn the preceding and succeeding sequence patterns within the sequence tokens in a sentence. The states of both the GRU layers are then concatenated to provide the final dense layer, which is the softmax classifier of the architecture and predicts the associated IOB tags for the input tokens. Here, we select the tag with the highest probability for each of the tokens.

Architecture hyper-parameters, which include supervised embedding shape ({20, 50, 100, 150, 200}), dropout ({01, .2, .3, .5, .7}), number of epochs ({50, 100, 200, 300}), and class weighting, were evaluated using a grid search approach. We use the Adam algorithm with weight decay [58] as our optimizer with an initial learning rate of 0.001. The learning rate is reduced by a factor of 0.1 after the first 10000 training steps, and reduced further by a factor of 0.1 after the next 15000 steps. The weight decays by a factor of 0.0001 after the first 10000 steps and further by another factor of 0.0001 after the next 15000 steps. We use sigmoid focal cross entropy [59] as the loss function. Sigmoid focal cross-entropy is particularly useful for cases where we have highly imbalanced classes. It reduces the relative loss for easy to classify, higher frequency examples, putting more focus on harder to classify, misclassified examples. This loss function uses α, also called the balancing factor, and β or modulating factor,

which are set to 0.25 and 2.0 respectively.

We also compare our approach to a large-scale masked language model - BERT.

5.1.2 Target Vector Representation

Target labels to be predicted are typically provided as a one-hot encoded vector where the size of the vector equals the number of output labels. In our case, the output labels correspond to the set of all GO terms. Typically, the value of the GO term to be predicted is set to 1 and the value of all other GO terms is set to 0. This approach of representing the target labels, however, does not allow the model to learn the ontology hierarchy nor does it allow for semantically similar partial predictions.

In this study, we use Jaccard semantic similarity scores as values in the label vector. The value of the GO term to be predicted is set to 1 and the value of all other GO terms in the vector is set to the Jaccard similarity score between that term and the GO term to be predicted. This representation allows the model to identify the target GO term followed by "similar" GO terms that are partially accurate predictions. This output label representation also helps the model optimize the weights to target more than one prediction label. We also add a weighting factor β to modulate the effect of the Jaccard similarity score on the target vector. So for each output tag, the representation is as follows:

$$Y = \begin{cases} l = [1], & \text{if } \mathcal{T} == \widehat{\mathcal{T}} \\ l = [\beta * \mathcal{J}_{sim}(\mathcal{T}, \widehat{\mathcal{T}})], & \text{if } \mathcal{T} \neq \widehat{\mathcal{T}} \ \& \ \mathcal{T} \neq \mathtt{0} \end{cases} \tag{5.1}$$

where, Y is the final target vector, l is the label for the word, $\mathcal{T}$ is the ground truth tag, $\widehat{\mathcal{T}}$ is the predicted tag, β is the Jaccard weight, and $\mathcal{J}_{sim}$ is the Jaccard similarity between $\mathcal{T}$ and $\widehat{\mathcal{T}}$. The target vector Y is computed by comparing the true tag ($\mathcal{T}$) with the possible tags ($\widehat{\mathcal{T}}$), where if the $\mathcal{T} == \widehat{\mathcal{T}} == O$ (no annotation) OR a GO annotation then the value is set to 1. Else, if the ground truth tag ($\mathcal{T}$) is a GO term, then we calculate its Jaccard similarity to all possible GO terms and create the target vector by weighting it with β. We evaluate β values between {0, .25, .5, 1}, where a β value 0 indicates the traditional one-hot vectorization (baseline in results) and a β

value 1 indicates the full Jaccard score taken into account.

The Jaccard similarity ($\mathcal{J}_{sim}$) of the ground truth concept $\mathcal{T}$ and a predicted concept $\widehat{\mathcal{T}}$ [18] is calculated as:

$$\mathcal{J}_{sim}(\mathcal{T}, \widehat{\mathcal{T}}) = \frac{|S(\mathcal{T}) \cap S(\widehat{\mathcal{T}})|}{|S(\mathcal{T}) \cup S(\widehat{\mathcal{T}})|} \tag{5.2}$$

where, $S(\mathcal{T})$ is the set of ontology subsumers of $\mathcal{T}$. Specifically, $\mathcal{J}_{sim}$ of two concepts (A, B) in an ontology is defined as the ratio of the number of concepts in the intersection of their subsumers over the number of concepts in the union of their subsumers [18].

5.1.3 Performance Evaluation Metrics

The performance of each experiment is evaluated using a modified F1 score as presented in our previous work [43]. To avoid biasing the F1 score, we omit accurate predictions of `'O'` tags from the calculation to report a relatively conservative F1 score.

F1 quantifies whether the model's prediction matches the actual annotation exactly. However, ontology-based prediction systems need to be evaluated while accommodating partially accurate predictions. For example, a model might not retrieve the exact ontology concept as the gold standard but a related concept (sub-class or super-class) achieving partial accuracy. Semantic similarity metrics [18] designed to measure different degrees of similarity between ontology concepts can be leveraged to measure the similarity between the predicted concept and the actual annotation to quantify the partial prediction accuracy. Here, we use Jaccard similarity which measures the ontological distance between two concepts, to access the model's performance for the partial similarity between the predicted tags and the actual ground truth.

5.1.4 Top two predictions

The model makes predictions for each GO instance in the test set. These predictions are expressed as a probability vector where each potential GO term is assigned a probability. When evaluating the performance of the model, we typically pick the GO term with the highest probability and use that as the model's prediction. However, it

is typical in NLP evaluations to consider the top 2, 5, and even 10% of probabilities in evaluating the performance of the model [60–62]. In one of our prior works, we showed that using the top 2 probabilities can result in a substantial increase in accuracy [9]. Here, we use the same practice of considering predictions with the top 2 probabilities to evaluate our model's performance.

5.2 Results and Discussion

The CRAFT v4.0.1 dataset contains 18,689 annotations pertaining to 974 concepts from the three GO sub-ontologies across 97 articles. The majority of these concepts belong to Biological Processes followed by Molecular Functions, and Cellular Components.

First, we look at a comparison of the three embeddings - CRAFT, GloVe, and ELMo. We establish a baseline performance by training the model with a binary target vector and not using Jaccard similarity scores. This baseline can help understand the performance improvements resulting from training the model with semantic similarity scores. Row 1 of Table 5.1 shows the baseline F1 and Jaccard similarity with the three embeddings. We see that ELMo results in the highest F1 (0.79) and the highest Jaccard score (0.82). Considering the top 2 predictions increases the F1 to 0.86 and the Jaccard score to 0.90.

Table 5.1. Comparison of baseline vs ontology aware model performance. Note: F1 scores are modified by omitting accurate predictions of non-annotations (indicated by 'O') for a conservative estimate of performance on annotations only.

