Advancing African NLP: UDMorph and flexiPipe

Maarten Janssen
UFAL, Faculty of Mathematics and Physics
Charles University, Czechia
janssen@ufal.mff.cuni.cz

Abstract

In this paper, we present some of our recent efforts to provide baseline NLP pipelines for African languages. These include an infrastructure called UDMorph to make UD-compatible training data available for resources that do not have dependency relations, and a Python package called flexiPipe to easily run an NLP pipeline in various NLP tools using a uniform front-end, including the models provided by UDMorph. flexiPipe also provides Unicode normalization, an often overlooked feature that has a significant impact on African NLP. flexiPipe currently provides an NLP pipeline for 33 African languages, a significant increase from the handful of models that are currently easily accessible. And UDMorph is designed to make it easy to provide training data for more languages.

1 Introduction

In this paper, we describe a set of our recent initiatives to widen the set of languages for which NLP techniques are available. The focus is on traditional NLP pipelines such as tagging, parsing, lemmatization, NER, sentiment analysis, etc., and not on the creation of LLM models for a wide range of languages. These initiatives are not specifically aimed at African languages, but at languages around the world. However, with around 30% of all living languages being African, while only a small percentage of them have any type of NLP support, African languages naturally play an important role in these initiatives.

The NLP resources are organized around the Universal Dependencies initiative[1] (UD), the initiative behind probably the largest advance in providing NLP data for more languages, in tandem with computational projects that use those data to create NLP tools. The status of NLP support for African languages is exemplified by UD: of the 339 treebanks included in version 2.17 of UD, only 16 are for African languages, with 2 being for historic languages (Egyptian and Coptic), and 4 are for Hausa, meaning that only 12 current African languages are represented. Furthermore, the majority of those are not yet large enough or finished enough to be used for NLP training, and of the 169 models for 92 languages provided by UDPipe, only 2 are for African languages: Afrikaans and Wolof. With over 2000 living languages in Africa, that means that currently an NLP pipeline exists for only 0.1% of the African languages.

In this paper we focus on two initiatives that provide some advances in the support for African languages in NLP: UDMorph, an infrastructure for NLP resources, and flexiPipe, a Python package to facilitate NLP processing for many languages we have recently developed. flexiPipe currently provides for 33 African languages (see Table 2, indicating either full UD support or POS only), mostly drawing on the resources provided by Masakhane (Dione et al., 2023) and SADilaR (Mabuya et al., 2020). After presenting those two projects, we discuss some issues with the current models in terms of accuracy, and a perspective on how to potentially improve this in the future. And we demonstrate how these tools together can help to create a rapid increase in the number of African languages for which NLP support is available.

2 Multilingual Resources

We have been working on various initiatives to provide NLP resources for as many languages as possible. The aim of this is threefold: to provide NLP training data, tools, and searchable annotated data. As mentioned in the introduction, these initiatives are not specifically aimed at African languages, but due to the current NLP support and the large amount of languages in Africa, African languages

[1] https://universaldependencies.org/

Proceedings of the 7th Workshop on African Natural Language Processing (AfricaNLP 2026), pages 142–148

naturally play an important role. These three goals are interrelated: we aim to locate and create NLP training data, primarily with the objective to provide NLP models in as many languages as possible. And a primary objective of those NLP models is to annotate massively multilingual resources, to provide resources for linguistic studies as well as community-oriented NLP based services.

A current objective is to provide annotated versions of all texts in StoryWeaver[2], a website by the Indian publisher Pratham Books, which provides 68,000 children's stories in 382 languages. StoryWeaver has a strong focus on Indian languages, but one of the resources used to start the project was the African StoryBook project, which contains around 14,000 stories in 270 languages, primarily though not exclusively African languages.

2.1 UDMorph

UDMorph (Janssen, 2024) is an initiative aimed at providing an infrastructure that follows the set-up of the Universal Dependencies, but for resources without dependency relations, that do follow the UD guidelines for lemmatization and tagging. Like UD, it provides repositories with training data in CoNLL-U format. But it also provides an environment to help build training data for new languages in a guided manner, and provides models trained on the training data that can be downloaded or used as a REST service.

The first major source of UD compatible POS tagged data comes from various initiatives from the Masakhane community, such as KenPos (Indede et al., 2022) and MasakhaPOS[3]. All these resources have been converted to the format used by UDMorph, and can be used as searchable annotated corpora, as downloadable training data in the UD format, or as trained models in either the online GUI or the REST service. The Masakhane data only provides POS labels, so the models trained on those data do not provide lemmatization.

