PALMYRA 3.0: A User-Friendly Cloud-Based Platform for Morphology and Dependency Syntax Annotation

Muhammed AbuOdeh,[1] **Long Phan,**[1] **Ahmed Elshabrawy,**[1,2] **and Nizar Habash**[1]
[1]Computational Approaches to Modeling Language (CAMeL) Lab, New York University Abu Dhabi
[2]Mohamed bin Zayed University of Artificial Intelligence
{m.abuodeh,lvp243,nizar.habash}@nyu.edu, ahmed.elshabrawy@mbzuai.ac.ae

Abstract

We present **PALMYRA 3.0**, a cloud-based, configurable, and user-friendly platform for morphology and syntax annotation through dependency-tree visualization. **PALMYRA 3.0** implements a robust system that stores data on the cloud. By default, **PALMYRA 3.0** comes with an Arabic dependency parser that generates highly accurate trees, but it is easily configurable to support dependency parsers in other languages. **PALMYRA 3.0** provides default configuration files for a number of predefined formalisms, such as UD and CATiB, and a number of user-friendly features to support annotators.

Keywords: Annotation, Morphology, Syntax, Parsing, Treebanking, Universal Dependencies, Arabic

1. Introduction

Treebank development is an important step in many research areas in natural language processing (NLP). Treebanks have been used to study linguistic phenomena and train parsing models (Jurafsky and Martin, 2009). In the neural age, the interest in using syntactic structures remains: they can be used as support tools for (a) studying large language models (Kulmizev, 2023), (b) guiding data augmentation for neural machine translation (Duan et al., 2023), (c) semantic role labeling (Tian et al., 2022), and grammatical error correction (Li et al., 2022; Zhang et al., 2022), etc.

High-quality treebanking is resource intensive. Well-known treebanks such as the Prague Dependency Treebank (Böhmová et al., 2003), the Penn Treebank (Marcus et al., 1993), and the Penn Arabic Treebank (Maamouri et al., 2004) are produced as a result of decade-long efforts by teams of experts and all involved the development of tools to support the annotation processes.

Of course, treebanks vary in terms of the syntactic formalism they use (e.g., the various types of dependency and constituency representations), the level of richness of their syntactic labels and morphological tags, not to mention language-specific tokenization decisions, among other details that should be carefully documented in predefined guidelines. The more complex the formalisms and guidelines, the more difficult it is to train annotators, and to annotate the data consistently, accurately, and efficiently. Hence, there is a great need for tools that facilitate the annotators' work to create quality data, while supporting different formalisms.

While many syntactic and morphological annotation tools exist (Pajas, 2008;Heinecke, 2019; Guibon et al., 2020; Little and Tratz, 2016; Klie et al., 2018), many are lacking in different functionalities that allow for faster annotation, configurability, ease of use, error prevention, and convenience. Many tools are designed with English or other European languages in mind and are cumbersome to use with morphologically rich and complex languages such as Arabic.

In this demo paper, we present **PALMYRA 3.0**, an open-source web-based tool for syntactic and morphological annotation of texts.[1,2] **PALMYRA 3.0** is natural-language agnostic, configurable, and cloud-based. It also provides offline capabilities. **PALMYRA 3.0** was designed to easily annotate morphologically rich languages, as it was originally created for Arabic text annotation (Habash et al., 2022). **PALMYRA 3.0** helps annotate efficiently by providing keyboard shortcuts to traverse and annotate trees, and also helps reduce erroneous annotations by providing undo/redo functionality.

In Section 2, we present a survey and comparison with some existing treebank annotation tools. Section 3 discusses the design decisions and methodology behind developing **PALMYRA 3.0**. In Section 4, we evaluate our tool in terms of accuracy and speed on two Arabic treebanking formalisms as a case study.

2. Related Work

Since we focus on aspects that provide online capabilities, morphosyntactic annotation features, and custom configurations and shortcuts, in this section we will show examples of some existing tools and point out limitations with respect to our requirements. While these tools may have other functionalities that **PALMYRA 3.0** does not have, our focus on our design decisions is intentional; **PALMYRA 3.0** is meant to be lightweight and easy to use. Table 1 provides a summary of the tools we compare.

