Nested Noun Phrase Identification using BERT

Shweta Misra and Johan Boye
KTH Royal Institute of Technology
Brinellvägen 8, 114 28 Stockholm, Sweden
shwetam, jboye@kth.se

Abstract

For several NLP tasks, an important substep is the identification of noun phrases in running text. This has typically been done by "chunking" – a way of finding minimal noun phrases by token classification. However, chunking-like methods do not represent the fact that noun phrases can be nested. This paper presents a novel method of finding all noun phrases in a sentence, nested to an arbitrary depth, using the BERT model for token classification. We show that our proposed method achieves very good results for both Swedish and English.

Keywords: noun phrase, chunking, partial parsing, nested phrases, language models, BERT

1. Introduction

For several NLP tasks, an important substep is the identification of phrases, in particular noun phrases (NPs), in running text. For instance, a co-reference resolution system needs to correctly identify the *mentions* of people, places, and concepts that might co-refer (Soon et al. 2001; Zheng et al. 2011), and in so-called "answer-aware" question generation, noun phrases often constitute the desired answer on which to base a question (Ma et al. 2020).

A group of methods for identifying noun phrases (without resorting to full parsing) dates back to Church (1988). These methods rely on token-based tagging, using a "BIO" scheme, where "B" "I" and "O" denote "beginning", "inside" and "outside" of an NP, respectively. For instance, the sentence "Alice and Ben took the bus to York" could be tagged as follows:

Alice	and	Ben	took	the	bus	to	York
B	O	B	O	B	I	O	B

However, this scheme does not represent the fact that noun phrases can be nested. In the example above, "Alice" and "Ben" are separate NPs but also part of the larger NP "Alice and Ben", and similarly "the bus" and "York" are parts of "the bus to York". One might imagine an alternative scheme, instead tagging maximal NPs, i.e. tagging "Alice and Ben´´ as "`BII`", but this would lose the information that "Alice´" and "Ben" also are separate NPs. And this information is important, for instance in co-reference resolution, where "Alice" might co-refer with "her" whereas "Alice and Ben" might co-refer with "them".

This paper presents a novel method of finding *all* noun phrases in a sentence, nested to an arbitrary depth, using the BERT model for token classification (Devlin et al. 2018) in a recursive manner. Our approach thus constitutes a useful middle ground between chunking and full parsing. We train and evaluate the method on the OntoNotes 5.0 corpus (Pradhan et al. 2012). Furthermore, to show that the method generalizes across languages, we train and evaluate a Swedish model using the KB-BERT model (Malmsten et al. 2020) and the Talbanken corpus (UD Swedish Talbanken).

Our method results in syntax tree structure(s), but using simple token classification without any pre-written grammar, and without any requirement to form a single tree from the sentence – we are merely interested in identifying the noun phrases. The basic idea is to iteratively merge consecutive words or phrases in the sentence to create tree structures, while at the same time determining which of these tree structures actually constitute NPs.

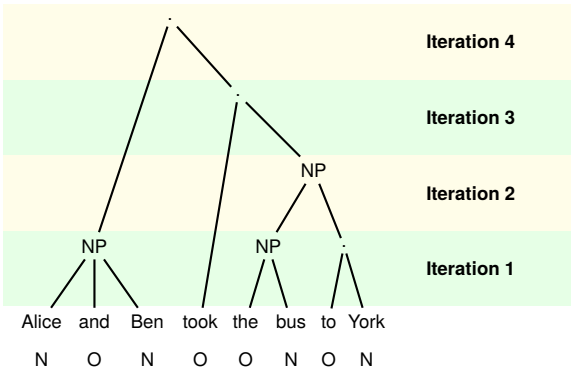

Figure 1: Bottom-up construction of a syntax tree, while simultaneously distinguishing noun phrases (NP), from other phrases (.) Words labeled "N" are nouns, which can be considered atomic NPs, whereas "O" are non-nouns.

For instance, consider the sentence "Alice and Ben took the bus to York" again. Here, our method would first identify "Alice", "Ben", "bus", and "York" as nouns, pronouns or names (which could all be the head word of a noun phrase). In the next iteration, "Alice and Ben" would be merged into a

LREC-COLING 2024, pages 12138–12143

phrase which additionally would be identified as an NP, as would "the bus", whereas "to York" would be merged into a phrase which would be identified as NOT an NP. In the next step, "the bus to York" would be merged and identified as an NP. Then, "took" is merged with "the bus to York, and finally, the whole sentence would be merged into a phrase. In the two final steps, neither of the resulting phrases would be identified as an NP. The whole process is illustrated in Figure 1.