The second major source is the CTexT NCHLT Web Service[4] that provides various NLP pipelines for all official languages from South Africa. This service does not natively produce the data in a UD compatible format, but UDMorph provides an on-the-fly translation between the tagset used by CTexT and UD, to produce results that are almost entirely UD compatible.

The third major source are the official UD treebanks that are not currently included in the training of UDPipe or Stanza (Qi et al., 2020), since the data are either not yet large enough to reach a reliable dependency parser, or not yet fully split up into trainable parts. Since tagging is a computationally easier task than parsing, many of those treebanks that are excluded can be used to produce adequate taggers and lemmatizers.

2.2 flexiPipe

flexiPipe[5] is an open-source Python package we recently released, meant to facilitate the use of various existing taggers, parsers, and other NLP pipelines by providing a uniform front-end that can run processes through a number of different backends in a uniform fashion. This includes REST services like UDPipe2[6] and UDMorph, Python packages like Stanza[7], fasttext[8], HuggingFace transformer models, and SpaCy[9], and (legacy) command line tools like TreeTagger (Schmid, 1994) and UDPipe1. And it contains a Viterbi based tagger called flexitag, which is a reimplementation of NeoTag (Janssen, 2012). The package has a modular design that makes it easy to add additional backends, which can also be created as independent packages. For instance, there is a backend for Flair[10], but since flair does not currently provide models for anything but English and is relatively heavy, it is not part of the flexiPipe repository, but provided as an independent package that can install an additional backend for flexiPipe. This makes it easy to include new NLP tools in a world in which algorithms change rapidly.

flexiPipe is designed for ease of use, so that you can just use a simple command like:

```
$ flexipipe 'This is a short test of a short English sentence.'
```

and the system will automatically detect the language as English, look for an NLP pipeline for English, and produce the tagged output in CoNLL-U format. But you can also control the behaviour by explicitly selecting a backend and/or a model, manually providing the language to make sure the correct language is used, and specifying the tasks

[2] https://storyweaver.org.in/

[3] https://github.com/masakhane-io/masakhane-pos

[4] https://hlt.nwu.ac.za/

[5] https://github.com/ufal/flexipipe

[6] https://lindat.mff.cuni.cz/services/udpipe/

[7] https://stanfordnlp.github.io/stanza/

[8] https://fasttext.cc/

[9] https://spacy.io/

[10] https://flairnlp.github.io/

to be performed (lemmatization, tagging, parsing, ner) and the output format. When asked, the system will automatically download models if there is a model available for the language/backend. Between the various backends, flexiPipe currently provides an NLP pipeline for 154 different languages, which includes 33 modern African languages as listed in Table 2. As can be seen, due to the data sources, most of these models only provide a POS tag. The list of available models for each backend is provided via a Git repository[11], that can also host models trained in flexiPipe, or in one of the backends it provides. flexiPipe pulls the data from that repository, to make sure that the list of available models always stays up-to-date.

Apart from NLP processing, flexiPipe also provides a uniform tagging command for various backends, including UDPipe1, SpaCy, fasttext, and Stanza. You just have to provide the path to the folder with the training data, and specify the language and name for the model, and the backend to use, and the system will train a model in that backend. So you can train a model in various backends with the exact same command:

```
$ flexipipe train -backend fasttext -train-data /path/to/conllu -name my-swahili -language swa
```

Once trained, the model will directly be available for local use. This makes it very easy to train new models, and try them in several backends to see which method works best for the training data. The system will automatically verify which data are provided in the dataset, and train the appropriate tasks in the model (tagging, lemmatization, parsing).

Finally, flexiPipe has a built-in benchmark tester, making it easy to attempt training various backends on the same data, and get a comparison of how well each backend works for a specific language and dataset. Although the commands are easy, the training can of course take a long time depending on the backend and the available hardware.

flexiPipe is integrated into the TEITOK corpus platform (Janssen, 2016), which is used for various less-resourced language projects and forms the backend of UDMorph (where flexiPipe itself is the NLP backend used by UDMorph). flexiPipe can directly annotate TEI/XML files as used in TEITOK, and new models can be trained on TEITOK corpora, so that internally a model can be used that has been specifically trained on the local corpus, which for LRL or historic languages often outperforms off-the-shelf models. Locally trained models can easily be made available via the flexipipe-models repository. This means that flexiPipe not only provides an NLP pipeline for a large number of languages, but also provides an infrastructure to easily add models for new languages.

