PoS Tagging, Lemmatization and Dependency Parsing of West Frisian

Wilbert Heeringa[1], Gosse Bouma[2], Martha Hofman[1], Jelle Brouwer[2], Eduard Drenth[1], Jan Wijffels[3], Hans Van de Velde[1,4]

[1]Fryske Akademy, [2]University of Groningen, [3]BNOSAC, [4]Utrecht University
[1]Leeuwarden, [2]Groningen, [3]Brussels, [4]Utrecht
[1]{wheeringa, mhofman, edrenth, hvandevelde}@fryske-akademy.nl,
[2]{g.bouma, jelle.brouwer}@rug.nl, [3]jwijffels@bnosac.be

Abstract

We present a lemmatizer/PoS tagger/dependency parser for West Frisian using a corpus of 44,714 words in 3,126 sentences that were annotated according to the guidelines of Universal Dependencies version 2. PoS tags were assigned to words by using a Dutch PoS tagger that was applied to a Dutch word-by-word translation, or to sentences of a Dutch parallel text. Best results were obtained when using word-by-word translations that were created by using the previous version of the Frisian translation program *Oersetter*. Morphologic and syntactic annotations were generated on the basis of a Dutch word-by-word translation as well. The performance of the lemmatizer/tagger/annotator when it was trained using default parameters was compared to the performance that was obtained when using the parameter values that were used for training the LassySmall UD 2.5 corpus. We study the effects of different hyperparameter settings on the accuracy of the annotation pipeline. The Frisian lemmatizer/PoS tagger/dependency parser is released as a web app and as a web service.

Keywords: Frisian, lemmatization, PoS tagging, dependency parsing, Universal Dependencies

1. Introduction

We present a corpus of lemmatized, tagged and annotated text in the West Frisian language together with a web application and web service which can be used for lemmatization, part-of-speech (PoS) tagging and dependency parsing of Frisian text. West Frisian is an autochthonous low-resource language spoken in the Dutch province of Fryslân where it is recognized as an official language, in addition to Dutch. In 2018 about 400,000 native speakers were reported (Klinkenberg et al., 2018).

1.1. Choice of methodology

A fast way getting a tool that can lemmatize, tag or parse West Frisian text may be the use of an approach that uses transformer-based context-sensitive language models such as BERT (Bidirectional Encoder Representations from Transformers). BERT was introduced by Google AI Research (Devlin et al., 2019) and is a language representation model pre-trained by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer on several NLP tasks. While BERT was trained on English, multilingual BERT (mBERT) has been trained on the 104 languages with the largest Wikipedias, including West Frisian.[1]

Multilingual BERT was used in the multilingual dependency parsing model UDapter that was developed by Üstün et al. (2020). With UDadater contextualized word representations are obtained. During training adapters were used to capture both task-specific and language-specific information. Adapters are small modules added between layers of a pre-trained network. The adapters were enabled to learn via language embeddings while sharing model parameters across languages. The language embeddings were learned by leveraging a mix of linguistically curated and predicted typological features. Those features were obtained from the URIEL language typology database (Littell et al., 2017). The authors found that UDapter outperforms strong monolingual and multilingual baselines on the majority of both high-resource and low-resource languages, where a zero-shot scenario was assumed for the latter (i.e. no task specific data was seen during task-specific fine-tuning for low-resource languages). Average POS-tagging accuracy for low-resource languages in this setting was 58.4% while LAS for dependency parsing was 36.5%.

Both monolingual and a multilingual BERT were used by De Vries et al. (2021). Three monolingual BERT models of English, German and Dutch (called BERTje (De Vries et al., 2019)) and a multilingual BERT were used in order to investigate whether the linguistic structure can be transferred to two regional languages in The Netherlands – Frisian and Gronings – by learning new sub-word embeddings. The training procedure consisted of two separate fine-tuning steps. In the first step the transformer layers in the three monolingual BERT models and mBERT were fine-tuned for the PoS tagging task. Independently, in the second step for each BERT model new lexical layers were trained for the

[1]See https://github.com/google-research/bert/blob/master/multilingual.md.

two target languages with a masked language modeling pre-training objective, where 15% of the input tokens were masked and a model is learned to predict the missing tokens. This gives the model the opportunity to learn textual patterns from unlabeled data. Finally, the retrained lexical layer and the fine-tuned transformer layers were combined to yield a PoS tagging model that was then adapted to the target language. The performance of the language models was evaluated on PoS annotated data of Gronings and West Frisian. For all monolingual models, task performance greatly improved by retraining the lexical layer for Gronings and West Frisian. Best results were obtained by using the Dutch language model fine-tuned on the Dutch Alpino dataset (Van Noord, 2006):[2] 92.4% for Gronings and 95.4% for Frisian. For Gronings the PoS tags of the data set that was used for evaluation were obtained with a UD model trained on the basis of Alpino and applied to a Dutch word-by-word translation of the Groningen text. For Frisian the data set that was used for evaluation was an earlier version of the corpus that is presented and described in this paper. The better performance for Frisian may be explained by the fact that the PoS tags in the Frisian data set were manually verified afterwards, while those in the Groningen data set were not.

