Ontology-Powered Boosting for Improved Recognition of Ontology Concepts from Biological Literature

Pratik Devkota[1][a], Somya D. Mohanty[1][b] and Prashanti Manda[2][c]
[1]*Department of Computer Science, University of North Carolina, Greensboro, NC, U.S.A.*
[2]*Informatics and Analytics, University of North Carolina, Greensboro, NC, U.S.A.*

Keywords: Named Entity Recognition, Automated Ontology Curation, Deep Learning, Biological Ontology.

Abstract: Automated ontology curation involves developing machine learning models that can learn patterns from scientific literature to predict ontology concepts for pieces of text. Deep learning has been used in this area with promising results. However, these models often ignore the semantically rich information that's embedded in the ontologies and treat ontology concepts as independent entities. Here, we present a novel approach called Ontology Boosting for improving prediction accuracy of automated curation techniques powered by deep learning. We evaluate the performance of our models using Jaccard semantic similarity – a metric designed to assess similarity between ontology concepts. Semantic similarity metrics have the capability to estimate partial similarity between ontology concepts thereby making them ideal for evaluating the performance of annotation systems such as deep learning where the goal is to get as close as possible to human performance. We use the CRAFT gold standard corpus for training our architectures and show that the Ontology Boosting approach results in substantial improvements in the performance of these architectures.

1 INTRODUCTION

Biological ontologies are critical for consistent knowledge representation that can be accessed by humans and computers alike. Since the introduction of the Gene Ontology, over 900 bio-ontologies have been created for representing knowledge in subdomains such as anatomy, human disease, chemicals and drugs, etc. Scientists and curators now use these ontology concepts to describe various aspects of biological objects thereby creating knowledge bases of ontology annotations. These annotations are critical for transferring knowledge in free text such as publications into a computationally amenable format that can power large-scale comparative analyses. Ontology annotation is still largely accomplished via human curation - a process where scientists manually read text and select ontology concepts that accurately represent the information in the text. While there has been a rapid growth in the number of ontologies as well as the number of ontology-powered annotations, ontology curation has not experienced the same level of advances.

Automated curation methods that can scale to the pace of publishing and show efficiency and accuracy are direly needed. The goal of these methods would be to process scientific literature and mark words or phrases in text with one or more ontology concepts thereby conducting automated curation. Automated curation tools can be used as stand-alone approaches that perform annotation without supervision or preliminary annotators that can make suggestions for human curators to accept or reject. In either case, it is important for the models to replicate the process of a human curator as closely as possible. One of the ways in which curators select appropriate ontology concepts is by looking at the concepts in the context of the ontology including the hierarchy and relationships in the ontology and not as concepts as individual constructs. This indicates that for automated methods to be successful at curation, they need to be cognizant of the ontology structure as well as relationships between different concepts.

The ultimate goal of automated ontology concept recognition is to develop intelligent systems that can understand the ontology hierarchy and make predictions that are cognizant of those relationships. For example, if a phrase in text corresponds to ontology

[a] https://orcid.org/0000-0001-5161-0798
[b] https://orcid.org/0000-0002-4253-5201
[c] https://orcid.org/0000-0002-7162-7770

Devkota, P., Mohanty, S. and Manda, P.
Ontology-Powered Boosting for Improved Recognition of Ontology Concepts from Biological Literature.
DOI: 10.5220/0011683200003414
In *Proceedings of the 16th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2023) - Volume 3: BIOINFORMATICS*, pages 80-90
ISBN: 978-989-758-631-6; ISSN: 2184-4305

concept X in the gold standard, the model should ideally recognize that X corresponds to the phrase. However, automated models are not perfect and sometimes make mistakes. In this case, the desired outcome would be for the model to recognize that semantically similar concepts to X (such as X's parent) exist in the ontology and that those concepts should be associated with the text. Ontology sentient models take the ontology structure or the ontology graph as input while training and make predictions accordingly. However, developing accurate ontology sentient models can be a challenge due to the size and complexities of the ontology graph that often results in models that are too large or require an inordinate time for training.