[1]https://palmyra.camel-lab.com/
[2]https://github.com/CAMeL-Lab/palmyra

LREC-COLING 2024, pages 12585–12591
20-25 May, 2024.

	TrEd	ConlluEditor	ArboratorGrew	Palmyra2.0	Palmyra3.0
Online interface			✓	✓	✓
Online storage			✓		✓
Online parsing			✓		✓
Word tokenization		✓		✓	✓
Configurable formalisms	✓	✓		✓	✓
Keyboard shortcuts	✓	✓		✓	✓
Undo/Redo	✓	✓	✓		✓

Table 1: We compare **PALMYRA 3.0** to four other systems across seven features.

Perl-based **TrEd** (Pajas, 2008) is not web-based and must be downloaded in order to be used. While **TrEd** has many functionalities, it sometimes is unintuitive which makes it difficult for less experienced annotators to use, and does not have a simple option for word-level tokenization.

ConlluEditor (Heinecke, 2019) is similar to **PALMYRA 3.0** in many ways, except that it is not web-based.

ArboratorGrew (Guibon et al., 2020) is an online annotation platform. It provides accounts for its users, and users can store their data in private or public repositories. Users can train dependency parsing models and use pre-trained models, and users may also syntactically annotate and add features to trees. It is limited to the UD formalism, and annotating trees can be slow since users cannot use keyboard shortcuts, nor collapse parts of trees for longer sentences in order to focus on specific subtrees.

PALMYRA 2.0 (Taji and Habash, 2020) is a platform independent graphical dependency tree visualization and editing software. **PALMYRA 2.0** also allows users to either use predefined configuration files or to create their own in order to add keyboard shortcuts. For example, if a user wishes to tag a token as `prep` they would move to the token using the arrow keys then press the letter `p`. If multiple tags require the shortcut `p`, e.g. `part`, `pron`, etc., then the user would press the letter `p` multiple times to rotate through the different tags. Users can also edit the morphological features (e.g., case, state, gender, ...) of tokens. **PALMYRA 2.0** added these features to the previous version, **PALMYRA 1.0** (Javed et al., 2018), which extended EasyTree (Little and Tratz, 2016). **PALMYRA 2.0**, however, lacks online features which forces users to download their trees regularly to prevent loss of work, and the lack of an undo/redo features may introduce annotation errors.

PALMYRA 3.0 was inspired by the tools above, and was built on **PALMYRA 2.0**; **PALMYRA 3.0** contains an online interface, as well as morphosyntactic annotation, word tokenization, and configurable formalisms. As shown in Table 1, **PALMYRA 3.0** provides additional web-based functionality, both default and customizable configurations, undo/redo features and keyboard shortcuts.

There are other tools primarily used for syntactic annotation which we did not cover here, some of which can be found on the Universal Dependencies website.[3] Other, more generic tools such as **INCEpTION** (Klie et al., 2018) are not exclusively focused on syntactic annotation. While **INCEpTION** is web-based and has many features such as configurability and undo/redo functionality, the system design does not help improve the treebank annotator efficiency.

3. Design Decisions & Methodology

In this section, we describe the features we added to **PALMYRA 2.0** as part of developing **PALMYRA 3.0** to enhance annotator efficiency. We discuss online capabilities (persistence and dependency parsing), configurability, and efficient error prevention (through the undo/redo feature).

3.1. Cloud-based Accounts

With some online tools including **PALMYRA 2.0**, once the user closes or reloads the browser, all progress is lost. This is because the progress is stored as Document Object Model (DOM) objects: once the browser is reloaded or closed, the DOM objects are unmounted and not stored anywhere. DOM objects could be forcibly stored as strings in the browser's local storage; however, since local storage has a limited memory and data can only be stored as strings, this is far from an acceptable solution. The only way to make the user's progress persistent is to store it on a server; and in order to distinguish between different users' progress, an account system must be in place.