Although the multilingual models work reasonably well, for languages for which no annotated data is available during training, the performance lags behind languages for which data is available. Even a small amount of data can already help a lot (see Lauscher et al. (2020)). For that reason, we want to build a large Frisian annotated corpus anyway, which will be further extended over time. We then also opt for the classic methodology in which a lemmatizer/tagger/parser is trained on the basis of this corpus. We plan to do fine tuning with mBERT or with a comparable multilingual model in the future.

1.2. Building the corpus

If annotated data is not yet available yet for a language, it may seem obvious to build the corpus from scratch. However, for smaller languages like Frisian the means for developing language tools are more restricted than for larger languages like Dutch or English. Therefore, scholars tried to develop more efficient strategies that make the development of tools for small languages possible as well. One strategy is taking a route via a larger and related language for which the desired tools or models have already been developed.

As for building the corpus, the simplest strategy that has been proposed is to tag texts of a small language by a tagger that is trained on the basis of tagged corpora of a closely-related language. This approach is mentioned by Scherrer (2014) who tagged Catalan data using a Spanish PoS tagger. He obtained an accuracy of 58.42%. This approach was also applied by Tjong Kim Sang (2016) to 17th-century Dutch texts which he tagged with a modern Dutch PoS tagger. He obtained accuracies of 62.8% and 63.7%. The low accuracies show that this approach is not enough, but it can serve as a baseline to which other approaches can be compared. In the case of West Frisian, we consider using a tagger that is trained on the basis of Dutch texts.

Higher accuracies of up to 89.08% were obtained by Scherrer (2014) when the Spanish tagger was adapted to Catalan. Words in the tagger parameter files were translated into the low-resource language by using translation dictionaries that were created with unsupervised lexicon induction techniques that rely only on raw textual data.

Results can also be improved by adapting the spelling of corpora prior to tagging them with a tagger. For example, Hupkes and Bod (2016) investigated PoS tagging of 17th-century Dutch texts. They found "that modernizing the spelling of corpora prior to tagging them with a tagger trained on contemporary Dutch results in a large increase in accuracy, but that spelling normalization alone is not sufficient to obtain state-of-the-art results" (p. 77).

The approach of Tjong Kim Sang (2016) goes one step further by not just adapting the 17th-century Dutch texts, but translating them word for word into Dutch. He evaluated four word-by-word translation methods. In the first method, the machine translation system Moses (Koehn and Hoang, 2009) was used with two versions of the Dutch Statenvertaling Bible, one from the year 1637 and one from 1888. The second approach used the Integrated Language Bank (GTB) (Instituut voor de Nederlandse taal, 2022), an online collection of historical dictionaries, with links of historical words to their modern counterparts. The lexicon service makes it possible to retrieve modern lemmas for historical words. The third used a lexicon that was learned from two versions of a Dutch Bible, one from the 17th century and one from the 19th century. The fourth method employed orthographic rules learned from the learned lexicon. The rules converted historical character sequences to their modern equivalent. The lexicon-based methods outperformed the method that used orthographic rules.

Sometimes linguists are in the fortunate position that a translation for the text in the small language is already available in the large language, i.e. when a parallel corpus is available. In that case the approach of Yarowsky et al. (2001) can be applied who tagged the source side of the parallel corpus with an existing tagger and projected the tags along the word alignment links on the target side. Then a new tagger was trained on the target side.

In line with the authors just mentioned, we decided not to build the corpus from scratch, but rather to take a route via Standard Dutch. We investigated some of the techniques mentioned above.

[2] See `https://universaldependencies.org/treebanks/nl_alpino/index.html`.

1.3. Outline

In Section 2 the building of the corpus is described, and different procedures are compared to each other. In Section 3 the training of the lemmatizer/tagger/annotator is described and performance measurements are provided. In Section 4 a web app and a web service are presented and some results are shown. Conclusions and future prospects are given in Section 5.

2. Building the corpus

2.1. Source texts

We aimed to develop a Frisian lemmatizer/PoS tagger/dependency parser that can be used for a wide range of texts. Therefore, texts of different genres should be included in the corpus. We considered several (well-known) corpora that were PoS tagged: the Brown Corpus, the Lancaster-Oslo-Bergen Corpus, the Stockholm-Umeå Corpus, the Balanced Corpus of Contemporary Written Japanese, the NOAH's Corpus of Swiss German Dialects, the STEVIN reference corpus of 500 million words, the SoNaR corpus. In all corpora we found news texts. Novels, legal texts/laws and Wikipedia texts were found in three corpora, and scientific texts were found in two corpora. We decided to include texts of the same genres in our newly built corpus.

Texts were selected from the Frisian-Dutch parallel corpus that was used for the previous version of the online Frisan/Dutch - Dutch/Frisian translation program *Oersetter* 'translator'.[3] The corpus consists of 120 texts that are available both in Frisian and Dutch. The Frisian texts consist of 3,002,327 words, and the Dutch texts consist of 2,941,614 words.