Model	Embedding	F1	Jaccard	Top two F1	Top two Jaccard
Baseline	CRAFT	0.74	0.75	0.82	0.86
	GloVe	0.75	0.76	0.832	0.87
	ELMo	0.79	0.82	0.86	0.90
Ontology aware model	CRAFT	0.80	0.83	0.86	0.91
	GloVe	0.79	0.82	0.86	0.90
	ELMo	0.81	0.84	0.87	0.92

We test the prediction performance with four different settings of the weight parameter β (0.25, 0.5, 0.75, and 1). The best performance is obtained with a weight of 0.5 (Row 2, Table 5.1). In both the baseline and the ontology-aware model, the ELMo

embedding outperforms the other two embeddings across all metrics.

The impact of training the model with the ontology hierarchy is starkly noticeable in both F1 and Jaccard across all three embeddings. The highest improvement is observed for CRAFT embeddings (8% F1, 10% Jaccard) followed by GloVe (6% F1, 8% Jaccard). ELMo showed modest improvements between the baseline and the ontology-aware model (2% F1 and Jaccard). These results suggest that intelligent models that are trained with the ontology hierarchy make more accurate predictions as compared to those that are trained just on the target ontology concept.

We present a few examples of instances where the model's predictions match the ontology term in the gold standard as well as instances where the model generates false negatives or partially accurate annotations (Table 5.2).

Table 5.2. Examples of accurate, partially accurate, and inaccurate annotation predictions

Phrase	Gold standard annotation	Model's annotation
Accurate annotation prediction		
endogenous **intracellular**	GO:0005622	GO:0005622
cell divisions before developmental	GO:0051301	GO:0051301
Partially accurate annotation prediction		
proper **rhabdomere morphogenesis**	GO:0061541	GO:0031069
auto **regulation of blood flow**	GO:1903522	GO:0008217
Inaccurate annotation prediction		
polysomal - associated RNA-binding protein	GO:0005844	O
protein degradation	GO:0030163	O

We also compare the transformer based BERT model which has been shown to perform state of art results in a large of named entity recognition and other NLP tasks. Results (table 5.3) show that our ontology-aware model outperforms BERT in both F1 and Jaccard by 5%. The baseline model that shows lower performance than the ontology-aware model also outperforms BERT in our test.

It is interesting to see that a generic ELMo embedding performs better than a domain-specific embedding such as CRAFT. However, it is to be noted that the ELMo and GloVe embeddings are pretrained on large corpora. They are computationally more

Table 5.3. Performance comparison between our best model and BERT. Note: F1 scores were modified by omitting accurate predictions of non-annotations (indicated by 'O') for a conservative estimate of performance on annotations only.

Model	F1	Jaccard
BERT	0.77	0.80
Ontology aware model (ELMo)	0.81	0.84

expensive to infer during model training and even more expensive to develop. In contrast, CRAFT utilizes a simple embedding layer with a significantly lower number of parameters making it more accessible for scientists.

Prior approaches in the area of automated ontology annotation treated ontological concepts as a binary outcome. In our approach, the Jaccard similarity target vector teaches the model to understand the latent semantic relationships between the GO concepts.

Chapter 6: Ontology-powered Boosting

Ontology-powered Boosting (OB) is a novel boosting approach that combines the prediction of a fully trained deep learning model with the graph structure of ontology concepts. Here, we are trying to improve the semantic similarity performance of a trained model during post-processing by taking the ontology knowledge graph into account. The goal of OB is to selectively boost the confidence of an ontology prediction to improve the overall prediction accuracy. We take a two-step approach, 1) identify candidates for boosting, and 2) boost the predictions with semantically similar concepts.

6.1 Methods

6.1.1 OB - A two-step process

In the first step, we identify the candidate predictions where the deep learning models had low confidence. We calculate the uncertainty in the predictions based on the last layer softmax output. The layer outputs a probability vector ($\nu_i =< p(x_0), p(x_1), \cdots p(x_m) >$, where i is the i'th input token and $p(x_j)$ probability of tag x_j), corresponding to all possible tags ($0 \cdots m$), for each token that is provided as input to the model. In general, we calculate the $argmax(\nu_i)$ to select the tag with the highest probability as the model output. Here we leverage the top k probabilities from ν_i vector to calculate the uncertainty in the model's predictions by evaluating the entropy $H({\nu_i}^k)$ using Shannon's information entropy where ${\nu_i}^k = p(argmax_k(\nu_i))$.

The highest predicted probability of the ν_i probability vector and the entropy ($H(\nu_i)$)

value are used to determine the threshold for the predictions where boosting needs to be applied. The intuition behind this is to only boost specific predictions where the model has low confidence. We choose the thresholds for the two parameters by analyzing the predictions graphs (Figures 6.3a, 6.3b), i.e. visualizing the thresholds of the parameters where the model makes the most errors. We also select the top k predictions ($argmax_k(x)$) from the ν_i vector to boost. Boosting all of the possible tag predictions does not benefit the model's performance and has a detrimental effect on computational overhead.

The second step boosts the predicted probabilities of the identified candidates by combining them with the model predictions of the candidate's ancestors/subsumers. Specifically, for each token i, we boost the probabilities of top k tags ($p(argmax_k(\nu_i))$) with the probabilities of their subsumers using the following computation:

$$\mathcal{I}(x_j) = -log(f_x/C)$$

$$\tilde{p}(x_j) = \beta * p(x_j) * \mathcal{I}(x_j) + \sum_{n=1}^{d} \frac{\alpha * p(x_j^n) * \mathcal{I}(x_j^n)}{n}$$

where, we first calculate the information content ($\mathcal{I}(x_j)$) of the tag x_j ($0 \leq j \leq m$, where m is the number of tags) as the negative log of concept frequency (f_x) over the total number of available concepts (C). $\mathcal{I}(x_j)$ is then utilized to calculate the boosted probability, $\tilde{p}(x_j)$, which consists of two components, the modulated original probability ($\beta * p(x_j) * \mathcal{I}(x_j)$) and supportive parent boosting ($\sum_{n=1}^{d} \frac{\alpha * p(x_j^n) * \mathcal{I}(x_j^n)}{n}$).

The modulated original probability combines the original probability with a weighting factor β and the information content of the concept $\mathcal{I}(x_j)$. The second part of the computation evaluates all of the subsumer probabilities of x_j by individually calculating the modulated probabilities of parents ($\alpha * p(x_j^n) * \mathcal{I}(x_j^n)$), where x_j has d ancestors while controlling the influence by normalizing with the depth factor n. Here α is a weighting parameter used to control the influence of the ancestor probabilities on the boosting. The calculated parent probabilities are then summed and added to the modulated probability of x_j.