PALMYRA 3.0 aims to be both lightweight and highly secure at the same time, which is why we integrated access to Google Drive into the

[3] https://universaldependencies.org/tools.html

[3] HSB Arabic transliteration (Habash et al., 2007)

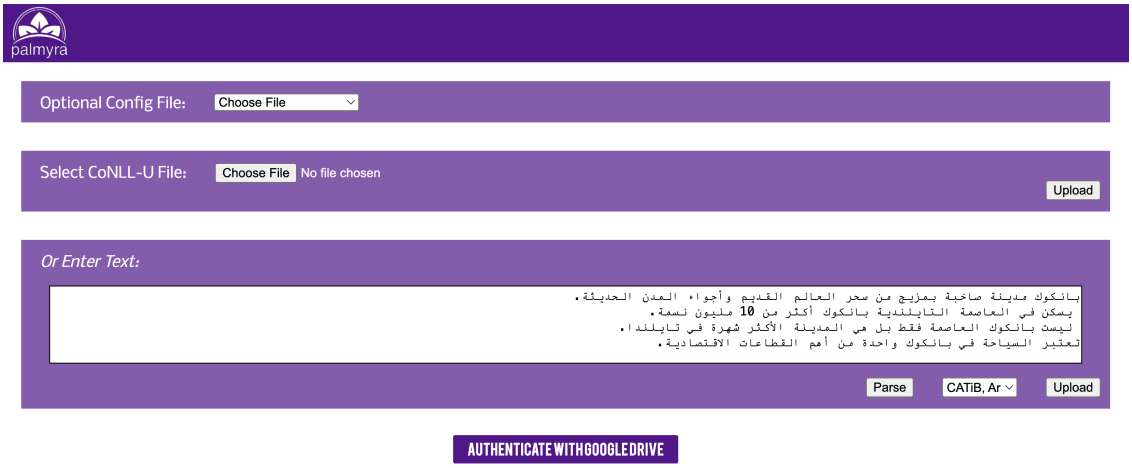

Figure 1: The upload page of **Palmyra 3.0**. In addition to uploading a CoNLL-U/X file, users may also paste sentences in the 'Enter Text' section, and run the built-in parser by clicking on the 'Parse' button. Users may also click on the 'Authenticate with Google Drive' button to work on a file stored on the cloud.