3 Accuracy Drop-off and Normalization

UDPipe2 is currently the best performing tagger and parser for many languages. According to the official measurements[12], the best model trained on UD2.17 (Latin-LLCT) is scoring 99.78% accuracy, and out of the 169 models, 18 models score over 99%, and 67 over 98%, and only 12 models score below 90%, with the worst scoring at 83.82%. The only two African languages in the list score as expected in that list: Wolof (95.29%), and Afrikaans (98.73%).

However, these accuracies are not necessarily real-world performance indicators: they are measured by taking a uniform treebank, like the English EWT treebank, which is split in three parts: 80% for training, 10% for fine-tuning the model once it is trained (dev), and 10% to test the final accuracy of the model. Since all parts are balanced parts taken from the same project, they have the same style, genre, transcription norms, etc. However, if we want to use the tagger on text of potentially different style, there is an accuracy drop-off: if we test the model trained on the EWT treebank for English on itself, we get an accuracy of 97.5%, but if we test it on the other English treebanks, the results are lower, as shown in Table 1, with the columns indicating the model used, and the rows the Treebank it is tested on. As you can see, the cross-treebank drop-off is noticeable, but relatively modest - the lowest score is when using the model trained on the spoken learner data on the small ATIS treebank, with only 90.2% accuracy - but a larger, more balanced model like EWT scores at least 94.8% when tested on any treebank, including the spoken ESL data. This means that a model like `english_ewt-ud-2.17-251125` can be expected to accurately tag any type of English text[13].

However, for other languages, including many

[11] https://github.com/ufal/flexipipe-models

[12] https://ufal.mff.cuni.cz/udpipe/2/models

[13] A large comparison like this can only be given for a few languages with various comparable models - even a well-supported language like German has only 2 UDPIPE model, and most language have only 1.

	ATIS	CHILDES	ESLSPOK	EWT	GUM	LINES	PARTUT
ATIS	99.1%	96.9%	90.2%	95.3%	96.1%	97.1%	97.1%
CHILDES	95.0%	96.8%	95.4%	96.4%	96.1%	95.6%	94.2%
ESLSPOK	94.9%	96.2%	98.6%	96.2%	95.6%	95.4%	94.1%
EWT	93.7%	96.6%	95.2%	97.5%	96.7%	94.4%	94.0%
GUM	95.4%	97.6%	96.6%	97.8%	98.3%	94.0%	94.0%
LINES	94.2%	95.2%	93.6%	94.8%	94.5%	97.8%	94.4%
PARTUT	94.5%	94.5%	92.7%	94.8%	93.0%	95.2%	97.7%

Table 1: Cross-model UPOS accuracy of English models

African languages, the drop-off is considerably more pronounced. For Yoruba, there are two training sets: on one hand, the training data provided by the KenPos project, which provides UPOS for a total of 43,598 tokens, and on the other hand, the Yoruba-YTB treebank from UD, which provides a full treebank for 8,243 tokens. When tested on the treebank they were trained on, both models perform adequately: 98.1% for the smaller yor-ytb model, and 98.5% for the KenPos model. However, if we straightforwardly test the KenPos model on the YTB treebank, it only scored 60.3%; and if we test the YTB model on the KenPos dataset, it scores even lower: 46.2%. There are various factors that can be pointed out as possible causes for this large drop in accuracy. The first is that the YTB is liturgical language from the Bible, while the KenPos data are modern Yoruba largely from the web. The second is a difference in orthography: YTB always uses tones, while KenPos uses them often but not always. A third difference is a difference in interpretation of the grammar: the word *àwọn* (the/that) is described in traditional grammars as a plural determiner, and the YTB treebank follows that standard. But since *àwọn* can also be used independently, KenPos treats it as a pronoun. These differences in style, dialect, orthography and tagging strategy cannot easily be resolved, and make for a much more pronounced accuracy drop-off. Taken together, these differences in style, dialect, orthography, and tagging strategy constitute a qualitative error analysis that points to data and annotation mismatches, rather than purely model limitations, as the main cause of the pronounced accuracy drop-off.