Using the aforementioned approach, $\tilde{p}(x_j)$ combines the predicted probabilities of the

tags with their ancestor's predictions from a single model. This is done only for the GO annotations, where a specific annotation probability might be boosted to make it the top prediction if it had supporting parent predictions from the model. The information content parameter modulates the effect on the boosted probability by taking the frequency of occurrence of a concept and its hierarchy into account. α and β parameters can further control the emphasis we put on the parental contribution vs original probability, where a β value of 0 nullifies the parental contribution and of 1 includes the parental support completely. As we go higher in the ancestor path, the depth factor n enforces lower contributions coming from higher subsumers.

We utilize Bayesian optimization to evaluate the different values of k (top k predictions), entropy threshold ($H(\nu_i)$), α, and β to maximize the model prediction accuracy. Specifically, we utilize Tree-structured Parzen Estimators (TPE) [63] approach to derive the optimal values for each of the parameters for maximizing our objective function. The objective function in the experiment is defined to maximize the mean semantic similarity. α and β are evaluated with continuous values between 0.0 to 1.0, while k was evaluated for values between 1 - 10. The range of entropy was defined to be continuous values ranging from 0.0 to the highest value of entropy of the top k predictions for each token.

We demonstrate the efficacy of the Ontology Boosting approach on two deep learning architectures from our previous work [43, 44].

6.1.2 Deep Learning Architectures

We evaluate our boosting approach on two deep learning architectures from our prior work [43, 44]. We describe the two architectures briefly here. For details refer to [43, 44].

Architecture 1 - Externally Augmented Predictor (A_1)

Figure 6.1 shows the three key components of A_1 — 1) Input Pipelines; 2) Embedding/Latent Representations; and 3) Sequence Modeler. This architecture was originally published in [43].

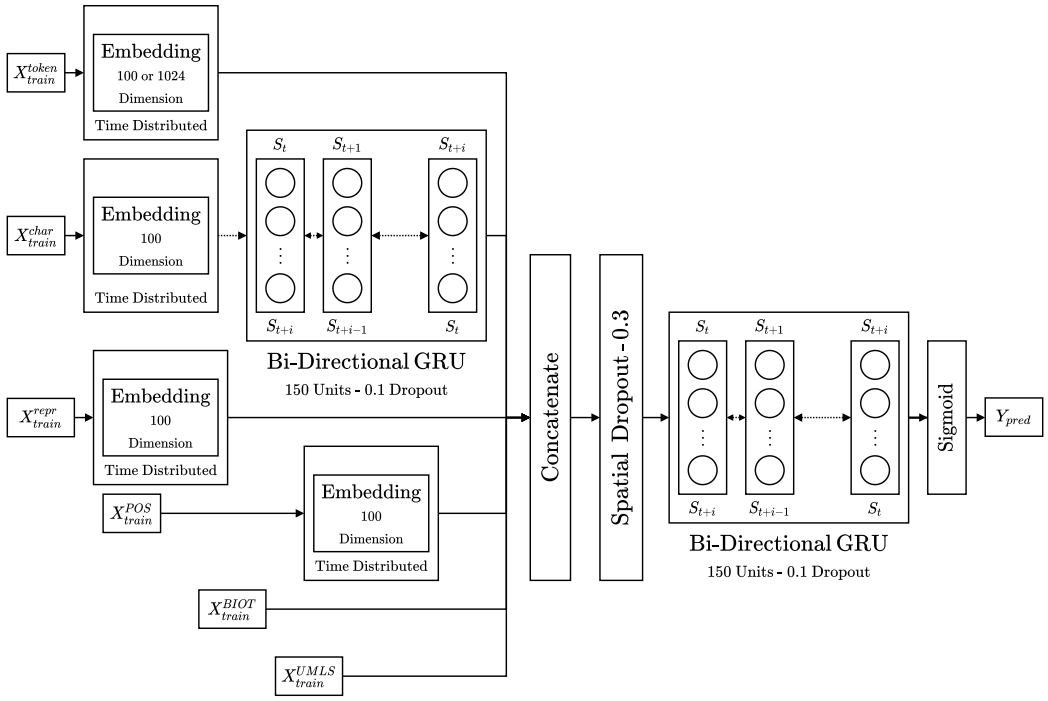


Figure 6.1. Architecture of a GRU model for ontology concept recognition. Figure originally published in [43]

Input Pipelines Each sentence and each token are provided six different components as input — 1) token (X^{token}_{train}), 2) character sequence (X^{char}_{train}), 3) token-character representation (X^{repr}_{train}), 4) parts-of-speech (X^{POS}_{train}), 5) BioThesaurus (X^{BIOT}_{train}), and 6) UMLS (X^{UMLS}_{train}).

For more details about the pipeline, refer to section 4.1.1.

Embedding/Latent Representations The supervised embedding is a bottleneck layer that learns to map the one hot encoded input into a smaller dimensional representation. We used ELMo pretrained embeddings for the X^{token}_{train} input. Embeddings in ELMo are learned via a bidirectional language model where the sequence of the words is also taken into account. We use the pretrained model on a 1 Billion Word Benchmark, which consists of approximately 800M tokens of news crawl data and has an embedding of 1024 dimensional output embedding vectors.

Sequence Modeler We utilize Bi-GRUs in two locations in the architecture, first to model the sequence of characters present in each token and second main Bi-GRU model to concatenate input pipelines together. After the embedding of the characters, they are passed via the first Bi-GRU (consisting of 150 units) resulting in a sequence representation of the characters in a sentence. 10% dropout is used in this pipeline to regularize the output to prevent overfitting.

The character sequence representation is then concatenated with the ELMo embeddings, character representation, parts of speech, and input tensors from Bio-Thesaurus and UMLS. This concatenated feature map representing each sentence is then passed to a spatial dropout, which removes 30% of the 1-D sequence features from the input to the main Bi-GRU. The main Bi-GRU processes the feature maps (with 10% dropout), and outputs to a single time-distributed dense layer of 1774 nodes (representing each of the output tags).

Architecture hyper-parameters, which include — supervised embedding shape ({20, 50, 100, 150, 200}), dropout ({01, .2, .3, .5, .7}), number of epochs ({50, 100, 200, 300}), and class weighting, were evaluated using a grid search approach. We used Adam [52] as our optimizer for all of the experiments with a default learning rate of 0.0001.

Architecture 2 - Intelligent Predictor (A_2)

A_2, as shown in figure 6.2, uses an intelligent prediction system by using the ontology hierarchy structure as opposed to A_1 [44]. This architecture was originally published in [44] and be referred from section 5.1.1.