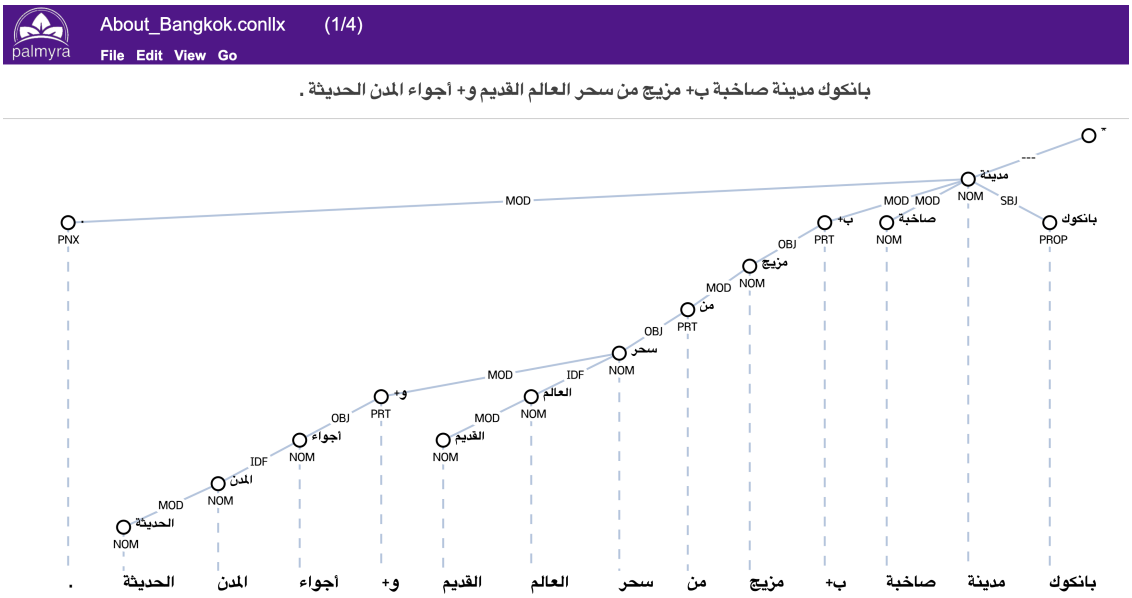

Figure 2: An example of a tree on **Palmyra 3.0**. The waterfall structure makes longer trees clearer to read, and the configurability allows for formalisms besides UD; in this example, CATiB is used. The automatically parsed example sentence is بانكوك مدينة صاخبة بمزيج من سحر العالم القديم وأجواء المدن الحديثة. *bAnkwk mdynħ SAxbħ bmzyj mn sHr AlςAlm Alqdym wÂjwA' Almdn AlHdyθħ*. 'Bangkok is a bustling city with a mix of old-world charm and modern-city vibe.'

Palmyra 3.0 platform. Google APIs allow users to log in with their Google account, browse, and store files on their own Drive.

3.2. Dependency Parser

By default, **Palmyra 3.0** comes with CamelParser2.0 (Elshabrawy et al., 2023), a state-of-the-art Arabic dependency parser, that allows users to annotate by starting from highly accurate trees in Universal Dependency (UD) (Nivre, 2014; Taji et al., 2017) and Columbia Arabic Treebank (CATiB) (Habash and Roth, 2009) formalisms. CamelParser2.0 uses a BERT embedding layer with a neural dependency parser with biaffine attention (Dozat and Manning, 2016; Devlin et al., 2018; Zhang, 2021), and is trained on the Penn Arabic Treebank (PATB) (Maamouri et al., 2004) and the CamelTreebank (Habash et al., 2022). The parsing pipeline also includes a disambiguation step to predict part-of-speech tags and ATB tokenization (Inoue et al., 2022). Figure 1 shows the upload screen where a sample of sentences was entered,

Dev Set Statistics	Predicted Tokenization		Gold Tokenization	
# Sentences	1,986			
# Words	63,042			
# Tokens	74,609		73,945	
Tokens/Sentence	37.6 (σ = 27.5)		37.2 (σ = 27.6)	
Performance	**CATiB**	**UD**	**CATiB**	**UD**
Tokenization Accuracy	99.1	99.1	100.0	100.0
LS	96.9	94.7	97.6	94.9
UAS	92.8	91.1	93.4	91.6
LAS	91.6	88.7	92.4	89.1
Time in Seconds	**CATiB**	**UD**	**CATiB**	**UD**
Reading Input	4.2	4.1	n/a	n/a
Preprocessing	141.2	141.4	n/a	n/a
Loading Model	2.6	2.8	4.1	3.5
Parsing	14.6	14.5	15.3	14.6
Printing Output	0.4	0.4	0.6	0.2
Total time	163.0	163.2	20.0	18.3
Sentence/Second	12.2	12.2	99.4	108.5
Words/Second	386.7	386.3	3,154.8	3,443.5
Tokens/Second	457.7	457.2	3,700.4	4,039.1

Table 2: Scores of various dependency parsing systems trained on the either CATiB or UD and evaluated on the development set of PATB-CATiB and PATB-UD/NUDAR.

and Figure 2 shows the parsed tree of the first sentence. Since **Palmyra 3.0** is open-source, other researchers may fork our frontend and backend repositories and add their own parsers.

3.3. Default Configuration Files

Additionally, **Palmyra 3.0** provides users with popular configurations such as the Universal Dependencies (UD) for both Arabic and English, and the Columbia Arabic Treebank (CATiB). Interface-wise, there is a drop-down of default configurations for users to choose from. Currently, these configurations are stored on **Palmyra 3.0**'s GitHub repository, and GitHub provides APIs that can be used to conveniently retrieve these files. Configuration files specify the tags and labels used in the formalism, and their keyboard shortcuts. As with **Palmyra 2.0**, **Palmyra 3.0** allows users to create and upload their own custom configuration files.

