Investigating NP-Chunking with Universal Dependencies for English

Ophélie Lacroix
Siteimprove
Sankt Annæ Plads 28
DK-1250 Copenhagen, Denmark
ola@siteimprove.com

Abstract

Chunking is a pre-processing task generally dedicated to improving constituency parsing. In this paper, we want to show that universal dependency (UD) parsing can also leverage the information provided by the task of chunking even though annotated chunks are not provided with universal dependency trees. In particular, we introduce the possibility of deducing noun-phrase (NP) chunks from universal dependencies, focusing on English as a first example. We then demonstrate how the task of NP-chunking can benefit PoS-tagging in a multi-task learning setting – comparing two different strategies – and how it can be used as a feature for dependency parsing in order to learn enriched models.

1 Introduction

Syntactic chunking consists of identifying groups of (consecutive) words in a sentence that constitute phrases (e.g. noun-phrases, verb-phrases). It can be seen as a shallow parsing task between PoS-tagging and syntactic parsing. Chunking is known as being a relevant preprocessing step for syntactic parsing.

Chunking got a lot of attention when syntactic parsing was predominantly driven by constituency parsing and was highlighted, in particular, through the CoNLL-2000 Shared Task (Tjong Kim Sang and Buchholz, 2000). Nowadays, studies (Søgaard and Goldberg, 2016; Hashimoto et al., 2017) still compare chunking –as well as constituency parsing– performance on these same data from the Penn Treebank. While dependency parsing is spreading to different languages and domains (Kong et al., 2014; Nivre et al., 2017), chunking is restricted to old journalistic data. Nevertheless, chunking can benefit dependency parsing as well as constituency parsing, but gold annotated chunks are not available for universal dependencies.

We want to automatically deduce chunks from universal dependencies (UD) (Nivre et al., 2017) and investigate its benefit for other tasks such as Part-of-Speech (PoS) tagging and dependency parsing. We focus on English, which has properties that make it a good candidate for chunking (low percentage of non-projective dependencies). As a first target, we also decide to restrict the task to the most common chunks: noun-phrases (NP).

We choose to see NP-chunking as a sequence labeling task where tags signal the beginning (B-NP), the inside (I-NP) or the outside (O) of chunks. We thus propose to use multi-task learning for training chunking along with PoS-tagging and feature-tagging to show that the tasks can benefit from each other. We experiment with two different multi-task learning strategies (training parameters in parallel or sequentially). We also intend to make parsing benefit from NP-chunking as a pre-processing task. Accordingly, we propose to add NP-chunk tags as features for dependency parsing.

Contributions. We show how to (i) deduce NP-chunks from universal dependencies for English in order to (ii) demonstrate the benefit of performing chunking along with PoS-tagging through multi-task learning and (iii) evaluate the impact of using NP-chunks as features for dependency parsing.

2 NP-Chunks

While chunks are inherently deduced from constituent trees, we want to deduce chunks from dependency trees in order to not rely on specific constituent annotations which would not be available for other domains or languages. In this case, it means that only partial information is provided by the dependencies to automatically extract the chunks. We thus choose to only deduce noun-phrase (NP) chunks (Ramshaw and Marcus, 1995) from the dependency trees.

Proceedings of the Second Workshop on Universal Dependencies (UDW 2018), pages 85–90
Brussels, Belgium, November 1, 2018.

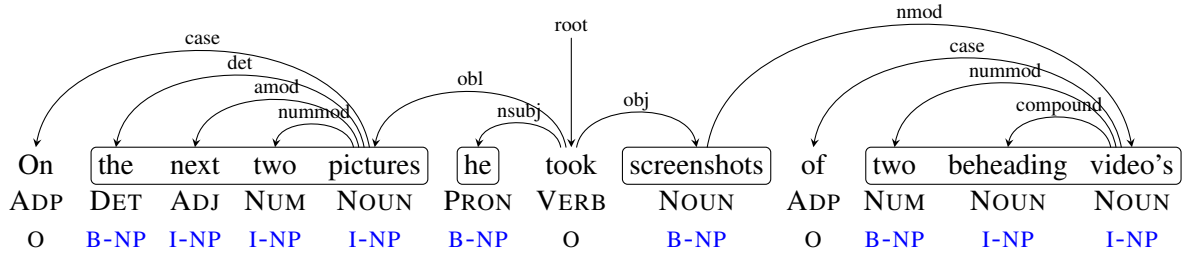

Figure 1: NP-chunks deduced from a UD tree of the English Web Treebank (EWT).