The next section describes how this process is achieved using BERT.

2. Method

The first step of the method is to introduce a new token "|" (the "pipe" symbol), which is put in-between every original token. For instance, "Alice and Ben took the bus to York" (8 tokens) is now expanded into the 15 tokens:

Alice | and | Ben | took | the | bus | to | York

The "|" symbol is a boundary, signifying that the word(s) on either side of it are not part of the same phrase. The idea is now to have the model (i) decide which of these "|"'s should be removed, in order to join the surrounding words into a phrase, (ii) decide which of these newly-formed phrases are NPs, and (iii) decide which of the original words is a noun, name or pronoun (for the sake of simplicity, we will refer to both as "nouns").

In order to achieve this, we fine-tune the BERT model to perform a five-class token classification problem, using the classes

N Noun

O Not a noun

NP Remove token. The resulting phrase is a noun phrase.

R Remove token. The resulting phrase is a NOT a noun phrase.

D Don't remove token

Though technically this is a five-class problem, the two first classes will only be assigned to the original words, where as the three last classes will only be assigned to "|" tokens.

Continuing the example, the correct tagging of the extended sentence would be:

Alice	and	Ben	took	the	bus	to	York
N	NP O	NP N	D	O	D	O	NP N D O R N

This tagging represents that "Alice", "Ben", "Alice and Ben", "bus", "the bus", and "York" are NPs, and that "to York" is a phrase which is not an NP. At training time, this extended sentence and its tagging would be a data point, while at inference time the model should produce this tagging as output.

We then remove all "|" labelled NP or R. In the next iteration, the correct tagging is:

Alice	and	Ben	took	the	bus	to	York
N	O	N D O	D	O	N	NP O	N

We can now conclude that "the bus to York" is an NP. The final two iterations do not yield any further NPs:

Alice	and	Ben	took	the	bus	to	York
N	O	N D O	R	O	N	O	N

Alice	and	Ben	took	the	bus	to	York
N	O	N R O		O	N	O	N

After removing the final "|" symbol tagged "R", the process terminates.

The above example shows the behavior of the method at inference time. BERT is iteratively called with the input strings shown above until no more "|" tokens are left, or the output contains no "R" or "NP" tags. Intuitively, this process corresponds to the bottom-up tree building illustrated in Figure 1.

The process may terminate before a complete tree has been built. This would happen if some "|" tokens are present in the input, but the model predicts the label "D" for all these tokens. In that case, no "|" tokens are removed, so the input for the next iteration would be unchanged. The process then terminates and returns a set of subtrees rather than a complete tree. Unless the top node of the tree was an NP, this would not impact the overall result.

If the above sentence was used as training material, we would obtain 4 data points: one data point for each of the expanded strings above (one for each iteration), with its corresponding tagging.

2.1. Data

We used English part of the OntoNotes 5.0 corpus (Pradhan et al. 2012), together with the OntoNotes-5-parsing program (Bondarenko 2021) to obtain a more easily readable format from the OntoNotes parse trees. The OntoNotes corpus contains 'syntax' and 'morphology' information for each sentence which was used in data pre-processing steps. The pre-processing was broadly divided into the following steps:

1. Firstly, the syntax and morphology labels were mapped from character indices to words in a sentence.

2. Then, different variants of multi-word noun phrases were standardized into a single label 'NP'. This concerned labels like NP, NP-SBJ, NP-PRD etc. in the OntoNotes corpus – all of

these were considered NPs. The labels for the remaining phrases, like 'VP', 'PP' etc., were kept as they were. Single-word nouns labeled 'NP', 'NN', 'NNS', 'NNP', 'NNPS', 'NP', and 'NS' in the OntoNotes corpus were relabeled into 'N', while all other words and special characters appearing in the sentence were labeled 'O'.

3. Lastly, training data points were created by the process described at the beginning of section 2. The boundary tokens "|" separating words and phrases in a sentence were removed by ascending order of phrase length (number of words in a phrase), as determined by the labels "NP" and "R".

We obtained 113,334 sentences from the OntoNotes corpus. 80% of the corpus, 90,667 sentences from which 511,281 data points were obtained, was used for training. The remaining 20%, 22,667 sentences, were used for evaluation.

For the Swedish data, we used Talbanken (UD Swedish Talbanken), from which we extracted 226,008 data points of 63,800 sentences from the conll files. 62,307 of these sentences were used for training and 1,493 for evaluation. The preprocessing steps involved with the Swedish data were similar, but less complicated as the syntax labels were already mapped to words in a sentence rather than character indices. All multi-word noun phrases were called 'NP', whereas single-word nouns, proper names, and pronouns ('NN', 'PM', and 'PN', respectively) were labeled 'N'.