The automated annotation models previously developed by this team (Manda et al., 2018; Manda et al., 2020; Devkota et al., 2022b; Devkota et al., 2022a) have shown good accuracy in recognizing ontology concepts from text. In most cases, these systems are able to predict the same ontology concept as the ground truth in the gold standard data achieving perfect accuracy. However, ontology-based prediction systems can also achieve partial accuracy. This happens when a model might not predict the exact ontology concept as the gold standard but a related concept (sub-class or super-class), thereby achieving partial accuracy. In cases where our models were not able to predict the ground truth exactly, they failed to achieve reasonable partial accuracy and predicted concepts that were highly unrelated to the ground truth. Thus, our models' accuracy could be improved by focusing on improving the partial accuracy for instances when the model fails to make an exact prediction.

With the above motivation, here, we present an alternative approach called Ontology-powered Boosting (OB) to improve the prediction performance of automated curation models by using information about the ontology hierarchy to post-process the model's predictions after training has completed. 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.

Note that the contribution of this work is not in presenting novel architectures for recognizing ontology concepts but rather in presenting the Ontology Boosting approach for further improving prediction accuracy of our previously published deep learning architectures. Hence, we will present architecture details briefly and will refer the reader to our prior work for complete details.

2 BACKGROUND

Automated methods of recognizing ontology concepts in literature have been developed in the last decade and the approaches range from lexical analysis to traditional machine learning to deep learning in more recent times.

Text mining tools that use traditional machine learning based methods employ supervised learning techniques using gold standard corpora (Beasley and Manda, 2018). In 2018, we conducted a survey of ontology-based Named Entity Recognition and conducted a formal comparison of methods and tools for recognizing ontology concepts from scientific literature. Three concept recognition tools (MetaMap (Aronson, 2001), NCBO Annotator (Jonquet et al., 2009), Textpresso (Müller et al., 2004) were compared (Beasley and Manda, 2018).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 (Lample et al., 2016; Habibi et al., 2017; Lyu et al., 2017; Wang et al., 2018; Manda et al., 2020). 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 (Casteleiro et al., 2018). We evaluated the feasibility of using deep learning for the task of recognizing ontology concepts in a 2018 study (Manda et al., 2018). 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 new deep learning model/architecture based on combining multiple GRUs with a character+word based input. 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.

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, inclusion of ontology semantics via subsumption reasoning yielded modest performance improvement (Manda et al., 2020). This result indicated that more sophisticated approaches to take advantage of the ontology hierarchy are needed.

Continuing this work, a 2022 study (Devkota et al., 2022b) 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 (Manda et al., 2020). In 2022, we presented an intelligent annotation system (Devkota et al., 2022a) 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 choice of embedding) improvements over a baseline model that doesn't use ontology hierarchies.

3 METHODS

3.1 Ontology Boosting

A key component of our approach is to combine the prediction of the deep learning architectures with the graph structure of ontological concepts. Here we "boost" the predictions of the deep learning model by supporting them with similar predictions while taking model uncertainty into account. Here we take a two step approach, 1) identify candidates for boosting, and 2) boost the predictions with semantically similar concepts.

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 3, 3a), 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:

$$I(x_j) = -log(f_x/C)$$

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

where, we first calculate the information content ($I(x_j)$) of 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). $I(x_j)$ is then utilized to calculate the boosted probability, $\tilde{p}(x_j)$, which consists of two components, the modulated original probability ($\beta * p(x_j) * I(x_j)$) and supportive parent boosting ($\sum_{n=1}^{d} \frac{\alpha * p(x_j^n) * I(x_j^n)}{n}$).

The modulated original probability combines the original probability with a weighting factor β and the

information content of the concept $I(x_j)$. The second part of the computation evaluates all of the subsumer probabilities of x_j by individually calculating the modulated probabilities of parents $(\alpha * p(x_j^n) * I(x_j^n))$, 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 to 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 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 ($H(v_i)$), α and β to maximize the model prediction accuracy. Specifically, we utilize Tree-structured Parzen Estimators (TPE) (Bergstra et al., 2013) approach to derive the optimal values for each of 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 (Devkota et al., 2022b; Devkota et al., 2022a).

3.2 Training Dataset

This study uses GO annotations from version v4.0.1 (https://github.com/UCDenver-ccp/CRAFT/releases/tag/v4.0.1) of The Colorado Richly Annotated Full Text Corpus (CRAFT) (Bada et al., 2012), a manually annotated corpus containing 97 articles each of which is annotated to 10 ontologies.