Automatic Deduction. We deduce minimal NP-chunks, which means that embedded prepositional (PP) chunks are not included in our NP-chunks, e.g. in Figure 1 "*screenshots of two beheading video's*" is split in two distinct NPs instead of one long NP with an embedded PP ("*of two beheading video's*").

We first identify the core tokens of NPs: the nouns (NOUN), proper nouns (PROPN) and some pronouns[1] (PRON). After identifying these core tokens, we form full NPs by joining these core tokens with their direct and indirect children which are not part of PPs. In practice, they are those for which the incoming dependency is labeled with one of the following relations (modulo some individual conditions):

- `compound`, `compound:prt`, `flat`, `flat:name`, `goeswith`, `fixed`, `nummod`;
- `det` if the child is located before its head;
- `conj` if the child and its head are adjectives. We want "*excellent and strong performers*" to be one NP and "*these challenges and possible solutions*" to be split in two NPs;
- `amod` if the child is not an adverb. We don't want to attach preceding adverbs such as "*not*" to a NP;
- `appos` if the child is directly before or after its head;
- `advmod` if the child is not a PART or a VERB and its head an adjective;
- `nmod:poss` if the child is not a NOUN or a PROPN. We want to group "*your world*" but not "*John's last day* (where "*John*" and "*last day*" would be two distinct NPs;
- following and preceding `obl:npmod` and `obl:tmod`;
- `obl` if its head has a `amod` incoming dependency.

In addition, when grouping a core token with one of its `det`, `compound`, `nummod` or `nmod:poss` children, we automatically attach tokens which are in between. If split chunks remain, we attach the non-attached tokens which are in between two part of a chunk. It allows us to attach the adverbs which modify adjectives such as "*very*" in "*my very best friend*" or some specific punctuation such as the slash in "*The owner/baker*".

Manual Annotation. To assert the correctness of the automatically deduced chunks, we manually annotate noun-phrases on a small portion of the test set of the EWT UD treebank data. For 50 sentences (from which 233 NP chunks were manually annotated), the accuracy of the automatic deduction reaches 98.7%. Errors in deduction are mostly due to punctual inconsistencies in the UD annotations.

3 Models

3.1 Sequence Labeling

We implement a deep recurrent neural network with an architecture based on bidirectional Long Short-Term Memory (bi-LSTM) (Graves and Schmidhuber, 2005) that can exploit contextual information for processing sequences.

The base network is composed of an embedding layer that feeds two hidden bi-LSTM layers (forward and backward). The outputs of the bi-LSTMs are then concatenated to feed the next layer. Multiple bi-LSTM layers can be stacked. In the end, these outputs are fed to a Softmax output layer. The embedding layer is a concatenation of a word embedding layer and a character embedding

[1] All pronouns but the interrogative and relative pronouns.

layer. It takes as input a sequence of n tokens. The output of the network is a sequence of n tags.

We use this architecture for PoS-tagging, feature-tagging (i.e. morpho-syntactic tagging) and NP-chunking. In order to make the tasks benefit from each other, we adapt the network to multi-task learning. We propose to compare two strategies for multi-task learning : *shared* or *stacked*.

Shared multi-task learning. In this architecture, different tasks are trained at the same level in a similar way as in Søgaard and Goldberg (2016). They share parameters through all the network and feed different outputs.

Stacked multi-task learning. In this architecture, different tasks are trained at different levels as proposed by Hashimoto et al. (2017). A bi-LSTM layer is dedicated to a task. The output of a layer for a given task feeds the next layer dedicated to the next task.

3.2 Dependency Parsing

Our dependency parser is a reimplementation of the arc-hybrid non-projective transition-based parser of de Lhoneux et al. (2017b).

In this version of the arc-hybrid system, the SWAP transition is added to the original transition set (Kuhlmann et al., 2011) made up of the standard transitions RIGHT, LEFT and SHIFT. The SWAP transition allows to build non-projective dependency trees. The standard transitions are trained using a dynamic oracle (Goldberg and Nivre, 2013), which alleviates error propagation, and a static oracle for training the SWAP transition.