3.1 Unicode Normalization

However, there is one difference between the two Yoruba datasets that is often overlooked due to the fact that it does not appear in English: in the YTB treebank, diacritics are stored as combined UTF characters (NFC), while in the KenPos data, they are stored as separate characters (NFD). And this is not specific for Yoruba, but occurs in many languages with diacritics, especially when not all combined characters exist in UTF - which is the case for many African languages. This difference in UTF normalization means that when naively running a benchmark, some things are counted as incorrect that should not be, since the result is just a difference in normalization. But it also means that words that are in principle known from the training data are nevertheless treated as OOV when they are sent in the wrong format to the backend, which not only affects those tokens, but also their context. That is why flexiPipe keeps track of which UTF normalization was used in each model, and sends the text in that format, and after that always produces the same format in the output, by default using NFC, to ensure consistency in downstream tasks. If we normalize the KenPos training data to NFC before training the model, and then also normalize in the benchmark comparison, testing the KenPos model against the YTB treebank gives a 66.2% accuracy rate (up from 60.3%), and the reverse testing goes up to 60.0% (up from 46.2%), resulting in a 30% accuracy boost by simply normalising the diacritics. And it should be noted that the errors hence include cases where *àwọn* is tagged as a PRON or a DET, which is not actually an error in the result but rather an intentional tagging difference, meaning that the actual real-world performance is higher than that (but not easily quantifiable).

4 Conclusion

The various projects listed in this paper, and especially UDMorph and flexiPipe, provide easy access to NLP pipelines for many more languages than typically available. This work builds upon

and amplifies the foundational contributions of the African NLP community, particularly Masakhane and SADilaR, where all of the training data and models originate from, by providing the infrastructure to make their models easily accessible and usable. It integrates the models trained on those data into an easy-to-use pipeline, making NLP processing for African languages much more accessible. Not only that, but it also provides an easy-to-use framework to create new training data for more (African) languages, train models on them in a choice of different backends, and have a place to make those models available so that they can be used alongside the many models already available. Together, UDMorph and flexiPipe have already enabled NLP processing for 33 African languages, which is 10 times as many as the 3 languages provided by UDPipe and Stanza together.

Of course, in this day and age in which large language models dominate the NLP landscape, the role of traditional NLP pipelines for lemmatization, tagging, parsing, NER, etc. is smaller than it used to be, but traditional pipelines still play an important role in the creation of searchable language resources, which in turn can help not only for linguistic research, but also in language teaching, and improving the accessibility of linguistic resources in general.

The increase from 3 to 33 models for African languages (available via easily accessible tools) is a significant increase, but still only represents a bit over 1% of the African languages. To reach a more significant percentage, much work on building training data is required by the African NLP community, but we hope that the resources and tools presented in this paper will facilitate that process, as well as make it possible to directly convert any additional training data into accessible NLP tools.

Limitations

Even though flexiPipe provides NLP support for many new languages in an easy to use environment, the information provided is limited for many of them when compared to UD models: most Masakhane models only provide a UPOS and do not provide detailed morphosyntactic analysis, lemmatization or dependency relations. The models trained on the UD treebanks that are not included in UDPipe have less accuracy since the amount of training data is still limited.

Of course, in this day and age, where large language models dominate the NLP landscape, the role of traditional NLP pipelines for lemmatization, tagging, parsing, NER, etc. is smaller than it used to be, but traditional pipelines still play an important role in the creation of searchable language resources, which in turn can help not only for linguistic research, but also in language teaching and in improving the accessibility of linguistic resources in general. And traditional NLP pipelines are less computationally demanding and hence faster, deterministic and hence more predictable, and even currently most often more accurate than generative models are for these tasks. Moreover, UDMorph and flexiPipe can complement LLM-based approaches for African languages by providing high-quality tagged and parsed corpora for fine-tuning or evaluation, by supplying linguistically enriched input to retrieval-augmented or tool-augmented LLM systems, and by enabling automatic checks of LLM outputs against UD-style analyses.

For underperforming models, an in-depth analysis of the errors to see whether they are likely due to data sparsity, annotation scheme mismatches, or architecture limitations would be very helpful, but will have to remain for future work involving native speakers of the languages under analysis.

Acknowledgments

The models for African languages presented in this paper would of course not have been possible without the work of the people at Masakhane, SADiLaR, and the various UD treebanks, as well as the researchers behind UDPipe, SpaCy, and Stanza. This work has been developed in the framework of the LINDAT/CLARIAH-CZ Research Infrastructure (https://lindat.cz), supported by the Ministry of Education, Youth and Sports of the Czech Republic (project no. LM2023062).