2.2. Models

For the English data, we used the pre-trained BERT model 'bert-base-cased' with 109M parameters (Devlin et al. 2018), available from the python `transformers` library (Bert-HuggingFace) with a softmax classifier added to the base model. We fine-tuned the model using 511,281 training examples created using the data in the OntoNotes corpus with a training batch size of 128 and a learning rate of 1×10^{-5}. Initially, we also trained the 'bert-base-uncased' model which performed only slightly worse than the bert-base-cased model, hence we decided to go ahead with the 'bert-base-cased' model for further analysis. Since most proper nouns are written with a capital first letter the case information helps in identifying NPs.

For the Swedish data, we used the pre-trained KB-BERT model ´bert-base-swedish-cased' (Malmsten et al. 2020) with a softmax classifier. We fine-tuned the model using 217,615 training examples from the Talbanken corpus, using a batch size of 64 and a learning rate of 1×10^{-5}.

To make the results for the Swedish model and English model more comparable, we also trained the English BERT on approximately half of the training data, 218,000 datapoints and calculated evaluation metrics on 1,500 sentence from the test corpus. All the models were trained using NVIDIA T4(x2) GPU provided by Kaggle (Kaggle GP T4).

2.3. Evaluation metrics

To evaluate the results, we compute precision, recall, and F1-score in terms of the number of identified NPs at the *sentence* level and the *sub-tree* level, as well as the number of completely correct trees.

At the *sequence* level, we compare predicted and ground truth NPs, irrespective of the syntax tree structure. This is the most lenient measure of performance in our evaluation. By contrast, at *sub-tree* level, we consider a NP as correctly identified if all the NPs below it in the tree have also been correctly identified. As example highlighting the difference between the two, consider the predicted tree:

(NP (NP a window with) (NP a wooden frame))

where the NP "a window with a wooden frame" would be considered correct on the sequence level but not on the sub-tree level, as the interior NP "a window with" is incorrect.

3. Results and discussion

The evaluation data set from the OntoNotes corpus consisted of 22,667 distinct sentences that were not used during training. Table 1 summarizes the average values of the metrics. We obtained very high recall values (> 0.9) indicating that our model produces a very low number of false negatives, i.e., it rarely misses out on identifying NPs in a sentence. The precision values are also equally high at the sequence and sub-tree level, indicating that our model produces a low number of false positives as well. The tree-level precision and recall are low, as it is enough for a single sub-tree to be identified incorrectly in a sentence for the tree level metrics to become 0 for that sentence.

Figure 2 shows the average precision and recall over all sentences in the test set as a function of the maximal NP depth of the sentence and the sentence length (number of space-separated tokens in a sentence). At the sequence and sub-tree levels, the trends show a slight linear decrease along the sentence length and the maximal depth of NPs, but remain in the range of 1 to 0.8. This indicates that as the length of the sentence or the depth of NPs in the sentence increases, the model performance

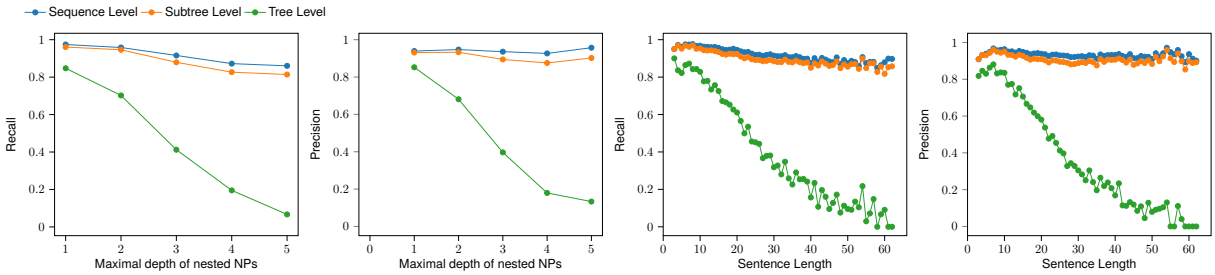

Figure 2: Recall and Precision as a function of the maximum nested NP depth and sentence length (number of space separated tokens). Metrics calculated for the fine-tuned English base-cased BERT model.