The overall structure of the A_2 is similar to A_1 in terms of the Sequence Modeler and Embedding/Latent Representation components. This architecture varies in the Input Pipelines provided as well as how the target vector is represented for training the model.

Input pipelines We provide three inputs for each word in a sentence - 1) token (X_{train}^{token}), 2) character sequence (X_{train}^{char}), 3) parts-of-speech (X_{train}^{POS}).

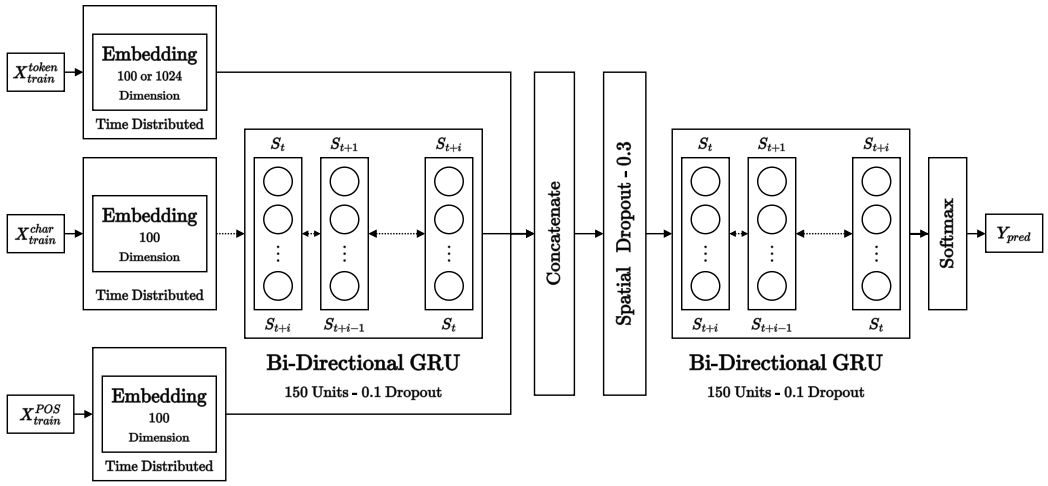


Figure 6.2. Architecture of intelligent ontology model.
Figure originally published in [44].

Target Vector Representation In A_2, we use Jaccard semantic similarity scores as values in the label vector. The value of the GO term to be predicted is set to 1 and the value of all other GO terms in the vector is set to the Jaccard similarity score between that term and the GO term to be predicted. This representation allows the model to identify the target GO term followed by "similar" GO terms that are partially accurate predictions. This output label representation also helps the model optimize the weights to target more than one prediction label. For more details about the creation of label vector and Jaccard semantic similarity, refer to section 5.1.2.

Sequence Modeler The sequence modeler for A_2 is similar to the sequence modeler in A_1. The differences are in the optimizer, the loss function, and the activation function in the final output layer. A softmax activation is used in the final layer which normalizes the output of the model to a probability distribution over the output tags. We use the Adam algorithm with weight decay [58] as our optimizer with an initial learning rate of 0.001. The learning rate is reduced by a factor of 0.1 after the first 10000 training steps, and reduced further by a factor of 0.1 after the next 15000 steps. The weight decays by a factor of 0.0001 after the first 10000 steps and further by another factor of 0.0001 after the next 15000 steps. We use sigmoid focal cross entropy [59] as the loss function. Sigmoid focal cross-entropy is particularly useful for cases where we have highly imbalanced classes. It reduces the relative loss for

easy to classify, higher frequency examples, putting more focus on harder to classify, misclassified examples. This loss function uses α, also called the balancing factor, and β or modulating factor, which are set to 0.25 and 2.0 respectively.

6.1.3 Performance Evaluation Metrics

Our primary evaluation metric in this study is semantic similarity. Semantic similarity metrics [18] designed to measure different degrees of similarity between ontology concepts can be leveraged to measure the similarity between the predicted concept and the actual annotation to quantify the partial prediction accuracy. Here, we use Jaccard similarity [18] that measures the ontological distance between two concepts to assess partial similarity.

We also provide a modified F1 score for our architectures. Since the majority of tags in the training corpus are non-annotations, the model predicts them with great accuracy. In order to avoid biasing the F1 score, we omit accurate predictions of `'O'` tags from the calculation to report a relatively conservative F1 score.

6.2 Results and Discussion

Table 6.1 presents a summary of how boosting affected the results of the two architectures. First, we see that the majority of tokens are selected for boosting via the Bayesian selection process (Row 2). However, the majority of tokens that are boosted remain unchanged indicating that when the model makes correct predictions, the boosting process largely retains the correct prediction (Row 3). Reassuringly, boosting does not change any of the GO predictions to an 'O' tag (Row 4). The majority of incorrect predictions made by the models happen when the ground truth is a GO concept but the model incorrectly predicts an 'O' (non-annotation). Boosting makes a substantial difference in this case by changing these instances from an 'O' to a GO concept (Row 5). Of these instances, 37% (A_1) - 41% (A_2) are corrected from an 'O' prediction to an exactly matching GO concept as the ground truth (Row 6). When boosting corrects an 'O' prediction to a GO term (exact or partial match to the ground truth), the average semantic similarity of these instances lies between 53 % - 60%.

Table 6.1. Effect of ontology boosting on the two architectures

Row	Description	A_1	A_2
1	Total number of tokens	5439	5495
2	Number of tokens boosted	4972	5197
3	Number of tokens boosted but unchanged	4411	4587
4	Number of tokens boosted from GO to O	0	0
5	Number of tokens boosted from O to GO	534	366
6	Number of tokens boosted from O to an exact GO	197	152
7	Average Semantic Similarity for O to GO	0.53	0.60

We examine the impact of our boosting approach on improving the prediction accuracy of the two architectures presented above (Table 6.2). The base scores refer to the output of the architecture before boosting was applied and the boosted scores reflect performance after boosting is applied. We see that boosting improves Jaccard semantic similarity scores by 7% for A_1 and 5.8% for A_2. The F1 scores experience a modest improvement of 2.5% for A_1 and no improvement for A_2.