We added texts from the Frisian broadcast station *Omrop Fryslân* as well as the Wikipedia text 'Frysk'. The distribution of the genres in the newly compiled corpus is shown in Table 1. Legal texts are not included yet.

2.2. Universal dependencies

We decided to annotate the corpus according to the guidelines of Universal Dependencies (UD) version 2[4] since UD makes cross-linguistical comparison of treebanks possible.[5]

Mostly we could apply the UD guidelines to Frisian in same way as has been done for Dutch in the Lassy Small corpus. The most striking difference compared to Dutch is the fuse of a verb in the second person singular with the pronoun that follows, for example, *Bist do* 'are you' is shortened to *Bist*, or *Hast do* 'have you' may be shortened to *Hasto*. We decided to tag these contracted words as verbs or auxiliaries (depending on the other content of the sentence) rather than as a pronoun or adding a new UD tag.

Another decision needed to be made with regard to relative pronouns like *dy't* 'who/that', *dêr't* 'where' and *wêr't* 'where'. Some scholars claim that *'t* represents a reduced form of the conjunction *dat* 'that', but this is also questioned (Dijkstra et al., 2017). We considered a word followed by *'t* as a whole, i.e. *'t* was not tagged individually.

2.3. Adding lemmas

In the corpus lemmas of the tokens were added manually. When the corpus consisted of about 30,000 tokens, a lemmatizer was trained on the basis of the lemmatized Frisian texts using the function `udpipe_train` from the R package `udpipe` (Wijffels, 2020). Using this tagger, newly added corpus texts were lemmatized, and the lemmas were manually corrected. The techniques that are made available in the R package `udpipe` are described by Straka et al. (2016) and Straka and Straková (2017).

2.4. Adding PoS tags

After the selection of the text to be included in the corpus, we could have added PoS tags to the words manually. However, we felt that the approaches like the ones introduced by Yarowsky et al. (2001), Scherrer (2014) and Tjong Kim Sang (2016) may speed up the process of adding PoS tags since they take a route via a larger and closely related language for which a PoS tagger is already available. We investigated three procedures.

2.4.1. Applying a Dutch PoS tagger directly to Frisian

For adding PoS tags three procedures were investigated. In the first procedure the Frisian text was tagged with a Dutch PoS tagger that was trained with the UD Dutch LassySmall 2.8 treebank (Bouma and Van Noord, 2017). This corpus contains sentences from the Wikipedia section of the Lassy Small Treebank. The Lassy Small Treebank is a manually verified treebank for Dutch. Since not all material in the Lassy Small Treebank could be made freely available, only the material from the Wikipedia (wiki) section was included in the UD Dutch LassySmall treebank (Van Noord et al., 2013).

2.4.2. Tagging via a Dutch word-by-word translation

In the second procedure we followed Tjong Kim Sang (2016). A Dutch word-by-word translation is made from the the Frisian text. This word-by-word translation is tagged by the Dutch PoS tagger that was trained with the LassySmall UD 2.8 corpus.[6] The tags of the Dutch words are projected on the corresponding Frisian

[3] This version was online available until December 15th, 2021.

[4] See `https://universaldependencies.org/guidelines.html`.

[5] See `https://universaldependencies.org/introduction.html`.

[6] The corpus is found at `https://github.com/UniversalDependencies/UD_Dutch-LassySmall`.

source	#tokens	%tokens	#words	%words	#sentences	%sentences
news	8,737	17	7,998	17	582	19
science	2,293	4	2,069	5	107	3
novels	17,176	34	14,272	32	1,446	46
museum	9,275	18	8,335	19	486	16
Wikipedia	13,780	27	12,040	27	505	16
total	51,261		44,714		3,126	

Table 1: Distribution of genres across the newly compiled corpus. Novels only include a fragment of a novel, and not shorter literary prose such as short stories and columns.

words. The need for a word-by-word translation can be illustrated by the following Frisian sentence:

It die bliken dat dit útlein wurde moast
It did appear that this explained be must

Both Google Translate and the Frisian translation program at `frysker.nl` give the following translation when the sentence is submitted in its entirety for translation:

Het bleek dat dit uitgelegd moest worden
It appeared that this explained must be

The number of words in the Dutch sentence and the word order differ from the Frisian sentence. Therefore, the PoS tags of the words in the Dutch sentence cannot be easily projected on the words of the Frisian sentence. A word-by-word translation would be:

Het deed blijken dat dit uitgelegd worden moest
It did appear that this explained be must

Using this sentence the PoS tags of the Dutch words can simply be projected on the words at the corresponding positions in the Frisian sentence.
In order to obtain word-by-word translations, we investigated three translation programs:

- Google Translate. Google Translate uses a neural machine translation engine called *Google Neural Machine Translation* (GNMT). With GNMT the quality of the translation is improved by applying an example-based machine translation method in which the system "learns from millions of examples" (Schuster et al., 2016). We called the API by using the function `gl_translate` from the R package `googleLanguageR` (Edmondson, 2020). As Turovsky (2016) wrote, GNMT translates "whole sentences at a time, rather than just piece by piece..." This makes Google Translate less suitable for generating a word-by-word translation. The best way to translate an individual word is submitting the word being enclosed in single quotes.
- The previous version of the online Frisan/Dutch - Dutch/Frisian translation program *Oersetter* 'translator'[7]. This program was a statistical machine translation (SMT) system. A parallel training corpus had been compiled and been used to automatically learn a phrase-based SMT model(Van Gompel et al., 2014). The translation system was built around the open-source SMT software Moses. In order to obtain a word-by-word translation the words of a sentence were individually submitted to this program. We used the web service of the *Oersetter*.
- The current version of the *Oersetter*[8]. This program is a neural machine translation (NMT) system. More specifically a transformer-based sequence-to-sequence model is employed. Marian-NMT is used as the software that powers the program. With the current version of the *Oersetter* higher scores on automatic evaluation measures were achieved compared to the previous version.[9] Like Google Translate this program is intended to translate word phrases or sentences. In order to get an individual word translated by an individual word, the word to be translated should be put between single quotes. We used the web service of this program[10]

We should be aware that all three programs are intended for translating whole sentences rather than translating single words.

2.4.3. Tagging via an aligned Dutch parallel text

In the third procedure the approach that was initiated by Yarowsky et al. (2001) was used. The Dutch parallel text was tagged with the tagger that was trained with the LassySmall 2.8 corpus. Then the sentences of the Dutch texts were aligned with the corresponding sentences in the Frisian texts using `fast_align` (Dyer et al., 2013)[11] and the tags of the words in the Dutch

[7] Until December 15, 2021 available at `https://taalweb.frl/oersetter`.

[8] Available at `https://frysker.nl`.

[9] See experiments 5 and 6 at `https://bitbucket.org/fryske-akademy/oersetter2-pipeline/src/master/data/evaluation/README.md`.

[10] We used the version that was online on December 18th, 2021.

[11] See `https://github.com/clab/fast_align`.

sentences were projected on the corresponding Frisian words in the Frisian sentences.

2.4.4. Comparison of the three approaches

We tested the different procedures to the scientific texts, the museum texts, a part of the news texts and a part of the novel texts. Remember that in each of the strategies mentioned above, the PoS tags are generated by the same Dutch PoS tagger that was trained with the LassySmall UD 2.8 corpus, and that the results in this section relate to different translation techniques or even the absence of translation. The results are shown in Table 2. The highest accuracy was found when tags were obtained via a Dutch word-by-word translation that was obtained with the previous version of the Oersetter.

	% correct
Baseline	51.5%
Alignment with Dutch text	74.2%
Google Translate	76.0%
Oersetter, previous version	89.8%
Oersetter, current version	87.1%

Table 2: Percentages of correct PoS tags per translation procedure.

			p value
Align. Dutch	$>$	Baseline	$< .0001$
Google Transl.	$>$	Align. Dutch	$< .001$
Oersetter curr.	$>$	Google Transl.	$< .0001$
Oersetter prev.	$>$	Oersetter curr.	$< .0001$

Table 3: Comparison of percentages of correct PoS tags among translation procedures.

Using Fishers exact test the percentages of correct PoS tags among the procedures were compared. The results are presented in Table 3. From this table it can be concluded that the previous version of the Oersetter performs significantly better than any of the other translation procedures. The lower percentage for the technique where Frisian and Dutch sentences are aligned to each other can be explained by that fact that the aligner was not able to process swaps correctly.

We found that the previous Oersetter outperforms the current Oersetter. The explanation for this may have to do with the fact that the current Oersetter will sometimes derive a Dutch word from a Frisian word rather than choosing a Dutch word directly. This may cause the Oersetter to prefer a translation that is visually more similar to the Frisian word. For example, the Frisian word *hân* may be a verb 'had' (past participle of 'to have') or a noun 'hand'. If it is a verb, the Dutch translation should be *gehad*, but when it is a noun the translation should be *hand*. However, we found that when *hân* was used as a verb in a Frisian sentence, the new Oersetter translates this as *hand*, while the previous Oersetter correctly translates *gehad*.

By the time that we built the corpus, we started tagging the corpus via the Dutch text that was aligned to the Frisian text. Later on we tagged the corpus via Dutch word-by-word translations that were generated by the previous version of the Oersetter program. The tags were corrected manually. When the corpus consisted of about 30,000 tokens, we were able to train a Frisian tagger that had a moderate accuracy. We used the function `udpipe_train` from the R package `udpipe`. Newly added Frisian texts were tagged with this tagger and manually corrected afterwards.

2.5. Adding morphologic and syntactic annotations

Initially, 575 Dutch sentences from the Oersetter parallel corpus were morphologically/syntactically annotated with a Dutch UD annotator, and the annotations were projected on the corresponding Frisian sentences. Corrections of the annotations were made where alignments between Dutch and Frisian sentences went wrong. Later, Dutch word-by-word translations were generated by the previous version of the Oersetter and subsequently annotated. Before annotating, the translations were manually corrected. A Dutch annotator was used that was trained on the Lassy Small corpus using the function `udpipe_train` from the R package `udpipe`.