3.4. Undo/Redo Feature

Palmyra 3.0 fixes a major drawback of **Palmyra 2.0** by helping annotators quickly recover from previous mistakes using a simple linear undo/redo model. The dependency tree is stored in a history buffer (stack), and only the most recent action may be undone by performing an inverse operation (Jakubec et al., 2014). An important point to note is that when the user carries out a new action, the history buffer must be cleared (i.e. there is no action to be redone). We optimize for space by limiting the size of the undo stack to latest ten elements, and by clearing the stack every time the user moves to a different tree.

4. Evaluation

4.1. Experimental Setup

As a case study, we evaluate the performance of the CamelParser2.0 parser we use in **Palmyra 3.0** in terms of accuracy and speed. We evaluate parsing both CATiB and UD formalisms on the Dev set of the PATB (Maamouri et al., 2004), using corresponding models. We also report results under both gold and predicted tokenization to study the effect of tokenization errors on overall quality.

The CATiB model is trained on data from both the CamelTreebank (Habash et al., 2022) and PATB parts 1v4.1, 2v3.1 and 3v3.2 (Maamouri et al., 2004,Maamouri et al., 2010a,b, 2011) converted to CATiB (Habash and Roth, 2009). The UD model is trained only on data from PATB-UD/NUDAR (Taji et al., 2017). For PATB data, we followed the splits specified by Diab et al. (2013). Since the speed metrics rely on the hardware, we ran our parser on

20 cores of the AMD EPYC 7742 64-Core Processor at 2.25 GHz, a single Nvidia Tesla A100 GPU, and 75 GB of memory.

4.2. Accuracy

We report tokenization accuracy for predicted tokenization in the performance section of Table 2, Label Score (LS), Unlabeled Accuracy Score (UAS), and the Labeled Accuracy Score (LAS). The parser achieves highly accurate results for all metrics in both CATiB and UD under gold and predicted conditions. These results suggest the annotators will only need to correct about one of every ten tokens in the parsed trees.

4.3. Speed

We report the time taken for various steps in the parsing pipeline. If the parser is given annotated data with existing tokenization in the form of a CoNLL-X or CoNLL-U file (as opposed to raw text/white-space tokenized text in the predicted tokenization case), parsing is much quicker as there is no preprocessing and disambiguation. The parsing time numbers are averages over 10 runs of the parser, and we observe that gold tokenization parsing is approximately 10 times quicker than predicted tokenization parsing. We observe similar speeds between CATiB and UD parsing with gold tokenization parsing being able to process approximately 100 sentences per second and predicted tokenization parsing being able to process approximately 12 sentences per second (with $\sim$ 38 tokens per sentence on average). Although slower, the predicted tokenization speed is still satisfactory for **Palmyra 3.0** interface users.

5. Conclusion & Future Work

We presented **Palmyra 3.0**, a tool for morphology and dependency syntax annotation. **Palmyra 3.0** is web-based, supports using an automatic parser, and can be linked to the cloud. It is lightweight, easily extensible, and was designed with annotators in mind, to help increase annotator efficiency while providing a low entry barrier.

In the future, we plan to continue improving **Palmyra 3.0**'s functionality, and extend its use to other languages and annotation projects.

Acknowledgements

We would like to thank Go Inoue and Ossama Obeid for helpful conversations. This work was supported in part through the resources, services, and staff expertise of the High Performance Computing Center at New York University Abu Dhabi.

6. Bibliographical References

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. *The Prague Dependency Treebank*, pages 103–127. Springer Netherlands, Dordrecht.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Sufeng Duan, Hai Zhao, and Dongdong Zhang. 2023. Syntax-aware data augmentation for neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2988–2999.

Ahmed Elshabrawy, Muhammed AbuOdeh, Go Inoue, and Nizar Habash. 2023. CamelParser2.0: A State-of-the-Art Dependency Parser for Arabic. In *Proceedings of The First Arabic Natural Language Processing Conference (ArabicNLP 2023)*.

Gaël Guibon, Marine Courtin, Kim Gerdes, and Bruno Guillaume. 2020. When collaborative treebank curation meets graph grammars. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5293–5302, Marseille, France. European Language Resources Association.

