Term Tagger for the NLP domain

Silviya Silwal
University of Lorraine

Averie (Ho Zoen) So
University of Lorraine

Abstract

Understanding domain-specific vocabulary is especially challenging but important for a rapidly growing field like Natural Language Processing (NLP). This paper presents a NLP domain-specific term tagger in order to facilitate communication among specialists of different sub-fields in NLP as well as the wider scientific community. In this paper, we present a gold-annotated dataset for the NLP domain, as well as a model which leverages a pre-trained language model, BERT, for the task of term recognition. The BERT model outperforms various baselines, including off-the-shelf models from NLTK and Spacy.

1 Introduction

Term Recognition is a task in natural language processing (NLP) that involves identifying and classifying term entities in text, within a specific domain. When applied to a specialist domain, term recognition can make significant contributions to the field as it can facilitate communication among specialists as well as with the wider scientific community. More specifically, term recognition can help to identify terms that require standardization across specialists, or to aid those who are in other (sub-)fields to better understand the key notions in domain specific documents. Furthermore, term tagging/recognition has also been found to be useful to a wide range of other downstream tasks, including information extraction, question answering, and text summarization.

As of 2022, NLP is a rapidly growing, interdisciplinary field with a large number of subfields, which means that many new terms are constantly appearing, making it particularly difficult to catch up with the vast amount of literature. This project aims to build a domain-specific term tagger for the NLP domain in an effort to aid specialists and the wider scientific community when reading publications in the NLP domain, which often contain many new and unfamiliar terminologies.

In this paper, we describe our approach to Term Recognition using BERT, and evaluate its performance on a gold-annotated dataset in the NLP domain.

2 Dataset

2.1 Procedure

The dataset for this project was gathered and annotated collaboratively by two-person groups, both of whom are masters students in NLP; each group was assigned a year and tasked with selecting a minimum of ten abstracts to annotate using the IOB annotation scheme. The abstracts were selected from the MIT Press Direct website, more specifically from articles under the Computational Linguistics publications. Abstracts from each year were then split into train (80%), test (10%) and development (10%) sets. Each group member annotated the article abstracts individually, resulting in two annotated texts. The task of resolving disagreements and producing a final annotated text from a different year was then assigned to each group. So, for each article abstract, there were three annotations: the annotations done by annotator1 and annotator2 of one group, and the final annotation produced by members of the other group.

2.1.1 IOB scheme

The IOB (Inside, Outside, Beginning) tagging scheme (Ramshaw and Marcus, 1999) is a

common way of representing the chunks of text that are extracted from a sentence during named entity recognition (NER) tasks. The IOB format is a sequence of tags, where each tag is represented by a 3-letter code that indicates whether the corresponding word is inside, outside, or the beginning of a named entity. For example, the sentence "The quick brown fox jumped over the lazy dog" might be annotated with the following IOB tags: "O O B I O O O O O". This format is widely used because it is simple to understand and easy to use with various NLP libraries and tools. It is also useful for evaluating the performance of NER models, as it provides a clear way of comparing the predicted named entities with the ground truth annotations.

2.1.2 Annotation guidelines

Besides the IOB scheme, a few other simple guidelines were agreed upon among all annotators:

- After obtaining the abstract, use the default spacy tokenizer and sentence segmenter to obtain the tokens
- Include any part-of-speech for the term (eg. "***the/B*** *named/I entity/I recognition/I model/I*" instead of "***the/O*** *named/B entity/I recognition/I model/I*")
- Annotate only contiguous terms, so a sequence like "B O I I" would not be allowed
- No recursive annotation and annotate only the longest form of the term (eg. tag "*natural language processing*" even though "*natural language*" is also a term)
- In case of big doubts for a candidate, do not annotate.

2.2 Annotation quality

2.2.1 Inter-annotator agreement

To investigate the quality of the gold-annotated dataset, we use Cohen's Kappa coefficient κ to measure inter-annotator agreement. The kappa statistic is the most commonly used for this purpose (Viera et al., 2005). Compared to simply calculating the percentage of agreement, the kappa statistic takes into account the expected chance of agreement among two annotators. We use the score function from sklearn and the results are shown in Table 1.