The parser uses a bi-LSTM network to learn vector representations of the tokens. These vectors are combined through a feature function and used for learning and evaluating the transitions using a multi-layer perceptron with one hidden layer. In de Lhoneux et al. (2017a), PoS tags are removed from the feature function and instead, the bi-LSTM is fed with only word and character embeddings. In our version of the parser, we reintroduce the PoS tags as features and also make use of the predicted NP-chunks. The PoS and/or NP-chunk tags are turned into embeddings and concatenated with the word and character embeddings to represent the tokens.

4 Experiments

As a baseline for PoS-tagging, feature-tagging and NP-chunking, we first train our sequence tagger for each task separately. We then train the tagger in a multi-task setting –with PoS-tagging as a main task– alternating the auxiliary tasks and the strategies (shared or stacked multi-task learning).

As a baseline for dependency parsing, we train the parser using only word and character embeddings as input to the bi-LSTM. We then add the PoS and NP-chunk embeddings, separately and simultaneously, for training enriched models. As an upper bound, we also propose to run the experiments with "gold" NP-chunks, i.e. we feed the parser (for training and testing) with NP-chunks that were automatically deduced from the dependencies.

Data. We evaluate all tasks on the three English treebanks included in the version 2.1 of the Universal Dependencies project (Nivre et al., 2017) : **EWT** (254k tokens), **LinES** (82k tokens) and **ParTUT** (49k tokens). In average, 3.8, 3.3 and 6.2 NP-chunks per sentence are deduced respectively for each treebank. [2] Note that the LinES treebank does not contain features (morpho-syntactic tags), so we exclude feature-tagging from the evaluation for this treebank.

Hyper-parameters. We use the development data to tune our hyper-parameters and to determine the number of epochs (via early-stopping) for each experiment.

For sequence tagging, we use the RMSProp optimizer with a learning rate at 0.0005. Hidden layers of dimension 300 is used for ParTUT and 100 for EWT and LinES. We use a dropout of 0.2 on the hidden layers. For dependency parsing, the hidden layer of the bi-LSTM has a dimension set at 125 and uses a dropout of 0.33.

The dimension of the word and character embeddings are respectively 200 and 50. For dependency parsing, embedding dimensions for PoS and NP-chunk tags are set respectively to 6 and 3.

Evaluation. We average the scores on 5 runs for each experiment. We evaluate accuracy on PoS-tagging and feature-tagging and F_1 [3] on chunking.

[2] EWT is the biggest treebank but the test contains small sentences (12.1 average length) while ParTUT is the smallest treebank but contains long sentences (22.3 average length).

[3] $F_1 = 2 * precision * recall / (precision + recall)$ where precision is the percentage of predicted chunks that are cor-

	EWT			LinES		ParTUT		
	PoS acc(%)	**Feats** acc(%)	**Chunks** F_1	**PoS** acc(%)	**Chunks** F_1	**PoS** acc(%)	**Feats** acc(%)	**Chunks** F_1
Baseline	93.16	**94.06**	89.32	**93.00**	82.74	92.61	91.03	88.01
Shared - P+F	93.29	93.97	-	-	-	93.04	91.49	-
Shared - P+C	93.11	-	89.98†	92.97	**85.63**†	93.19†	-	89.20†
Shared - P+F+C	**93.30**	94.01	**89.99**†	-	-	**93.20**†	**91.74**†	89.26†
Stacked - P+F	93.18	93.92	-	-	-	92.67	91.00	-
Stacked - P+C	93.16	-	89.09	92.82	83.14	92.96	-	88.28
Stacked - P+F+C	93.00	93.75	89.08	-	-	93.13	91.25	**89.74**†

Table 1: Results of PoS-tagging (**P**), feature-tagging (**F**) and NP-chunking (**C**) trained as **one task** (baseline) or via multi-task learning (**Shared** vs **Stacked** strategies). Bold scores are the highest of each column. Statistical significance (T-test>0.05) is marked with †.

	EWT			LinES			ParTUT		
	LA	UAS	LAS	LA	UAS	LAS	LA	UAS	LAS
Baseline	**87.83**	**86.26**	**82.27**	82.54	82.06	75.35	87.28	86.00	81.28
+ P	87.01	85.58	81.20	**83.71**†	**83.10**†	**76.83**†	87.63	86.11	81.51
+ C	87.66	86.19	81.86	82.66	82.53	75.57	87.98	86.53	82.17
+ P+C	87.32	85.98	81.59	83.38†	82.87†	76.37†	**88.05**	**86.88**†	**82.24**†
+ gold C	89.99	89.07	85.45	84.31	84.05	77.94	89.47	87.92	84.13

Table 2: Results of dependency parsing using PoS (**P**) and/or NP-chunk (**C**) features.The baseline uses only word and character embeddings. Highest scores are in bold. † indicates statistical significance.