3.3 Data Preprocessing

The following preprocessing steps are performed to translate annotations from the CRAFT corpus to the desired input formats for the deep learning models. Please refer to previous work (Devkota et al., 2022b) for full details.

3.3.1 Sentence Segmentation and Tokenization

Annotations for each CRAFT article are recorded in the corresponding `xml` annotations file via character index spans. In order 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.).

3.3.2 IOB Tagging

Each extracted word/token is mapped to a GO term or an *out-of-concept* annotation. Each token is mapped to one of three tags: 1) `GO` to indicate an annotation, `'O'` for a non-annotation, and `'EOS'` to indicate the end of sentence.

3.3.3 POS Tagging and Token Encoding

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), and space.

We represent character level aspects of a token using character encodings. These encodings represent upper-case and lower-case characters with 'C' or 'c' respectively. Numbers are represented using an 'N' and punctuation (such as commas, periods, and dashes) are retained in the encoding.

3.3.4 BioThesaurus Encoding

One of the architectures tested in this study uses external information from existing large scale knowledge bases. The first data source we use is BioThesaurus (Liu et al., 2006), which is a database of protein and

gene names mapped to the UniProt Knowledgebase. If a token is present in the database, we map if it identifies as a protein name, biomedical terms, chemical terms, and/or macromolecule.

3.3.5 Unified Medical Language System (UMLS) Encoding

Next, we query the UMLS (Lindberg et al., 1993) database for tokens extracted from the articles. 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`.

3.4 Deep Learning Architecture

We evaluate our boosting approach on two deep learning architectures from our prior work (Devkota et al., 2022b; Devkota et al., 2022a). We describe the two architectures briefly here and refer the reader to the articles for comprehensive details.

3.4.1 Architecture 1 - Externally Augmented Predictor (A_1)

Figure 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 (Devkota et al., 2022b).

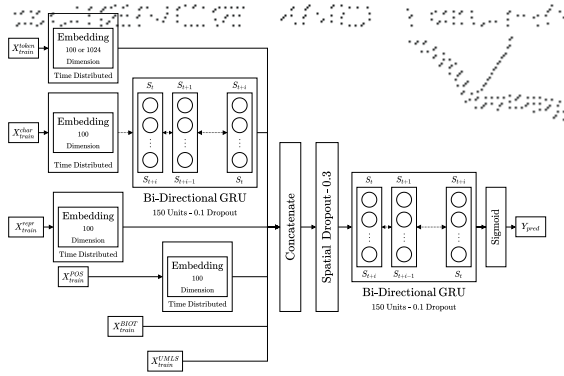


Figure 1: Architecture of a GRU based model for ontology concept recognition. Figure originally published in (Devkota et al., 2022b).

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

The token (X_{train}^{token}) input, is a sequential tensor where each token is represented with a high dimensional one hot encoded vector. Similarly, the character sequence (X_{train}^{char}) is also a sequential tensor consisting of character sequences present in a word/token.

Character representations (X_{train}^{repr}) and POS tags (X_{train}^{POS}) are based on words/tokens in sentences. 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 an one hot encoded vector sequence, where `1` indicates a token's presence and `0` indicates absence in UMLS.

Embedding/Latent Representations. The supervised embedding is a bottleneck layer which learns to map the one hot encoded input into a smaller dimensional representation. We used ELMo (Peters et al., 2018) pretrained embeddings for the X_{train}^{token} input. Embeddings in ELMo are learned via a bidirectional language model where the sequence of the words are 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 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 a 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, and 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 (Kingma and Ba, 2017) as our optimiser for all of the experiments with a default learning rate of 0.0001.

3.4.2 Architecture 2 - Intelligent Predictor (A_2)

A_2, as shown in figure 2 uses an intelligent prediction system by using the ontology hierarchy structure as opposed to A_1 (Devkota et al., 2022a). This architecture was originally published in (Devkota et al., 2022a). 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.

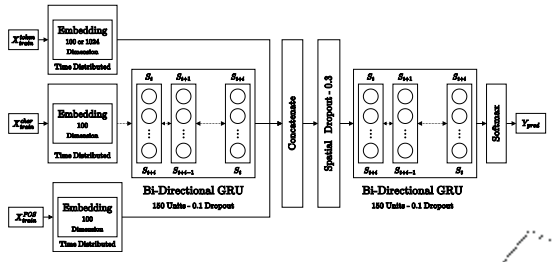


Figure 2: Architecture of an intelligent ontology prediction system based on GRUs. Figure originally published in (Devkota et al., 2022a).