According to the interpretation chart, the overall inter-annotator agreement is considered to be above *moderate agreement* (0.629). Out of the 13 groups, there are some variation in inter-annotator agreement, ranging from *fair agreement* (0.392) to *substantial agreement* (0.793). Within our own group, there is a relatively low inter-annotator agreement at *fair agreement* (0.398).

year	**#**	κ
2009	10	0.684
2010	9	0.686
2011	9	0.697
2012	10	0.659
2013	10	0.392
2014	10	0.489
2015	10	0.749
2016	**10**	**0.398**
2017	5	0.587
2018	10	0.782
2019	10	0.622
2020	10	0.793
2021	10	0.641
overall	123	0.629

Table 1: Inter-annotator agreement by Cohen's Kappa coefficient κ. The # indicates the number of documents that were taken into account while scoring. Boldface indicates the inter-annotator agreement of our group (2016). Interpretation chart: 0 - Less than chance agreement, 0.01–0.20 - Slight agreement, 0.21– 0.40 - Fair agreement, 0.41–0.60 - Moderate agreement, 0.61–0.80 - Substantial agreement, 0.81–0.99 - Almost perfect agreement

2.2.2 Errors in human annotation

While computing inter-annotator agreement, we found that a portion of initial annotation pairs did not match in lengths, which were most likely manually corrected by the final annotators. As seen in Table 1, the # indicates the number of documents where the two pairs

of initial annotations are of identical or comparable lengths. A portion of annotation pairs had similar lengths, so for simplicity we allow a difference of $\pm$ 2 tokens, and simply append "O" (non-term) in the shorter annotation in order to calculate the kappa coefficient. However, annotations from some years had significant differences in the number of tokens annotated and therefore could not be included in the inter-annotator agreement calculations (eg. year 2010, 2011, 2017).

Furthermore, we discovered instances of illegal prediction sequences in the gold dataset, which were most likely the result of misunderstandings about how to apply the annotations, or simply errors made while following the guidelines. For example, the term "continuum of compositionality" was tagged as B O I, which is an illegal span and should be tagged as B I I instead. These errors can have serious consequences because they can lead to data inaccuracies and potentially invalidate the results of any analyses or models built using the annotated data. Therefore, to ensure accuracy, it is critical that we carefully design and communicate the annotation guidelines, as well as thoroughly review and validate the annotations.

3 Experiments

3.1 Data Preprocessing

When we obtained the dataset for our term recognition task, we noticed that a significant portion of the labels were incorrect or inconsistent. This was likely due to errors in the manual annotation process, or differences in the annotators' interpretations of the guidelines. In order to ensure that the dataset was suitable for training and testing we decided to do some cleaning. This involved making sure that the labels present in the data were of the IOB scheme, so for example we had some cases of tokens that were labeled as 0 (zero), 'b' (lowercase) or 'II' which needed to be modified into the correct labels. Additionally, to ensure proper structure for our dataset we gathered all files from Gitlab repository and stored data in csv format.

3.2 Baseline models

We select three baselines models for comparison. This includes the Max rule algorithm as well as two off-the-shelf models, NLTK and Spacy.

Max rule The Max rule is a simple baseline approach for named entity recognition (NER) tasks that involves assigning each word in a sentence to the named entity class that has the highest conditional probability. In other words, for each word in the sentence, the Max rule predicts the named entity class that is most likely to be associated with that word, based on its probability distribution over the named entity classes. This approach is simple to implement and can be used as a baseline for comparing the performance of more complex NER models. However, this rule does not take into account the context or dependencies between words in a sentence, so it may not perform well on complex or noisy input data.

NLTK The NLTK named entity recognition model utilises POS tags and chunking to identify terms. By the NLTK definition, named entities are definite noun phrases[1]. As such, the process of chunking identifies the entire spans of definite noun phrases and detect named entity recognition only from those spans. In terms of the NER model itself, NLTK provides a classifier that has already been trained to recognize named entities.

Spacy Spacy is another leading open-source library for NLP tasks. For Named Entity Recognition, their model is a pre-trained, transition-based algorithm[2]. The Spacy model optimizes on whole entity accuracy, which means that it would not perform well if the inter-annotator agreement on entity boundary is low. The transition-based property means that it assumes the most decisive information is at the initial tokens of each entity span. It is also made to be trainable. For our experiments, we use the off-the-shelf "en_core_web_sm" model without finetuning.

[1] https://www.nltk.org/book/ch07.html
[2] https://spacy.io/api/entityrecognizer

3.3 BERT model

For our main model we used BertForTokenClassification which is part of the transformers library[3] by HuggingFace. It is a type of pre-trained language model based on BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) that is specifically designed for token-level classification tasks. In other words, it can take in a piece of text and assign a classification label to each individual word (or "token") in the text. This is useful for tasks like named entity recognition, where the goal is to identify and classify named entities (such as people, organizations, and locations) in a given text. BERT is trained on a large corpus of text, so it is able to capture the context and meaning of words in a sentence, which helps it make more accurate predictions.