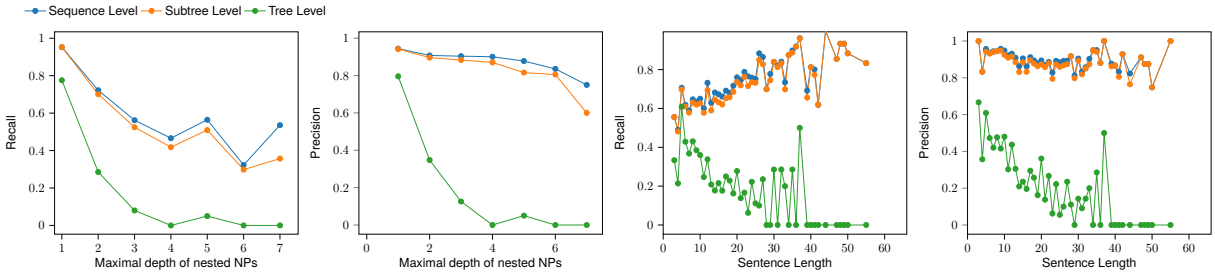

Figure 3: Recall and Precision as a function of the maximum nested NP depth and sentence length (number of space separated tokens), calculated for the fine-tuned Swedish BERT model.

moderately degrades. The tree-level metrics display an exponential drop with respect to both the NP depth and the sentence length. This shows that our model requires further improvement to create completely correct syntax tree structure especially for longer and more complex sentences.

Metric Level	Recall	Precision	F1 score
Sequence	0.94	0.94	0.94
Sub-tree	0.92	0.91	0.91
Tree	0.59	0.57	0.56

Table 1: Average Recall, Precision and F1 at the *sentence* level, the *sub-tree* level and the *tree* level using the fine-tuned English base-cased BERT model.

Metric Level	Recall	Precision	F1 score
Sequence	0.68	0.91	0.74
Sub-tree	0.66	0.89	0.71
Tree	0.27	0.31	0.25

Table 2: Average Recall, Precision and F1-score at the *sentence* level, the *sub-tree* level and the *tree* level on the fine-tuned Swedish KB-BERT model.

We evaluated the fine-tuned Swedish KB-BERT model on 1,493 sentences that had not been seen by the model during training. Table 2 summarizes the overall metric values, while figure 3 plots them along the maximal NP depth and sentence lengths. Similar trends as for the English model are observed. High precision values are achieved at the sequence and the sub-tree level. Again, there is a slight linear decrease with increasing sentence length and depth of NP nested. Recall values however display a more interesting trend. With increasing NP depth there is a steeper decrease in performance indicating that the model creates a greater number of false negatives for more complex sentences. The recall trend across sentence length seems to show a slight linear increase. However, as the sentence length increases the number of NPs in the sentence also increases. Thus, the recall values across sentence length are majorly consistent. The plots along the sentence length seem spiky towards the end as the number of examples for longer sentences was smaller in the Talbanken test corpus.

Metric Level	Recall	Precision	F1 score
Sequence	0.92	0.88	0.90
Sub-tree	0.88	0.83	0.85
Tree	0.44	0.39	0.37

Table 3: Average Recall, Precision and F1-score at the *sentence* level, the *sub-tree* level and the *tree* level on the fine-tuned English base-cased BERT model on half of the training data.

For comparison, the evaluation results for the English model trained with only half of the total English training data available is presented in table 3 .The recall for the Swedish model is lower as compared to the English BERT. A possible explanation for this is that the English training data consists of 732,626 NPs including one-word nouns and all nested NPs while the Swedish training data contains only 416,341 NPs. Despite this difference, high precision values suggest that our method of recursively applying the BERT model to identify NPs nested to an arbitrary depth can be generalized across languages.

4. Related work

Church (1988) was the first to study the chunking problem (i.e. retrieving non-recursive phrases without the use of a pre-written grammar or finite-state automaton), using a statistical technique. The problem has later been attacked using memory-based learning (Daelemans et al. 1999), support vector machines (Kudo and Matsumoto 2001), conditional random fields (Sha and Pereira 2003), hidden Markov models with majority voting (Shen and Sarkar 2005), convolutional neural networks (Collobert et al. 2011), and various kinds of recurrent neural networks like LSTMs and GRUs (Huang et al 2016; Yang et al. 2016; Zhai et al. 2017).

The problem of retrieving recursive chunks has not attracted the same attention. However, Socher et al. presented a recursive method for constructing binary syntax trees, in which a vector of real numbers represents words and phrases (Socher et al. 2011). At every step in the recursion, segments (words/phrases) are merged together, thus creating a parse tree in a bottom-up fashion. An RNN was used to determine, in each iteration, which two segments should be merged, and what the vector representation of the resulting segment would be. This vector representation contains 'syntactic and compositional-semantic information' which could be used for classification of the words/phrases into NP, VP, PP etc. The process terminates when a tree covering the whole sentence has been constructed.