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}).

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 (as in the case of A_1). In our studies, the output labels correspond to the set of all GO terms. Typically, the value of the GO term to be predicted is set to a 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 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.

So for each output tag, the representation is as follows:

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

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 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 $\widehat{\mathcal{T}}$ (Pesquita et al., 2009) is calculated as:

$$\mathcal{J}_{sim}(\mathcal{T}, \widehat{\mathcal{T}}) = \frac{|S(\mathcal{T}) \cap S(\widehat{\mathcal{T}})|}{|S(\mathcal{T}) \cup S(\widehat{\mathcal{T}})|} \tag{2}$$

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 their union of their subsumers (Pesquita et al., 2009).

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 (Loshchilov and Hutter, 2017) as our optimiser 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 (Lin et al., 2017) 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 bal-

ancing factor and β or modulating factor, which are set to 0.25 and 2.0 respectively.

3.5 Performance Evaluation Metrics

Our primary evaluation metric in this study is semantic similarity. Metrics such as F1 are designed for traditional information retrieval systems that either retrieve a piece of information or fail to do so (a binary evaluation). However, this is not a true indication of the performance of ontology-based retrieval or prediction systems where the notion of partial accuracy applies. A model might not predict the exact concept as a gold standard but might predict the parent or an ancestor of the ground truth as indicated by the ontology. The parent will have a higher degree of similarity to the ground truth thereby resulting in high semantic similarity (but would count as a failure for F1 computation). Semantic similarity metrics (Pesquita et al., 2009) 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 (Pesquita et al., 2009) that measures the ontological distance between two concepts to assess partial similarity.

We also provide a modified F1 score for our architectures. 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. 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.

4 RESULTS AND DISCUSSION

Table 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%.

We examine the impact of our boosting approach on improving prediction accuracy of the two architectures presented above (Table 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 the A_2. The F1 scores experience a modest improvement of 2.5% for A_1 and no improvement for A_2.

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 3a and 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 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 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 reflect 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.

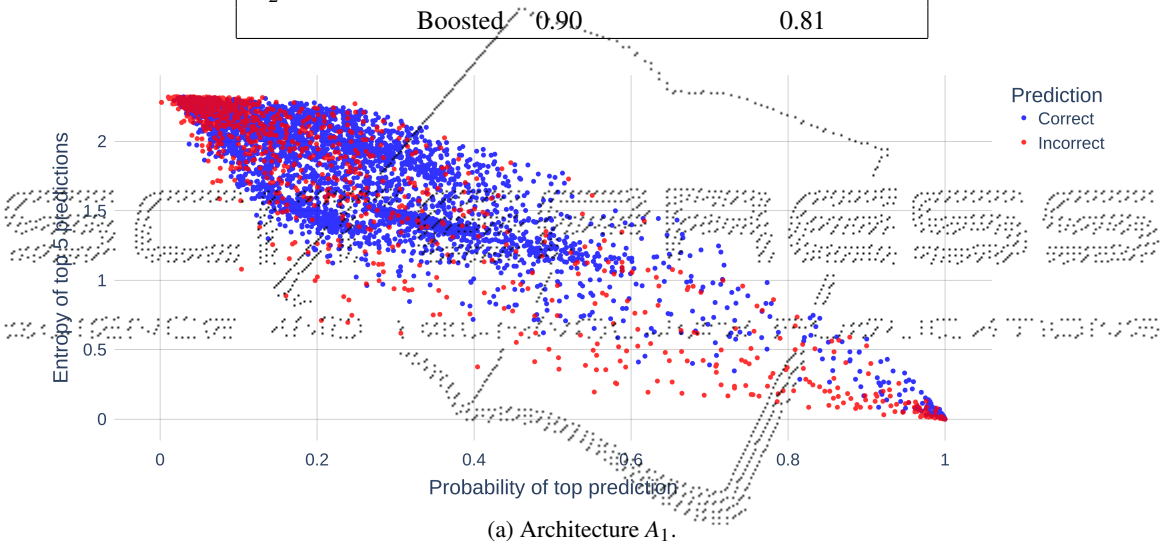
Figure 5a shows the incorrect predictions by A_2. The majority of these instances were cases where the

Table 1: Effect of ontology boosting on the two architectures.