The first process of using BertForTokenClassification for our term recognition task is to convert our input text into appropriate format that is compatible with the model. This involves tokenizing the text into individual words or sub-word units, and then converting the tokens into numerical indices that represent their corresponding entries in BERT's vocabulary. The challenge in this part is that BERT uses a technique called Wordpiece tokenization which breaks words into smaller sub-units called "wordpieces". This is useful when BERT needs to model words that are not in its vocabulary. A word like "book" could be tokenized as "boo" or "k". For our task, we want the model to predict the labels for each full-token rather than the subtokens. As a result, we make a design decision similar to the one seen in the original BERT paper: we only create token labels for the first wordpieces of a tokenized word.

To prepare our dataset for training and evaluation we define a Pytorch dataset class where we use BertTokenizerFast to encode sentences using padding and truncation upto max length. We make use of the "return_offset_mapping" functionality in the tokenizer to get the locations of the start and end of our token and sub-tokens. Then, we use this information to create labels only for the first word pieces of a tokenized word. Once the labels for the tokens have been assigned based on their offset position, our dataset is ready for training and evaluation.

[3]https://github.com/huggingface/transformers

3.4 Experimental settings

For our experiment, we defined our BertForTokenClassification model with predefined weights from "bert-based-uncased", as well as the number of labels in our training dataset, which determines the architecture of the classification head. We define *Adam* optimizer with a learning rate of 1e-4 and use *Cross Entropy* for our loss function. Following that, we define a training and validation function, which is partly based on a repository about Multilingual-NER, and track the results for a total of seven epochs.

3.4.1 Evaluation metrics

The term recognition models are automatically evaluated with standard performance metrics: precision, recall and F1. This evaluation is done at the token level as well as the span level. The token-based measure determines the exact matches in terms of IOB tags. The span-based measure determines whether the whole span of a term is correct or not, this is done by matching the starting and ending indices of each term in a sentence.

4 Results

4.1 Automatic evaluation

	Precision		**Recall**		**F1**	
Models	token	span	token	span	token	span
Max rule	0.25	-	0.33	-	0.28	-
Spacy	0.65	0.37	0.39	0.09	0.39	0.15
NLTK	0.73	0.32	0.39	0.06	0.39	0.10
BERT	**0.82**	**0.57**	**0.82**	**0.64**	**0.82**	**0.60**

Table 2: Precision, Recall and F1 scores by token and span. The Max rule model does not have span performance since no terms are predicted.

The term recognition performance of the various models measured by precision, recall and F1 are presented in Figure 2. Overall, the

BERT model outperforms the three other baselines, across token- and span-based measures. Among the baseline models, both off-the-shelf models (NLTK and Spacy) have comparable performance and both outperform the Max rule model.

4.2 Qualitative analysis

To further investigate the types of errors made by the various models, we conduct error analysis based on the methods in Ichihara et al. (2015) and Nguyen et al. (2019). We categories errors into 5 categories, which are **missing**, **wrong tag**, **wrong range**, **extra wrong range**, **extra no match** respectively. This adds up to 6 categories including the system outputs that are completely correct. We describe the error categories below:

missing refers to terms that only exists in the reference but there are no overlapping terms in the system output.
wrong tag refers to terms where the spans are identical in the reference and system output, but at least one tags within the span does not match.
wrong range refers to terms where the reference and system output have overlapping spans, but do not match exactly.
extra wrong range refers to the number of terms in "wrong range" whenever one reference term is mapped onto more than one system output terms, or vice versa.
extra no match refers to terms that only exists in the system output but there are no overlapping terms in the reference.

Table 4 and 5 demonstrate two example sentences and the respective tagging errors from the models. Note that the computations for *extra wrong range* do not correspond to exact terms extracted, but simply a count of how many extra terms are tagged within the same span of one reference term. For example, in sentence 2, the BERT model tags "word", "prefixes", and "suffixes" as B, and all are equally identified as *wrong range*, while only two (which does not refer to any specific term) would be counted as *extra wrong range*. Additionally, *extra wrong range* terms are categorically different from *extra no match*, as in "model" in sentence 2. This is because the extra term does not overlap with any span of reference terms.

Table 3 shows the overall distributions of error types in different models. The error types made by our model with BERT appear to be qualitatively different from the two off-the-shelf models. The majority of errors of the BERT model is *wrong range*, while NLTK and Spacy missed most of the terms. Across all models, *wrong tag* is the least frequent error, this is not surprising as this type of error usually only involves single token terms and thus are rarer. In terms of extra terms, BERT has made this error significantly more than the other two models - this also means that BERT extracts a lot more terms than off-the-shelf models.