Nizar Habash, Muhammed AbuOdeh, Dima Taji, Reem Faraj, Jamila El Gizuli, and Omar Kallas. 2022. Camel treebank: An open multi-genre Arabic dependency treebank. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2672–2681, Marseille, France. European Language Resources Association.

Nizar Habash and Ryan Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 221–224, Suntec, Singapore.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den

Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, pages 15–22. Springer, Netherlands.

Johannes Heinecke. 2019. Conllueditor: a fully graphical editor for universal dependencies treebank files. In *Proceedings of the Third Workshop on Universal Dependencies (UDW, SyntaxFest 2019)*, pages 87–93.

Go Inoue, Salam Khalifa, and Nizar Habash. 2022. Morphosyntactic tagging with pre-trained language models for Arabic and its dialects. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1708–1719, Dublin, Ireland. Association for Computational Linguistics.

Karel Jakubec, Marek Polák, Martin Nečaskỳ, and Irena Holubová. 2014. Undo/redo operations in complex environments. *Procedia Computer Science*, 32:561–570.

Talha Javed, Nizar Habash, and Dima Taji. 2018. Palmyra: A Platform Independent Dependency Annotation Tool for Morphologically Rich Languages. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan.

Dan Jurafsky and James H. Martin. 2009. *Dependency Parsing*. Pearson Prentice Hall.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The inception platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9. Association for Computational Linguistics. Event Title: The 27th International Conference on Computational Linguistics (COLING 2018).

A Kulmizev. 2023. *The Search for Syntax: Investigating the Syntactic Knowledge of Neural Language Models Through the Lens of Dependency Parsing*. Ph.D. thesis, Uppsala University.

Zuchao Li, Kevin Parnow, and Hai Zhao. 2022. Incorporating rich syntax information in grammatical error correction. *Information Processing & Management*, 59(3):102891.

Alexa Little and Stephen Tratz. 2016. Easytree: A graphical tool for dependency tree annotation. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Portorož, Slovenia.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *Proceedings of the International Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Joakim Nivre. 2014. Universal dependencies for Swedish. In *Proceedings of the Swedish Language Technology Conference (SLTC)*.

Dima Taji and Nizar Habash. 2020. Palmyra 2.0: A configurable multilingual platform independent tool for morphology and syntax annotation. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 168–177.

Dima Taji, Nizar Habash, and Daniel Zeman. 2017. Universal dependencies for Arabic. In *Proceedings of the Workshop for Arabic Natural Language Processing (WANLP)*, Valencia, Spain.

Yuanhe Tian, Han Qin, Fei Xia, and Yan Song. 2022. Syntax-driven approach for semantic role labeling. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 7129–7139, Marseille, France. European Language Resources Association.

Yue Zhang, Bo Zhang, Zhenghua Li, Zuyi Bao, Chen Li, and Min Zhang. 2022. SynGEC: Syntax-enhanced grammatical error correction with a tailored GEC-oriented parser. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2518–2531, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

7. Language Resource References

Mohamed Maamouri, Ann Bies, Seth Kulick, Fatma Gaddeche, Wigdan Mekki, Sondos Krouna, Basma Bouziri, and Wadji Zaghouani. 2010a. Arabic treebank: Part 1 v 4.1. Linguistic Data Consortium (LDC2010T13).

Mohamed Maamouri, Ann Bies, Seth Kulick, Fatma Gaddeche, Wigdan Mekki, Sondos Krouna, Basma Bouziri, and Wadji Zaghouani. 2011. Arabic treebank: Part 2 v 3.1. Linguistic Data Consortium (LDC2011T09).

Mohamed Maamouri, Ann Bies, Seth Kulick, Sondos Krouna, Fatma Gaddeche, and Wajdi Zaghouani. 2010b. Arabic treebank: Part 3 v 3.2. Linguistic Data Consortium (LDC2010T08).

Petr Pajas. 2008. Tred: Tree editor. http://ufal.mff.cuni.cz/~pajas/tred.

Yu Zhang. 2021. SuPar GitHub repository - v1.1.4.