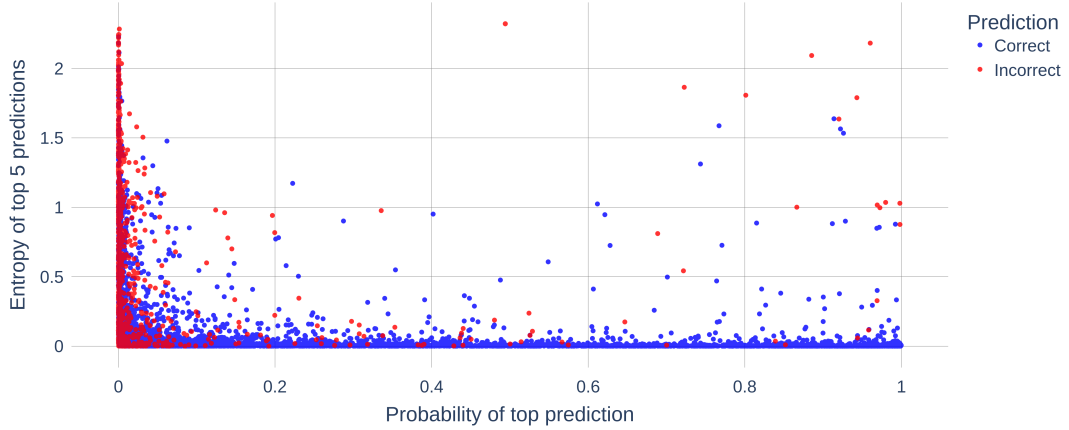
Row	Description	A_1	A_2
1	Total number of tokens	5439	5495
2	Number of tokens boosted	4972	5197
3	Number of tokens boosted but unchanged	4411	4587
4	Number of tokens boosted from GO to O	0	0
5	Number of tokens boosted from O to GO	534	366
6	Number of tokens boosted from O to an exact GO	197	152
7	Average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

Table 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

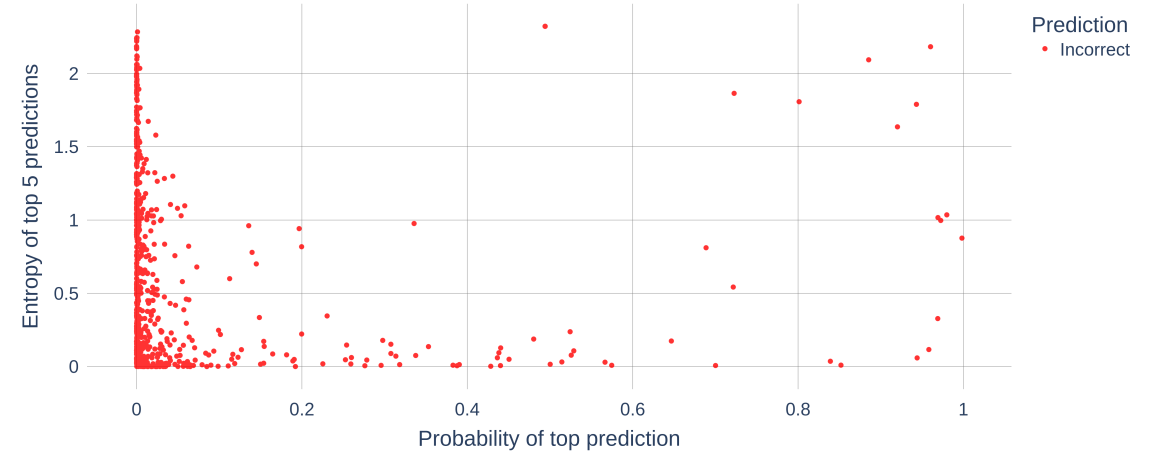


(a) Architecture A_1.