Table 6.2. Impact of boosting on the two architectures

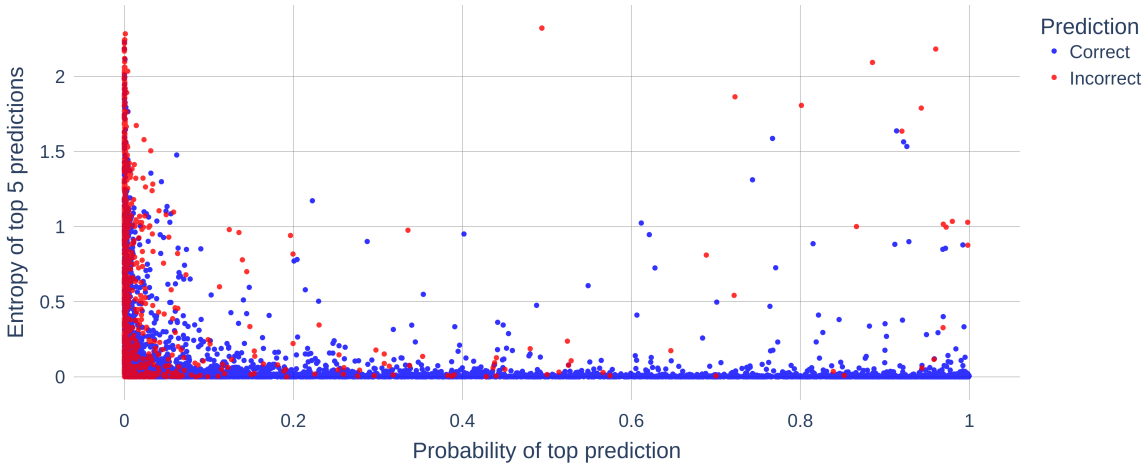
Architecture		Semantic Similarity	Modified F1
A_1	Base	0.84	0.83
	Boosted	0.90	0.85
A_2	Base	0.85	0.81
	Boosted	0.90	0.81

Ontology Boosting corrected 201 incorrect predictions (A_1) while changing 6 correct predictions to a semantically similar concept to the ground truth. Similarly, boosting corrected 174 incorrect predictions (A_2) while changing 113 correct predictions to a different GO concept (semantically similar). The net effect of these two contributions appears to result in modest improvements or keeps the F1 score unchanged. However, the real contribution of boosting is reflected in the semantic similarity scores which show an improvement.

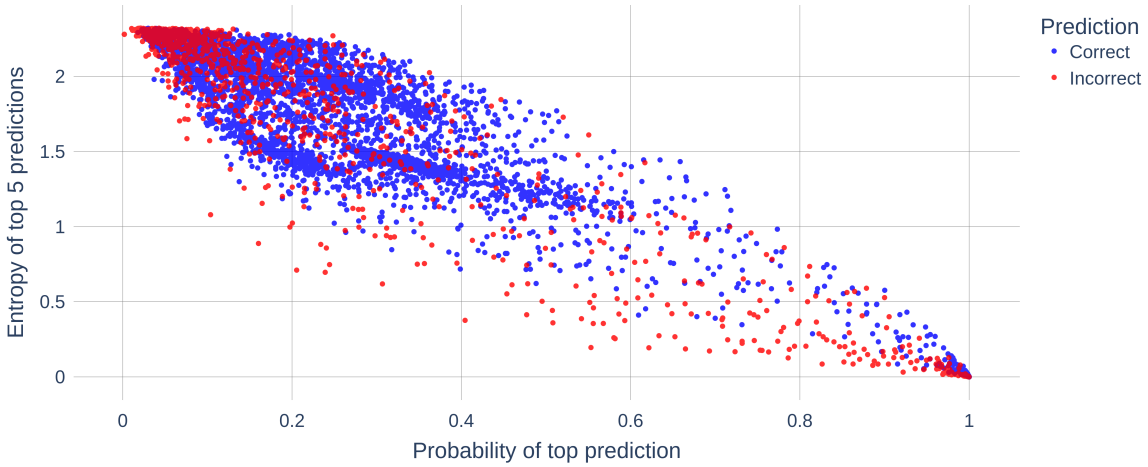
Figures 6.3a and 6.3b show the probability of the highest prediction and entropy (of the top 5 predictions) across all tokens in the dataset. Correct predictions are shown in blue and incorrect ones are shown in red. These graphs indicate the probability and entropy zones where the architectures make incorrect predictions that can be corrected using boosting. We use the Bayesian method described above to select incorrect predictions on this graph to be boosted.

Figure 6.4a shows the incorrect predictions by A_1. Like above, the majority of these instances were cases where the ground truth is a GO concept and the prediction is an 'O' term (non-annotation). Figure 6.4b shows instances incorrectly predicted as 'O' that were corrected by boosting. In this figure, blue instances represent cases where the boosted prediction was an exact match to the ground truth (100% accuracy) whereas the purple instances represent cases where the boosted prediction was a partial match to the ground truth. The size of the purple instances reflects the degree of partial relatedness to the ground truth - larger indicates higher semantic similarity to the ground truth. We see that boosting has a substantial effect on correcting inaccurate predictions.

Figure 6.5a shows the incorrect predictions by A_2. The majority of these instances were cases where the ground truth is a GO concept and the prediction is an 'O' term (non-annotation). Figure 6.5b shows instances incorrectly predicted as 'O' that were corrected by boosting.

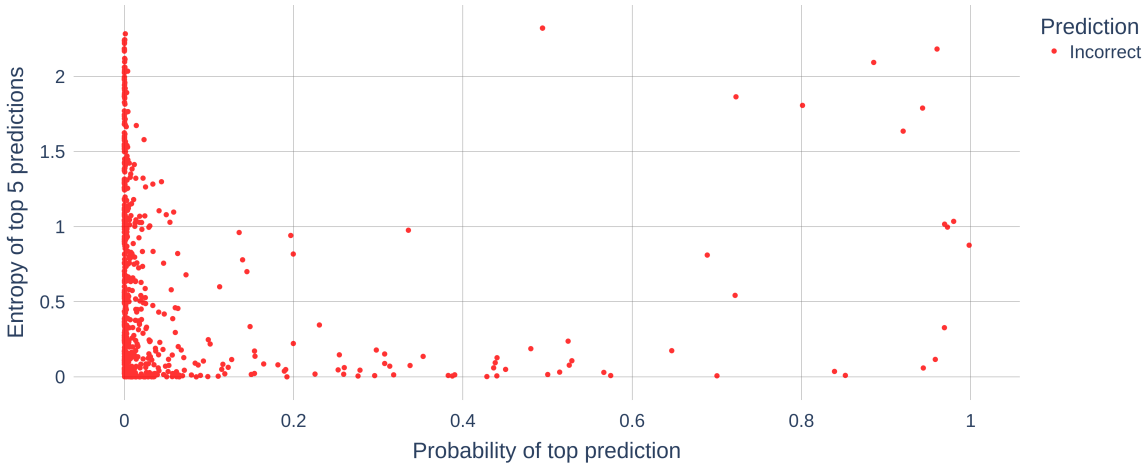


(a) Architecture A_1.