References

Cheikh M. Bamba Dione, David Ifeoluwa Adelani, Peter Nabende, Jesujoba Alabi, Thapelo Sindane, Happy Buzaaba, Shamsuddeen Hassan Muhammad, Chris Chinenye Emezue, Perez Ogayo, Anuoluwapo Aremu, Catherine Gitau, Derguene Mbaye, Jonathan Mukiibi, Blessing Sibanda, Bonaventure F. P. Dossou, Andiswa Bukula, Rooweither Mabuya, Allahsera Auguste Tapo, Edwin Munkoh-Buabeng, and 25 others. 2023. MasakhaPOS: Part-of-speech tagging for typologically diverse African languages. In *Proceedings*

of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10883–10900, Toronto, Canada. Association for Computational Linguistics.

Florence Indede, Owen McOnyango, Lilian D.A. Wanzare, Barack Wanjawa, Edward Ombui, and Lawrence Muchemi. 2022. KenPos: Kenyan Languages Part of Speech Tagged dataset.

Maarten Janssen. 2012. NeoTag: a POS tagger for grammatical neologism detection. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2118–2124, Istanbul, Turkey. European Language Resources Association (ELRA).

Maarten Janssen. 2016. TEITOK: Text-faithful annotated corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4037–4043, Portorož, Slovenia. European Language Resources Association (ELRA).

Maarten Janssen. 2024. UDMorph: Morphosyntactically tagged UD corpora. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16933–16940, Torino, Italia. ELRA and ICCL.

Rooweither Mabuya, Dimakatso Mathe, Mmasibidi Setaka, and Menno Zaanen. 2020. Digitizing humanities in south africa: Computational linguistic resources, training, and community building. *Pop! Public Open Participatory*, 02.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees.

Table 2: African Languages in Flexipipe

Language	Country(s)	Backend	Data source	Features
Bambara (bam)	Mali	UDMorph	Masakhane	POS
Ewe (ewe)	Ghana, Togo	UDMorph	Masakhane	POS
Fon (fon)	Benin	UDMorph	Masakhane	POS
Hausa (hau)	Nigeria, Niger	UDMorph	Masakhane	POS
Igbo (ibo)	Nigeria	UDMorph	Masakhane	POS
Naijá (pcm)	Nigeria	Stanza, UDPipe	UD Treebank	Full
Twi (twi)	Ghana	UDMorph	Masakhane	POS
Wolof (wol)	Senegal, Gambia	Stanza, UDPipe	UD Treebank	Full
Yoruba (yor)	Nigeria, Benin	UDMorph	Masakhane	POS
Lubukusu (bxk)	Kenya	UDMorph	Masakhane	POS
Kinyarwanda (kin)	Rwanda	UDMorph	Masakhane	POS
Luganda (lug)	Uganda	UDMorph	Masakhane	POS
Lumarachi (lri)	Kenya	UDMorph	Masakhane	POS
Dholuo (luo)	Kenya	UDMorph	Masakhane	POS
Lulogooli (rag)	Kenya	UDMorph	Masakhane	POS
Swahili (swa)	East Africa	UDMorph	Masakhane	POS
Ghomala (bbj)	Cameroon	UDMorph	Masakhane	POS
Moore (mos)	Burkina Faso	UDMorph	Masakhane	POS
Nyanja (nya)	Malawi, Zambia	UDMorph	Masakhane	POS
Shona (sna)	Zimbabwe	UDMorph	Masakhane	POS
Sesotho (sot)	Lesotho, South Africa	UDMorph	CText	POS
Siswati (ssw)	Eswatini, South Africa	UDMorph	CText	POS
Setswana (tsn)	Botswana, South Africa	UDMorph	CText	POS
Xitsonga (tso)	South Africa	UDMorph	CTexT	POS
Tshivenda (ven)	South Africa	UDMorph	CTexT	POS
Afrikaans (af/afr)	South Africa	Stanza, UDPipe	UD Treebank	Full
Khoekhoe (naq)	Namibia, South Africa	UDMorph	CTexT	POS
isiNdebele (nde)	Zimbabwe, South Africa	UDMorph	CTexT	POS
Sesotho sa leboa (nso)	South Africa	UDMorph	CTexT	POS
isiXhosa (xho)	South Africa	UDMorph	CTexT	POS
isiZulu (zul)	South Africa	UDMorph	CTexT	POS
Maghrebi arabic french (ary)	North Africa	Stanza	Stanza	Full
Kabyle (kab)	Algeria	UDPipe	UD Treebank	POS