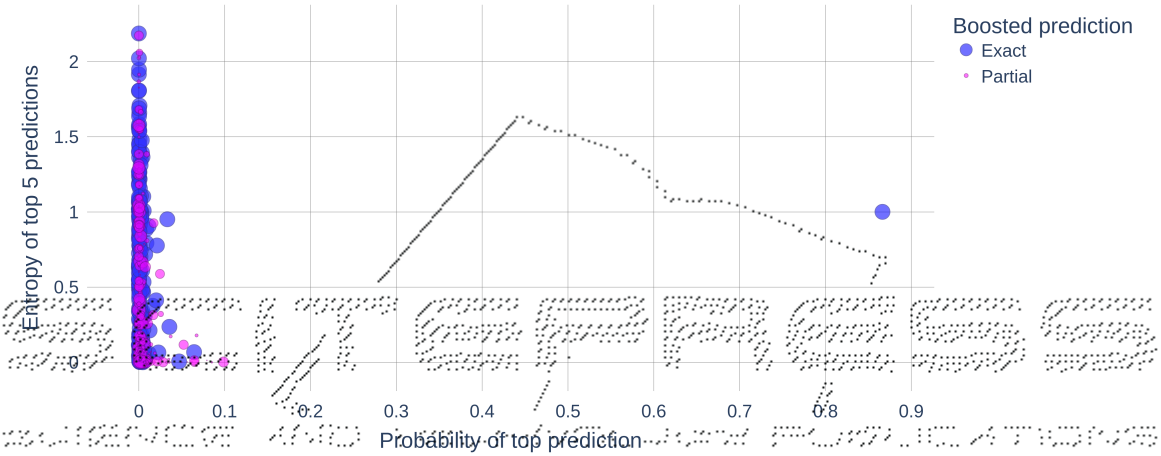


(b) Architecture A_2.

Figure 3: Distribution of correct and incorrect predictions with respect to probability and entropy of predictions.



(a) Incorrect predictions.



(b) Corrected predictions after boosting.

Figure 4: A_1 predictions corrected via Ontology Boosting.

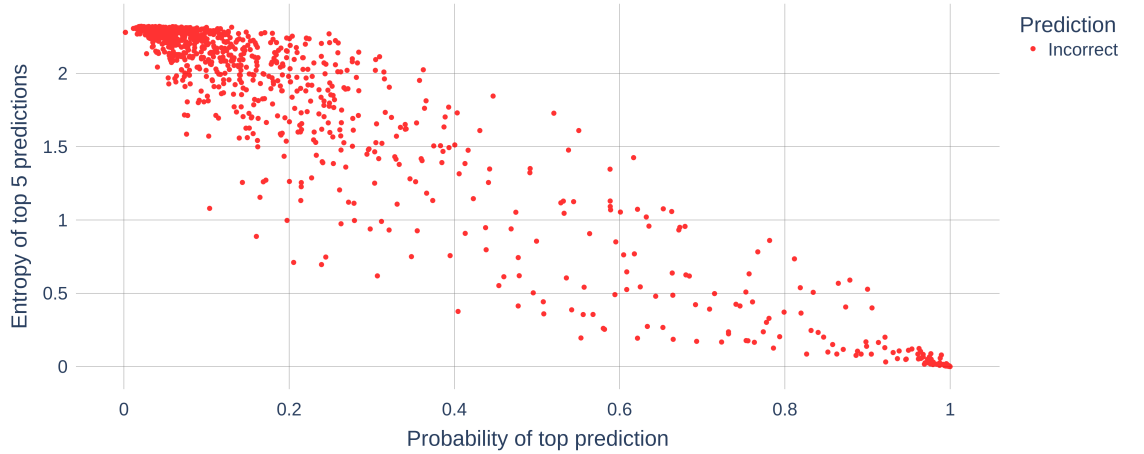
ground truth is a GO concept and the prediction is an 'O' term (non-annotation). Figure 5b shows instances incorrectly predicted as 'O' that were corrected by boosting.