Since correction work is involved in both procedures, it does not really matter which one is used. But the latter method also works if there is no Dutch translation of the Frisian text available.

3. Training and performance

Once a fully lemmatized, tagged and annotated Frisian corpus had been obtained, we were able to train the UDPipe lemmatizer/tagger/annotator on the basis of this corpus. The methodology is described by Straka and Straková (2017). The lemmatizer uses a guesser that produces (lemma rule, universal PoS tag) pairs. The lemma rule generates a lemma from a word by stripping some prefix and suffix and prepending and appending new prefix and suffix. The guesser generates the results according to the last four characters of a word and by using word prefix. The disambiguation is performed by an averaged perceptron tagger. The PoS tagger uses a guesser that generates several triplets for each word according to its last four characters. The generated tags are disambiguated by an averaged perceptron tagger with a fixed set of features. The dependency parser "utilizes fast transition-based neural dependency parser. The parser is based on a simple neural network with just one hidden layer and without any recurrent connections, using locally normalized scores."

When training UDPipe on the basis of our Frisian corpus, we evaluated the use of two sets of hyperparameters. We also compared the results of our Frisian

corpus to the results of the – much larger – Dutch LassySmall UD 2.8 corpus. The results of this comparison may give us an indication whether our corpus needs further extension.

3.1. Training Frisian with the default settings

We started by randomizing the order of the sentences in the lemmatized/tagged/annotated data set and split the sentences in training data (80% of the sentences), validation data (10%) and test data (10%).

A lemmatizer, PoS tagger and dependency parser was trained by using the function `udpipe_train` in the R package `udpipe` (Wijffels, 2020) [12] on the basis of the training data. As hyperparameter settings for the tokenizer, PoS tagger, lemmatizer and dependency parser the default values were used.[13]

We present the tagger, lemmatizer and parser performance, measured on the testing portion of the data, evaluated in three different settings: using raw text only, using gold tokenization only, and using gold tokenization plus gold morphology. A k-fold cross-validation was carried out with k=10. First, the data set was split into k sets. Then k times (or folds) a set is selected as test set and another set is selected as validation set. When the sets are numbered from 1 to k in the ith fold set i is selected as test set, and set $i + 1$ is selected as validation set. If i is k the first set is selected as validation set. The joint remaining $k - 2$ sets are used as training sets. The results are shown in Table 4 in the appendix and show the mean and standard deviation across the 10 folds per metric and for each of the three settings.

The accuracy of the tokenization of words and sentences is represented by 'f1 words' and 'f1 sents' respectively. The f1 score is the harmonic mean of the precision and recall. UPOS represents the accuracy of the universal PoS tags, and XPOS the accuracy of language-specific PoS tags. UFeats is the accuracy of the morphological features from the universal feature inventory. 'Lemma' represents the accuracy of the lemmatizer. UAS represents the unlabeled attachment score, and LAS is the labeled attachment score.

3.2. Training Frisian with the LassySmall UD 2.5 settings

Given the close relationship between Frisian and Dutch - both languages belong to the West Germanic language family -, we also trained a Frisian lemmatizer, PoS tagger and dependency parser using the hyperparameter settings that were used for training on the basis of the Dutch LassySmall UD 2.5 corpus.[14]

The dependency parser makes use of FORM, UPOS, FEATS, and DEPREL embeddings. The form embeddings are precomputed with word2vec using the training data. For this purpose we used the R function `word2vec` from the `word2vec` package. The model was trained using skip-gram in dimension 50, a window of size 10, 5 negatives and 15 iterations. A word should occur at least 2 times in order to be included in the training. The other embeddings are initialized randomly, and all embeddings are updated during training (see Straka and Straková (2017)).

The results are shown in Table 5 in the appendix. Compared to the results in Table 4, we found that the accuracy of the lemmatizer has improved significantly. ($p < 0.0001$)[15] both for 'Raw text' and 'Gold tok'. Therefore, for further training we will use the LassySmall UD 2.5 settings rather than the default settings.

Alternatively, we used pre-trained word vectors that were provided by fastText.[16] They were trained on Common Crawl and Wikipedia, using fastText, CBOW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives.

The results are shown in Table 6 in the appendix. Compared to the results in Table 5, we found a significant improvement for 'f1 sents' (93.1% versus 89.7%). For 'f1 words' the new score was significantly lower (99.9% versus 100.0%).

3.3. Training Dutch with the LassySmall UD 2.5 settings

A lemmatizer, PoS tagger, and dependency parser were trained using the LassySmall UD 2.8 corpus with the hyperparameter settings that were used for training on the basis of the LassySmall UD 2.5 corpus. A word2vec model was trained using the R function `word2vec` from the `word2vec` package. The model was trained using skip-gram in dimension 50, a window of size 10, 5 negatives and 15 iterations. A word should occur at least 2 times in order to be included in the training. The LassySmall UD 2.8 corpus as it is available at GitHub is much larger than the current Frisian corpus and consists of 7,341 sentences, 83,571 words and 98,242 tokens when we put the train, development, and test sections together. The performance results are shown in Table 7 in the appendix.