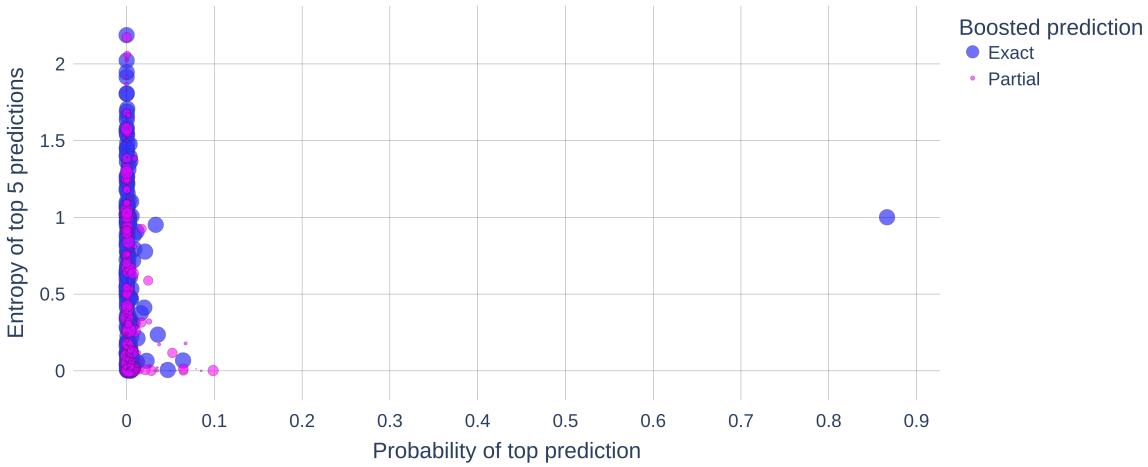


(b) Architecture A_2.

Figure 6.3. Distribution of correct and incorrect predictions with respect to probability and entropy of predictions.

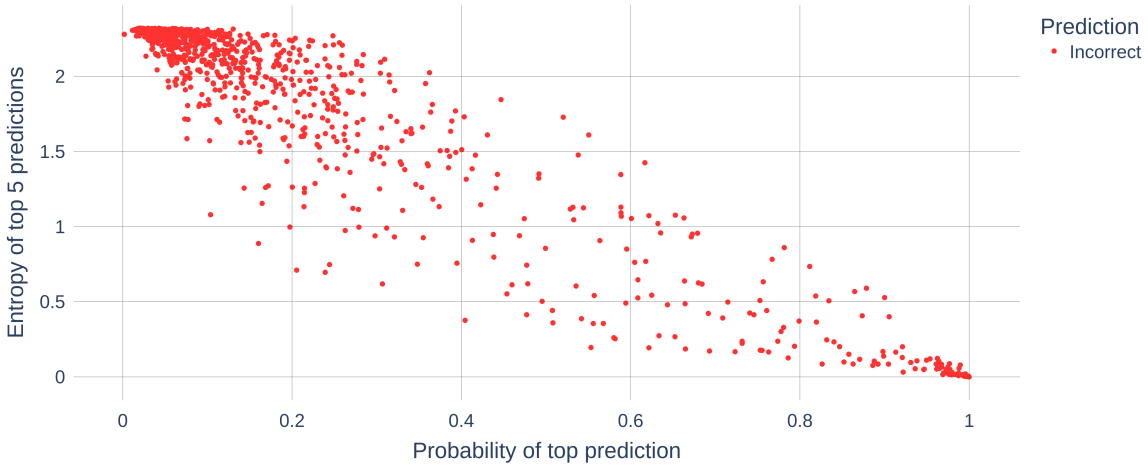


(a) Incorrect predictions.

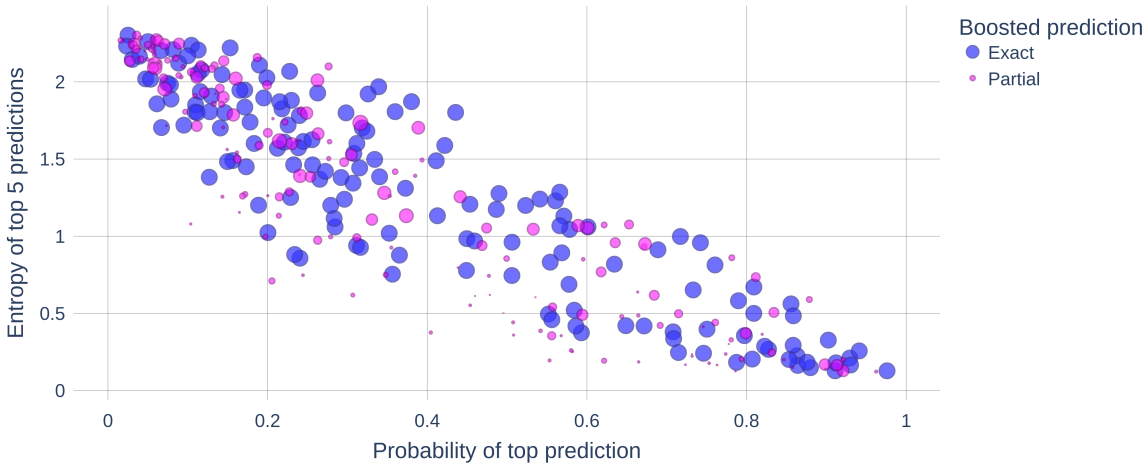


(b) Corrected predictions after boosting.

Figure 6.4. A_1 predictions corrected via Ontology Boosting.



(a) Incorrect predictions.



(b) Corrected predictions after boosting.

Figure 6.5. A_2 predictions corrected via Ontology Boosting.

Chapter 7: Conclusion and Future works

7.1 Conclusion

Using GRU as our deep learning architecture and biomedical literature as our corpus, we develop different models capable of automating ontology annotations. The results from these models underscore the positive impact they have in accuracy over other variants of existing deep learning models. The augmentation of biological information from different knowledge bases to those present in the gold standard corpus shows a distinct improvement in prediction accuracy.

We can further improve the model's performance by incorporating the ontology hierarchy into model training so that the model can make more accurate concept predictions for text. We show that our intelligent ontology-aware model results in higher annotation accuracy over a naive baseline model. Here, the focus is more on improving the model's accuracy in terms of semantic similarity than in terms of F1 score. This is because F1 metrics completely focus on accurate prediction and do not consider the significance of partially accurate prediction. Semantic similarity metrics, on the other hand, give a better evaluation of the model's performance for this approach. This work also paves the way for more sophisticated approaches for enabling deep learning architectures to gain an understanding of the ontology space and semantics.

As another approach to integrating the ontology hierarchy into concept prediction,

we present a novel approach called Ontology Boosting. It allows post-processing of ontology predictions by already trained deep learning models to selectively improve the confidence of certain predictions by using information from the ontology such as immediate parent or subsumers, information content, depth of a concept in the ontology, etc. We show that this computationally inexpensive step can result in substantial improvements to our key performance metric - semantic similarity. Our results clearly show that the predictions made by the deep learning model are closer to the human ground truth after applying the boosting process as compared to before.