On the other side of the spectrum, there are the statistical or machine-learning-based full parsing approaches. One of the first representatives of this line of research was Collins' statistical parser (Collins 1997). More lately, several researchers have constructed full constituency parsers using neural approaches, either transition-based (Cross and Huang 2016) or graph-based (Stern et al. 2017). There are also many neural dependency parsers, but we leave them aside since we are interested in consituency-based syntax.

Finally, we mention the approaches that have used BERT in some way in the parsing process. In particular, BERT has been used as a feature extractor, to obtain features that can be used to predict stack operations in chart-based neural parsers (Fried et al 2019; Cui et al 2021). We are not familiar with any work that has used BERT in the way we are doing in this paper.

5. Conclusion

With this paper we present a novel method to identify nested noun phrases to an arbitrary depth in a sentence, using a fine-tuned BERT model in a recursive fashion. The method also produces syntactical tree structures for a given sentence in the process. We have achieved high precision levels for both English and Swedish, especially at the sequence and sub-tree levels. Future work involves the simultaneous prediction of several kinds of phrases (like noun phrases, verb phrases, and prepositional phrases) and investigating the approach for more languages than English and Swedish.

The code developed for this research project is available through this link.

6. Bibliographical References

K. Church (1988) A stochastic parts program and noun phrase parser for unrestricted texts. In *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, Texas.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. arXiv preprint cmp-lg/9706022.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12, pp. 2493-2537.

Cross, J. and Huang, L. (2016) Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of EMNLP*, pages 1–11.

Cui, L., Yang, S., & Zhang, Y. (2021). Investigating non-local features for neural constituency parsing. *arXiv preprint arXiv:2109.12814*.

Daelemans, W., Buchholz, S., Veenstra, J. (1999). Memory-based shallow parsing. *arXiv preprint cs/9906005*.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fried, D., Kitaev, N., & Klein, D. (2019). Cross-domain generalization of neural constituency parsers. *arXiv preprint arXiv:1907.04347*.

Huang, Z., Xu, W., Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Kudo, T. and Matsumoto, Y. (2001). Chunking with Support Vector Machines. In *Proc. of NAACL*, pages 192–199

Ma, X., Zhu, Q., Zhou, Y., & Li, X. (2020). Improving question generation with sentence-level semantic matching and answer position inferring. In *Proceedings of the AAAI conference on artificial intelligence*, (Vol. 34, No. 05, pp. 8464-8471).

Malmsten, M., Börjeson, L., & Haffenden, C. (2020). Playing with words at the National Library of Sweden–Making a Swedish BERT. *arXiv preprint arXiv:2007.01658*.

Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., & Zhang, Y. (2012). CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint conference on EMNLP and CoNLL-shared task* (pp. 1-40).

Sha, F. & Pereira, F. (2003). Shallow parsing with conditional random fields. In *Proceedings of the 2003 human language technology conference of the North American Chapter of the Association for Computational Linguistics*, (pp. 213-220).

Shen, H., Sarkar, A. (2005). Voting between multiple data representations for text chunking. In *Advances in Artificial Intelligence: 18th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2005, Victoria, Canada, May 9-11, 2005. Proceedings 18*, (pp. 389-400).

Socher, R., Lin, C. C., Manning, C., Ng, A. Y. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, (pp. 129-136).

Soon, W. M, Ng, H. T., and Lim, D. C. Y. (2001) A machine learning approach to coreference resolution of noun phrases. *Computational linguistics* 27.4 (2001): 521-544.

Stern, M., Andreas, J. and Klein, D (2017). A minimal span-based neural constituency parser. In *Proceedings of ACL*, pages 818–827.

Yang, Z., Salakhutdinov, R., Cohen, W. (2016). Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Zhai, F., Potdar, S., Xiang, B., Zhou, B. (2017, February). Neural models for sequence chunking. In *Proceedings of the AAAI conference on artificial intelligence*, (Vol. 31, No. 1).

Zheng, J., Chapman, W. W., Crowley, R. S., Savova, G. K. (2011). Coreference resolution: A review of general methodologies and applications in the clinical domain. *Journal of biomedical informatics*, 44(6), 1113-1122.

7. Language Resource References

bert-base-cased. `https://huggingface.co/bert-base-cased`

Bondarenko, I. (2021) ontonotes-5-parsing 0.0.5. `https://pypi.org/project/ontonotes-5-parsing` Released June 9, 2021.

UD Swedish Talbanken `https:// universaldependencies.org/treebanks/sv_talbanken`

T4s on Kaggle https://www.kaggle.com/discussions/product-feedback/361104