5 Discussion

We present a manually annotated corpus for term tagging in the NLP domain and we compare our model, which leverages BERT word embeddings, against three other baselines models with various approaches. We find that our model outperforms all other baselines, in terms of both token and span based measures. From the qualitative analysis, we find that our model most commonly extracts the wrong spans when compared to the reference terms, while other off-the-shelf models usually completely miss the terms (no overlapping). Our model also has a tendency to identify a lot more terms (overlapping or not) than the reference terms, which is not a feature shared with the other off-the-shelf models.

Other than word embeddings from BERT, our model has also uniquely benefited from being trained on the domain-specific dataset. This factor is significant because other models have very similar performance despite dissimilar architectures: NLTK relies on POS tagging and chunking, while Spacy is pre-trained and transition-based.

Across the two baseline models, while NLTK has better performance than Spacy in

model	correct count	correct rate %	missing count	missing rate %	wrong tag count	wrong tag rate %	wrong range count	wrong range rate %	+ wrong range count	Total count	+ no match count
NLTK	20	6.0	278	83.7	1	0.3	39	11.7	6	332	11
Spacy	31	9.3	283	85.2	0	0.0	19	5.7	1	332	34
BERT	207	62.3	46	13.9	5	1.5	87	26.2	13	332	80
Total	258	25.9	607	60.9	6	0.6	145	14.6	20	996	125

Table 3: Count and rate of error types across models. + wrong range / + no match refer to *extra wrong range* and *extra no match* respectively. The total count is calculated by: correct + missing + wrong tag + wrong range - extra wrong range. The rates are the count of error type divided by the total number of reference terms, similar to recall. However, note that the correct rates here are slightly different from Table 2 Recall since "wrong tag" would be considered correct in the span-based automatic measure.

	sentence 1								
Models	...neural	NLP	technologies	,	a	factorization	-	based	parser
reference	B	I	I	O	O	B	I	I	B
NLTK	O	B	O	O	O	O	O	O	O
Spacy	O	B	O	O	O	O	O	O	O
BERT	B	I	I	O	O	B	O	I	I

Table 4: Error types for example sentence 1. green indicates **correct** terms. orange indicates terms categorized as **wrong range**. yellow indicates terms categorized as **wrong tag**. gray indicates terms categorized as **missing**.

token precision, Spacy outperforms NLTK in span-based measures. This is likely attritable to the design in Spacy's model in which whole entity accuracies are prioritized. In error analysis, it was found that both models suffer from similar types of errors, which is completely missing the term span. This is likely attributable to the lack of finetuning on our domain-specific data. In particular, NLTK only recognises definite noun phrases, which could have significant impact when a word is unseen and could not be identified as a noun.

In terms of the manually annotated dataset, we find various limitations, including differences in annotator formatting and agreement and illegal strings. This suggests that the procedure of obtaining manual data could have been improved, for example, by performing tokenization centrally (instead of every annotator doing it themselves) as well as adding a step which requires the two annotators to check the file formats before submitting the annotations. Additionally, we hypothesise that eliminating illegal strings from the dataset would have improved certain types of errors, such as *wrong tag* and *wrong range*.

In order to more robustly confirm the most important factors in improving term tagging performance, we believe that future experiments should be conducted with more comparable models, such as those that are also fine-tuned on the domain specific data. Conducting ablation studies would also be beneficial. Additionally, more in-depth error analysis could be done, such as by identifying the POS distribution of errors, or whether the error spans are more common at initial tokens or ending tokens.

6 Conclusion

We present a term tagged dataset and model for the NLP domain. Our experiments show that our model, with BERT, outperforms baselines of various architectures. We attribute finetuning to be the most important factor in improving term tagging performance. However, our methodology could also be improved in various aspects, which could potentially remove the bottleneck of the various off-the-shelf models. This includes improving the quality of the manually annotated dataset.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of

	sentence 2						
Models	the	model	...	word	prefixes	and	suffixes
reference	O	O	...	B	I	I	I
NLTK	O	O	...	O	O	O	O
Spacy	O	O	...	O	O	O	O
BERT	O	B	...	B	B	O	B

Table 5: Error types for example sentence 2. green indicates **correct** terms. orange indicates terms categorized as **wrong range**. pink indicates terms categorized as **extra no match**. purple indicates terms categorized as **extra wrong range**.

deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Masaaki Ichihara, Kanako Komiya, Tomoya Iwakura, and Maiko Yamazaki. 2015. Error analysis of named entity recognition in bccwj. *Recall*, 61:2641.

Binh An Nguyen, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen. 2019. Error analysis for vietnamese named entity recognition on deep neural network models. *arXiv preprint arXiv:1911.07228*.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Anthony J Viera, Joanne M Garrett, et al. 2005. Understanding interobserver agreement: the kappa statistic. *Fam med*, 37(5):360–363.