Even though this study uses GO ontologies for all purposes, the scope of this work is not limited to just GO or even bio-ontologies. These approaches can be implemented in any other domain that has established a formal representation to describe their entities and relationship. These measures are also applicable in other deep learning architectures based on one's requirements.

7.2 Future works

We observe from the analysis of predicted ontologies that concepts/annotations that are underrepresented in the gold standard corpus i.e. there are present in lower frequency across the entire corpus are less confident in making predictions. Furthermore, most of the incorrect predictions have lower frequency terms as ground truth. To overcome this shortcoming, we can use synonyms of underrepresented terms from the GO ontology to increase their frequency. GO ontology already contains synonyms for all ontology concepts and thus can be used to properly represent all annotations in the corpus.

Additionally, we explore a few approaches to integrate the information from the ontology hierarchy to improve prediction accuracy. Capitalizing on ontology hierarchy, we can help the deep learning models learn low-dimensional presentation or graph embedding. Knowledge graph embedding or simply graph embedding is a deep learning algorithm to learn the low-dimensional representation of knowledge graph's entities and relationships while maintaining their semantic properties. Using these embedded representations, we can potentially improve the deep learning model's prediction accuracy. Furthermore, we can predict ontologies that are present outside our gold standard corpus, with good accuracy.

References

[1] L. Beasley and P. Manda, "Comparison of natural language processing tools for automatic gene ontology annotation of scientific literature," *Proceedings of the International Conference on Biomedical Ontology*, 2018.

[2] W. Dahdul, T. A. Dececchi, N. Ibrahim, H. Lapp, and P. Mabee, "Moving the mountain: analysis of the effort required to transform comparative anatomy into computable anatomy," *Database*, vol. 2015, 05 2015.

[3] D. Rebholz-Schuhmann, S. Kafkas, J.-H. Kim, C. Li, A. J. Yepes, R. Hoehndorf, R. Backofen, and I. Lewin, "Evaluating gold standard corpora against gene/protein tagging solutions and lexical resources," *Journal of Biomedical Semantics*, vol. 4, p. 28, Oct 2013.

[4] G. Petasis, F. Vichot, F. Wolinski, G. Paliouras, V. Karkaletsis, and C. D. Spyropoulos, "Using machine learning to maintain rule-based named-entity recognition and classification systems," in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, (USA), p. 426–433, Association for Computational Linguistics, 2001.

[5] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[6] M. Habibi, L. Weber, M. Neves, D. L. Wiegandt, and U. Leser, "Deep learning with word embeddings improves biomedical named entity recognition," *Bioinformatics*, vol. 33, no. 14, pp. i37–i48, 2017.

[7] C. Lyu, B. Chen, Y. Ren, and D. Ji, "Long short-term memory rnn for biomedical named entity recognition," *BMC bioinformatics*, vol. 18, no. 1, p. 462, 2017.

[8] X. Wang, Y. Zhang, X. Ren, Y. Zhang, M. Zitnik, J. Shang, C. Langlotz, and J. Han, "Cross-type biomedical named entity recognition with deep multi-task learning," *arXiv preprint arXiv:1801.09851*, 2018.

[9] P. Manda, S. SayedAhmed, and S. D. Mohanty, "Automated ontology-based annotation of scientific literature using deep learning," in *Proceedings of The International Workshop on Semantic Big Data*, SBD '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[10] P. Manda, J. P. Balhoff, H. Lapp, P. Mabee, and T. J. Vision, "Using the phenoscape knowledgebase to relate genetic perturbations to phenotypic evolution," *genesis*, vol. 53, no. 8, pp. 561–571, 2015.

[11] T. Groza, S. Köhler, D. Moldenhauer, N. Vasilevsky, G. Baynam, T. Zemojtel, L. M. Schriml, W. A. Kibbe, P. N. Schofield, T. Beck, *et al.*, "The human phenotype ontology: semantic unification of common and rare disease," *The American Journal of Human Genetics*, vol. 97, no. 1, pp. 111–124, 2015.

[12] H. Cho, W. Choi, and H. Lee, "A method for named entity normalization in biomedical articles: application to diseases and plants," *BMC Bioinformatics*, vol. 18, p. 451, Oct 2017.

[13] H. Liu, Z.-Z. Hu, J. Zhang, and C. Wu, "Biothesaurus: a web-based thesaurus of protein and gene names," *Bioinformatics*, vol. 22, pp. 103–105, 11 2005.

[14] O. Bodenreider, "The unified medical language system (UMLS): integrating biomedical terminology," *Nucleic Acids Research*, vol. 32, pp. 267D–270, Jan. 2004.

[15] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Association for Computational Linguistics, oct 2014.

[16] P. Manda, L. Beasley, and S. Mohanty, "Taking a dive: Experiments in deep learning for automatic ontology-based annotation of scientific literature," *Proceedings of the International Conference on Biomedical Ontology*, 2018.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] C. Pesquita, D. Faria, A. O. Falcão, P. Lord, and F. M. Couto, "Semantic similarity in biomedical ontologies," *PLOS Computational Biology*, vol. 5, pp. 1–12, 07 2009.

[19] G. Zehetner, "Ontoblast function: From sequence similarities directly to potential functional annotations by ontology terms," *Nucleic acids research*, vol. 31, no. 13, pp. 3799–3803, 2003.

[20] S. Khan, G. Situ, K. Decker, and C. J. Schmidt, "Gofigure: Automated gene ontology™ annotation," *Bioinformatics*, vol. 19, no. 18, pp. 2484–2485, 2003.

[21] S. Hennig, D. Groth, and H. Lehrach, "Automated gene ontology annotation for anonymous sequence data," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3712–3715, 2003.

[22] M. Deng, T. Chen, and F. Sun, "An integrated probabilistic model for functional prediction of proteins," *Journal of Computational Biology*, vol. 11, no. 2-3, pp. 463–475, 2004.

[23] M. Deng, Z. Tu, F. Sun, and T. Chen, "Mapping gene ontology to proteins based on protein–protein interaction data," *Bioinformatics*, vol. 20, no. 6, pp. 895–902, 2004.

[24] S. Letovsky and S. Kasif, "Predicting protein function from protein/protein interaction data: a probabilistic approach," *Bioinformatics*, vol. 19, no. suppl_1, pp. i197–i204, 2003.

[25] N. Nariai, E. D. Kolaczyk, and S. Kasif, "Probabilistic protein function prediction from heterogeneous genome-wide data," *Plos one*, vol. 2, no. 3, p. e337, 2007.