For dependency parsing, we calculate the label accuracy (LA), the unlabeled attachment score (UAS) and the labeled attachment score (LAS). As for the CoNLL 2017 Shared Task (Hajič and Zeman, 2017), only universal dependency labels are taken into account (ignoring language-specific subtypes), i.e. we consider a predicted label correct if the main type of the gold label is the same, e.g. `flat:name` is correct if the gold label is `flat`. We also exclude punctuations from the evaluation.

5 Results

5.1 Tagging results

See PoS-tagging, feature-tagging and NP-chunking results in Table 1. For all three treebanks, multi-task learning is beneficial for at least one task. Only the LinES treebank does not benefit from it for PoS-tagging (i.e. equivalent performance), however it greatly improves NP-chunking (+2.9). For the smallest treebank (ParTUT), multi-task learning is beneficial for all tasks (at best, +0.6 for PoS-tagging, +0.7 for feature-tagging and +1.73 for NP-chunking). For the EWT treebank, equivalent scores are achieved for feature-tagging but PoS-tagging and NP-chunking are enhanced through multi-task learning (respectively +0.14 and 0.67).

Globally, the shared multi-task learning strategy achieves the best results. The stacked strategy outperforms the baseline for the small treebank but gets lower scores on the big treebank.

It is also worth noting that multi-task learning makes the models more stable. We observe a significant decrease of the standard deviation for most of the experiments.[4]

5.2 Dependency Parsing Results

See dependency parsing results in Table 2. Adding PoS and NP-chunk tags as features significantly improve dependency parsing performance for the smallest treebank, ParTUT (+0.96 LAS). Using

rect and recall is the percentage of gold chunks that are correctly predicted.

[4] 10 out of 12 standard deviations are lower when comparing the baseline to the shared multi-task learning (including chunking as an auxiliary task).

NP-chunks alone is also beneficial on the LinES data (+0.22 LAS over the baseline) but using only PoS-tags is actually more relevant than including both features. For the biggest treebank, EWT, the baseline outperforms all other enriched models. However, the upper-bound shows that the NP-chunk tags as features are relevant for improving dependency parsing, suggesting that the quality of the predicted NP-chunks –as well as the PoS-tags– is not sufficient for improving parsing.

It is worth noting that training converges faster when using features (17.6 epochs on average VS 25.8 for the baseline) which might also indicate a training issue since models that stop after few epochs (11/12) achieve lower performance.

6 Conclusion

We showed that it is possible to extract NP-chunks from universal dependencies that can be useful for improving other tasks such as PoS-tagging and dependency parsing. While the improvement for PoS-tagging is systematic on all English UD treebanks, the results are mixed for dependency parsing suggesting that NP-chunks as features might be useful for training on small datasets.

Further experiments will be performed in future work in order to extend the results to other languages and to investigate the possibility of extracting embedded chunks.

Acknowledgment

The author would like to thank the anonymous reviewers for their comments and suggestions, and add as well a special thanks to her colleagues from the Data Science team at Siteimprove.

References

Yoav Goldberg and Joakim Nivre. 2013. Training Deterministic Parsers with Non-Deterministic Oracles. *Transactions of the Association of Computational Linguistics*, 1:403–414.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, pages 5–6.

Jan Hajič and Dan Zeman, editors. 2017. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A Smith. 2014. A Dependency Parser for Tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic Programming Algorithms for Transition-Based Dependency Parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017a. From Raw Text to Universal Dependencies-Look, No Tags!

Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017b. Arc-Hybrid Non-Projective Dependency Parsing with a Static-Dynamic Oracle. In *Proceedings of the 15th International Conference on Parsing Technologies (IWPT 2017)*.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, John Lee, Phương

Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Robert Östling, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jonathan North Washington, Mats Wirén, Tak-sum Wong, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Lance Ramshaw and Mitch Marcus. 1995. Text Chunking using Transformation-Based Learning. In *Third Workshop on Very Large Corpora*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 2nd Workshop on Learning language in logic and the 4th conference on Computational Natural Language Learning*.