We compared the results with the performance results in Table 5. For all metrics we found significant differences ($p < 0.0001$), usually the LassySmall figures are significantly better than the ones for the Frisian corpus.

[12] See also `https://cran.r-project.org/web/packages/udpipe/vignettes/udpipe-annotation.html`.

[13] See Sections 3.3, 3.4 and 3.5 at `https://ufal.mff.cuni.cz/udpipe/1/users-manual`.

[14] For the settings see `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-3131` where the parameters can be found in the folder 'models-ud-2.5' in 'udpipe-ud-2.5-191206-reproducible_training.zip'.

[15] Rand Wilcox's function `medmcp` from the Rallfun-v38 package was used. This function compares medians using a percentile bootstrap method. This function is very robust, even for small samples with a non-normal distribution (Wilcox, in press).

[16] See `https://fasttext.cc/docs/en/crawl-vectors.html`.

This indicates that further extension of the corpus is desirable. However, exceptions are ‘f1 word’, ‘f1 sents’, and ‘lemma’ (both for ‘Raw text’ and ‘Gold tok’).

4. Tools and results

The Frisian lemmatizer/PoS tagger/dependency parser is released as a web app for human end users[17], as well as a web service for software to interact with[18]. The source code is available at Bitbucket.[19] The user can choose from different output formats: a tab-separated file, an Excel file or a file in CoNLL-U format.

The online app also provides a visualization tool which enables the user to visualize the frequencies of PoS tags in the submitted text, the frequencies of the most frequently occurring tokens per PoS tag, frequencies of keyword word sequences, co-occurrence networks and word clouds. The graphs are made by using several R packages.[20] An example is shown in Figure 1. The graph is obtained on the basis of the Wikipedia text ’Ingelsk’ (English)[21] and shows the frequencies of the PoS tags in the text. On the basis of the same text Figure 2 was made. This figure visualizes frequencies of noun lemmas. The most frequent lemmas are *taal*, ‘language’, *dialect* ‘dialect’ and *útspraak* ‘pronunciation’.

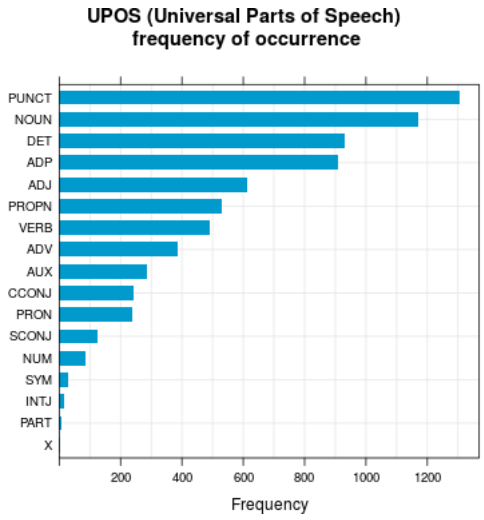

Figure 1: Frequencies of PoS tags in the Frisian Wikipedia text ‘Ingelsk’.

5. Conclusions and future prospects

In this paper we presented a lemmatizer/PoS tagger/dependency parser for West Frisian. To this end we built a West Frisian corpus as training material. Much time was saved by not building the corpus from scratch, but rather by taking a route via a larger and related language, in this case Dutch.

A Dutch tagger/annotator was applied to a Dutch word-by-word translation of the Frisian text. As for obtaining PoS tags, we investigated several alternatives such as aligning the Frisian text to a Dutch parallel text, or translating the Frisian text into Dutch by using Google Translate. However, the better PoS tags were obtained on the basis of a Dutch word-by-word translation that was created by using the previous version of the Frisian/Dutch - Dutch/Frisian translation program *Oersetter*. Manual correction of the PoS tags was still necessary.

Morphological and syntactical annotations were also preferably generated on the basis of an *Oersetter* translation, but the translations needed to be carefully checked and corrected before submitting them to a Dutch annotator.

The current corpus consists of 44,714 words. We plan to make this corpus available as part of the Universal Dependencies corpus.[22]

When training a lemmatizer/PoS tagger/dependency parser on the basis of this corpus, reasonable accuracies were obtained.

The performance of the lemmatizer/tagger/annotator when it was trained using default parameters was compared to the performance that was obtained when using the parameter values that were used for training the LassySmall UD 2.5 corpus. A significant improvement was found for ‘lemma’. Once the corpus is complete, the parameters will be tuned to achieve optimum performance.

We compared the performance obtained on the basis of

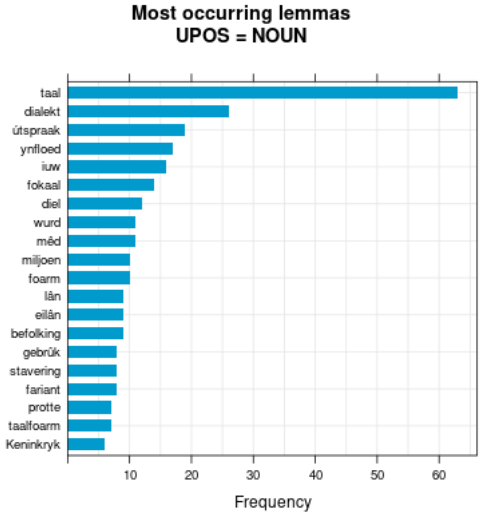

Figure 2: Frequences of noun lemmas in the Frisian Wikipedia text ‘Ingelsk’.