5 CONCLUSIONS

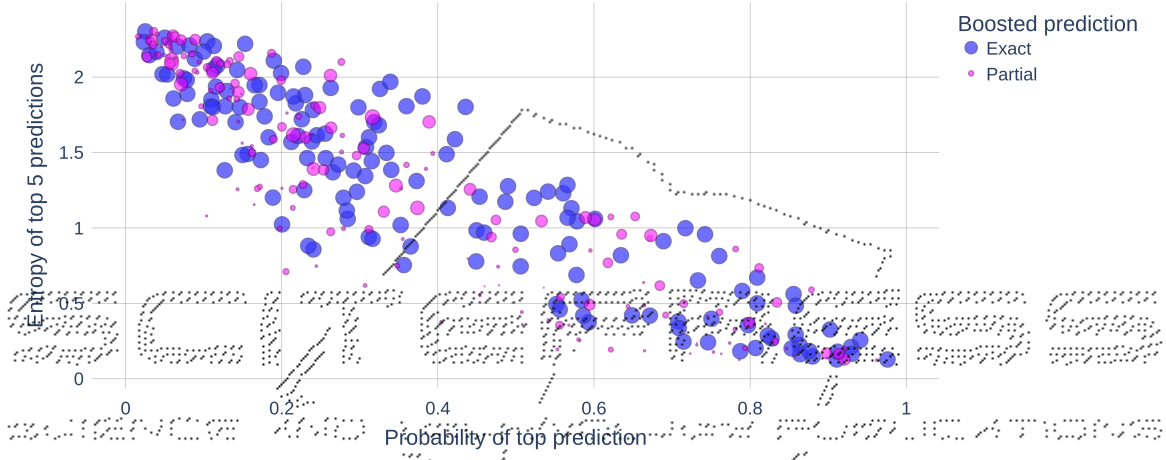
Automated methods for ontology-based annotation of scientific literature are important for representing knowledge in a consistent machine-readable format. Deep learning models have shown promising improvements and performance at this task. However, the majority of these architectures do not account for or take advantage of the ontology hierarchy thereby losing valuable information. Encoding and representing complex ontologies can lead to massive models that are computationally and financially infeasible to train. Here, we presented a novel approach called Ontology Boosting that allows post processing of ontology predictions by deep learning architectures to selectively improve the confidence of certain predictions by using information from the ontology such as 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.

ACKNOWLEDGEMENTS

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



(a) Incorrect predictions.



(b) Corrected predictions after boosting.

Figure 5: A_2 predictions corrected via Ontology Boosting.

REFERENCES

Aronson, A. R. (2001). Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association.

Bada, M., Eckert, M., Evans, D., Garcia, K., Shipley, K., Sitnikov, D., Baumgartner, W. A., Cohen, K. B., Verspoor, K., Blake, J. A., and Hunter, L. E. (2012). Concept annotation in the craft corpus. *BMC Bioinformatics*, 13(1):161.

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

Bergstra, J., Yamins, D., and Cox, D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, Atlanta, Georgia, USA. PMLR.

Casteleiro, M. A., Demetriou, G., Read, W., Prieto, M. J. F., Maroto, N., Fernandez, D. M., Nenadic, G., Klein, J., Keane, J., and Stevens, R. (2018). Deep learning meets ontologies: experiments to anchor the cardiovascular disease ontology in the biomedical literature. *Journal of biomedical semantics*, 9(1):13.

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

Devkota, P., Mohanty, S. D., and Manda, P. (2022b). A gated recurrent unit based architecture for recognizing ontology concepts from biological literature. *BioData Mining*, 15(1):1–23.

Habibi, M., Weber, L., Neves, M., Wiegandt, D. L., and Leser, U. (2017). Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48.

Jonquet, C., Shah, N., H Youn, C., Musen, M., Callendar,

C., and Storey, M.-A. (2009). Ncbo annotator: Semantic annotation of biomedical data.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

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

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection.

Lindberg, D. A., Humphreys, B. L., and McCray, A. T. (1993). The unified medical language system. *Yearbook of Medical Informatics*, 2(01):41–51.

Liu, H., Hu, Z.-Z., Zhang, J., and Wu, C. (2006). Biothesaurus: a web-based thesaurus of protein and gene names. *Bioinformatics*, 22(1):103–105.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization.

Lyu, C., Chen, B., Ren, Y., and Ji, D. (2017). Long short-term memory rnn for biomedical named entity recognition. *BMC bioinformatics*, 18(1):462.

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

Manda, P., SayedAhmed, S., and Mohanty, S. D. (2020). Automated ontology-based annotation of scientific literature using deep learning. In *Proceedings of The International Workshop on Semantic Big Data*, pages 1–6.

Müller, H.-M., Kenny, E. E., and Sternberg, P. W. (2004). Textpresso: An ontology-based information retrieval and extraction system for biological literature. *PLOS Biology*, 2(11).

Pesquita, C., Faria, D., Falcao, A. O., Lord, P., and Couto, F. M. (2009). Semantic similarity in biomedical ontologies. *PLoS computational biology*, 5(7).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *CoRR*, abs/1802.05365.

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