[26] Y. A. Kourmpetis, A. D. Van Dijk, M. C. Bink, R. C. van Ham, and C. J. ter Braak, "Bayesian markov random field analysis for protein function prediction

based on network data," *PloS one*, vol. 5, no. 2, p. e9293, 2010.

[27] A. Vinayagam, C. del Val, F. Schubert, R. Eils, K.-H. Glatting, S. Suhai, and R. König, "Gopet: a tool for automated predictions of gene ontology terms," *BMC bioinformatics*, vol. 7, no. 1, pp. 1–7, 2006.

[28] A. Lobley, M. B. Swindells, C. A. Orengo, and D. T. Jones, "Inferring function using patterns of native disorder in proteins," *PLoS computational biology*, vol. 3, no. 8, p. e162, 2007.

[29] J. Jung, G. Yi, S. A. Sukno, and M. R. Thon, "Pogo: Prediction of gene ontology terms for fungal proteins," *BMC bioinformatics*, vol. 11, no. 1, pp. 1–9, 2010.

[30] R. You, Z. Zhang, Y. Xiong, F. Sun, H. Mamitsuka, and S. Zhu, "Golabeler: improving sequence-based large-scale protein function prediction by learning to rank," *Bioinformatics*, vol. 34, no. 14, pp. 2465–2473, 2018.

[31] A. R. Aronson, "Effective mapping of biomedical text to the umls metathesaurus: the metamap program.," in *Proceedings of the AMIA Symposium*, p. 17, American Medical Informatics Association, 2001.

[32] C. Jonquet, N. Shah, C. H Youn, M. Musen, C. Callendar, and M.-A. Storey, "Ncbo annotator: Semantic annotation of biomedical data," 01 2009.

[33] H.-M. Müller, E. E. Kenny, P. W. Sternberg, and M. Ashburner, "Textpresso: an ontology-based information retrieval and extraction system for biological literature," *PLoS biology*, vol. 2, no. 11, p. e309, 2004.

[34] M. A. Casteleiro, G. Demetriou, W. Read, M. J. F. Prieto, N. Maroto, D. M. Fernandez, G. Nenadic, J. Klein, J. Keane, and R. Stevens, "Deep learning meets ontologies: experiments to anchor the cardiovascular disease ontology in the biomedical literature," *Journal of biomedical semantics*, vol. 9, no. 1, p. 13, 2018.

[35] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar, "Deep active learning for named entity recognition," *arXiv preprint arXiv:1707.05928*, 2017.

[36] Q. Zhu, X. Li, A. Conesa, and C. Pereira, "Gram-cnn: a deep learning approach with local context for named entity recognition in biomedical text," *Bioinformatics*,

vol. 34, no. 9, pp. 1547–1554, 2018.

[37] M. R. Boguslav, N. D. Hailu, M. Bada, W. A. Baumgartner, and L. E. Hunter, "Concept recognition as a machine translation problem," *BMC bioinformatics*, vol. 22, no. 1, pp. 1–39, 2021.

[38] N. D. Hailu, M. Bada, A. T. Hadgu, and L. E. Hunter, "Biomedical concept recognition using deep neural sequence models," *bioRxiv*, p. 530337, 2019.

[39] L. Furrer, J. Cornelius, and F. Rinaldi, "Uzh@ craft-st: a sequence-labeling approach to concept recognition," in *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pp. 185–195, 2019.

[40] D. Sousa, A. Lamurias, and F. M. Couto, "Using neural networks for relation extraction from biomedical literature," in *Artificial Neural Networks*, pp. 289–305, Springer, 2021.

[41] R. Xing, J. Luo, and T. Song, "Biorel: towards large-scale biomedical relation extraction," *BMC bioinformatics*, vol. 21, no. 16, pp. 1–13, 2020.

[42] S. Yadav, S. Ramesh, S. Saha, and A. Ekbal, "Relation extraction from biomedical and clinical text: Unified multitask learning framework," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[43] P. Devkota, S. D. Mohanty, and P. Manda, "A gated recurrent unit based architecture for recognizing ontology concepts from biological literature," *BioData Mining*, vol. 15, no. 1, pp. 1–23, 2022.

[44] P. Devkota, S. Mohanty, and P. Manda, "Knowledge of the ancestors: Intelligent ontology-aware annotation of biological literature using semantic similarity," *Proceedings of the International Conference on Biomedical Ontology*, 2022.

[45] O. Bodenreider and R. Stevens, "Bio-ontologies: current trends and future directions," *Brief. Bioinform.*, vol. 7, pp. 256–274, sep 2006.

[46] T. G. O. Consortium, "The gene ontology resource: 20 years and still GOing strong," vol. 47, pp. D330–D338, jan 2019.

[47] M. Bada, M. Eckert, D. Evans, K. Garcia, K. Shipley, D. Sitnikov, W. A.

Baumgartner, K. B. Cohen, K. Verspoor, J. A. Blake, and L. E. Hunter, "Concept annotation in the craft corpus," *BMC Bioinformatics*, vol. 13, p. 161, Jul 2012.

[48] L. Ramshaw and M. Marcus, "Text chunking using transformation-based learning," in *Third Workshop on Very Large Corpora*, p. 6, 1995.

[49] A. Segev and Q. Z. Sheng, "Bootstrapping ontologies for web services," *IEEE Transactions on Services Computing*, vol. 5, no. 1, pp. 33–44, 2010.

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

[51] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *CoRR*, vol. abs/1802.05365, 2018.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[53] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[54] M. Basaldella, L. Furrer, C. Tasso, and F. Rinaldi, "Entity recognition in the biomedical domain using a hybrid approach," *Journal of biomedical semantics*, vol. 8, no. 1, pp. 1–14, 2017.

[55] C. Funk, W. Baumgartner, B. Garcia, C. Roeder, M. Bada, K. B. Cohen, L. E. Hunter, and K. Verspoor, "Large-scale biomedical concept recognition: an evaluation of current automatic annotators and their parameters," *BMC bioinformatics*, vol. 15, no. 1, pp. 1–29, 2014.

[56] M. Tanenblatt, A. Coden, and I. Sominsky, "The conceptmapper approach to named entity recognition," in *Proceedings of the seventh international conference on language resources and evaluation (LREC'10)*, 2010.

[57] M. Sung, M. Jeong, Y. Choi, D. Kim, J. Lee, and J. Kang, "Bern2: an advanced neural biomedical named entity recognition and normalization tool,"

arXiv preprint arXiv:2201.02080, 2022.

[58] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017.

[59] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017.

[60] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[61] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, pp. 1–54, 2019.

[62] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.

[63] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 115–123, PMLR, 17–19 Jun 2013.