[17] See https://frisian.eu/udpipeapp/.

[18] See https://frisian.eu/udpipeservice/.

[19] Online at https://bitbucket.org/fryske-akademy/udpipe/src/master/.

[20] The following packages are used: lattice, udpipe, textrank, igraph, ggraph, ggplot2, ggwordcloud.

[21] The text was retrieved on June 3rd, 2021.

[22] See https://universaldependencies.org.

the latter parameters with the performance of a lemmatizer/tagger/annotator that was trained on the basis of the LassySmall UD 2.8 corpus using the same parameters. The LassySmall UD 2.8 corpus is much larger than the current Frisian corpus and the performance when using this corpus was significantly better for most metrics compared to the Frisian corpus. Therefore, we aim to extend the Frisian corpus up to 100,000 words. Not only texts of the genres that are currently in the corpus will be added, but also legal texts, minutes, and texts from manuals. When extending the corpus, we will strive for the different genres to be represented in a balanced way in the corpus. Furthermore, we noticed that many mistakes are made when processing text that contains hollandisms and non-standard Frisian variants. This will also be taken into account when extending the corpus.

The tools that we developed will be of great help for studying the Frisian language on the basis of (large) corpora. One can think of named entity resolution, coreference resolution, sentiment analysis, question answering, textometry, authorship discrimination, language detection, and measuring the influence of Dutch on Frisian.

In addition to the output formats currently provided, the XML/TEI format will be added. This will make the output also searchable for search engines like BlackLab (De Does et al., 2017) which is used in the publicly available search system of Frisian corpora.[23]

We also plan to develop a Frisian monolingual BERT model, similar as BERTje was developed as a monolingual Dutch model (De Vries et al., 2019). For the LassySmall dataset, the developers of BERTje found a PoS tagging accuracy score of 96.6%, thus outperforming the 95.98% accuracy score achieved by UDPipe 2.0. Therefore, we would like to investigate whether a similar improvement would be achieved by training and using a Frisian monolingual BERT model.

6. Acknowledgements

This research was made possible by a CLARIAH-Plus project financed by the Dutch Research Council (Grant 184.034.023). We thank two anonymous reviewers for their valuable comments.

Appendix

The tables below show the performance of the lemmatizer, tagger and parser, measured on the testing portion of the data, evaluated in three different settings. The metrics are briefly explained in Section 3.1.

[23]See `https://frisian.eu/frisian-corpora`, where currently mainly Middle Frisian and minimally annotated modern Frisian texts can be searched.

	Raw text		Gold tok		Gold tok + mor	
	mean	sd	mean	sd	mean	sd
f1 words	100.0	0.01				
f1 sents	91.4	1.88				
UPOS	94.4	0.48	94.4	0.47		
XPOS	87.8	0.56	87.8	0.56		
UFeats	89.6	0.43	89.6	0.42		
Lemma	94.9	0.48	94.9	0.48		
UAS	72.3	1.32	72.7	1.26	78.4	1.18
LAS	66.3	1.35	66.6	1.32	73.9	1.36

Table 4: The performance using the Frisian corpus. When training the default UDPipe hyperparameter settings were used.

	Raw text		Gold tok		Gold tok + mor	
	mean	sd	mean	sd	mean	sd
f1 words	100.0	0.01				
f1 sents	89.7	2.41				
UPOS	94.6	0.28	94.6	0.30		
XPOS	88.1	0.50	88.1	0.52		
UFeats	89.8	0.47	89.8	0.50		
Lemma	96.0	0.25	96.0	0.25		
UAS	72.5	1.15	73.1	1.12	78.5	0.87
LAS	66.4	1.08	67.0	1.13	73.9	1.21

Table 5: The performance using the Frisian corpus. The hyperparameter settings for training on the basis of the LassySmall UD 2.5 corpus were used.

	Raw text		Gold tok		Gold tok + mor	
	mean	sd	mean	sd	mean	sd
f1 words	99.9	0.06				
f1 sents	93.1	2.32				
UPOS	94.5	0.41	94.5	0.38		
XPOS	88.0	0.51	88.1	0.48		
UFeats	89.8	0.47	89.9	0.45		
Lemma	96.1	0.34	96.1	0.29		
UAS	72.5	1.48	73.0	1.36	78.2	1.33
LAS	66.4	1.74	66.9	1.62	73.8	1.40

Table 6: The performance using the Frisian corpus. The hyperparameter settings for training on the basis of the LassySmall UD 2.5 corpus were used. Pre-trained word vectors that were provided by fastText were used.

	Raw text		Gold tok		Gold tok + mor	
	mean	sd	mean	sd	mean	sd
f1 words	99.9	0.03				
f1 sents	81.3	1.31				
UPOS	95.6	0.30	95.9	0.30		
XPOS	93.7	0.38	94.1	0.36		
UFeats	95.1	0.26	95.6	0.26		
Lemma	94.2	0.25	94.4	0.25		
UAS	81.3	0.74	83.4	0.71	87.1	0.59
LAS	77.5	0.73	79.4	0.65	84.0	0.66

Table 7: The performance using the Dutch LassySmall UD 2.8 corpus. The hyperparameter settings for training on the basis of the LassySmall UD 2.5 corpora were used.

7. Bibliographical References

De Does, J., Niestadt, J., and Depuydt, K. (2017). Creating research environments with BlackLab. In Odijk J. et al., editors, *CLARIN in the Low Countries*, pages 245–257. Ubiquity Press, London.

De Vries, W., Bartelds, M., Nissim, M., and Wieling, M. (2021). Adapting monolingual models: Data can be scarce when language similarity is high. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4901–4907, Online, August. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Dijkstra, J., Heeringa, W., Yilmaz, E., van den Heuvel, H., van Leeuwen, D., and Van de Velde, H. (2017). A real time study of contact-induced language change in frisian relative pronouns. In *Proceedings of the International Symposium on Monolingual and Bilingual Speech*, pages 113–119.

Dyer, C., Chahuneau, V., and Smith, N. A. (2013). A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Hupkes, D. and Bod, R. (2016). POS-tagging of historical Dutch. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 77–82.

Klinkenberg, E., Jonkman, R., and Stefan, N. (2018). *Taal yn Fryslân. De folgjende generaasje [Language in Fryslân. The next generation]*. Morgan Kaufman Publishers, Ljouwert.

Lauscher, A., Ravishankar, V., Vulić, I., and Glavaš, G. (2020). From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online, November. Association for Computational Linguistics.

Littell, P., Mortensen, D. R., Lin, K., Kairis, K., Turner, C., and Levin, L. (2017). Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14.

Edmondson, M., (2020). *googleLanguageR: Call Google's 'Natural Language' API, 'Cloud Translation' API, 'Cloud Speech' API and 'Cloud Text-to-Speech' API*. R package version 0.3.0.

Scherrer, Y. (2014). Unsupervised adaptation of supervised part-of-speech taggers for closely related languages. In *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects (VarDial)*, pages 30–38. Association for Computational Linguistics and Dublin City University.

Schuster, M., Johnson, M., and Thorat, N. (2016). Zero-shot translation with Google's multilingual neural machine translation system. *Google AI Blog*, 22.

Straka, M. and Straková, J. (2017). Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Straka, M., Hajic, J., and Straková, J. (2016). UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297.

Tjong Kim Sang, E. (2016). Improving part-of-speech tagging of historical text by first translating to modern text. In *International Workshop on Computational History and Data-Driven Humanities*, pages 54–64. Springer.

Turovsky, B. (2016). Found in translation: More accurate, fluent sentences in Google Translate. *Blog. Google. November*, 15.

Üstün, A., Bisazza, A., Bouma, G., and van Noord, G. (2020). Udapter: Language adaptation for truly universal dependency parsing. *arXiv preprint arXiv:2004.14327*.

Wijffels, J., (2020). *udpipe: Tokenization, Parts of Speech Tagging, Lemmatization and Dependency Parsing with the 'UDPipe' 'NLP' Toolkit*. R package version 0.8.4-1.

Wilcox, R. R. (in press). *Introduction to Robust Estimation and Hypothesis Testing*. Academic press, San Diego, CA, 5th edition.

Yarowsky, D., Ngai, G., , and Wicentowski, R. (2001). Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the 1st International Conference on Human Language Technology Research (HLT)*, pages 161–168.

8. Language Resource References

Bouma, G. and Van Noord, G. (2017). Increasing return on annotation investment: the automatic construction of a Universal Dependency treebank for Dutch. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 19–26.

De Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., and Nissim, M. (2019).

BERTje: A Dutch BERT model. *arXiv preprint arXiv:1912.09582*.

Instituut voor de Nederlandse taal. (2022). Geïntegreerde Taal-Bank (GTB). Retrieved 4 January 2022.

Koehn, P. and Hoang, H. (2009). *Statistical Machine Translation System - User Manual and Code Guide*. University of Edinburgh.

Van Gompel, M., Van den Bosch, A., and Dijkstra, A. (2014). Oersetter: Frisian-Dutch statistical machine translation. In P. Boersma, et al., editors, *Philologia Frisica anno 2012*, pages 287–296. Fryske Akademy, Ljouwert.

Van Noord, G., Bouma, G., Van Eynde, F., de Kok, D., van der Linde, J., Schuurman, I., Sang, E. T. K., and Vandeghinste, V. (2013). Large scale syntactic annotation of written Dutch: Lassy. In Peter Spyns et al., editors, *Essential Speech and Language Technology for Dutch: Results by the STEVIN programme*, pages 147–164. Springer Berlin Heidelberg, Berlin, Heidelberg.

Van Noord, G. (2006). At last parsing is now operational. In Piet Mertens, et al., editors, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*, pages 20–42.