Approximating a Zulu GF concrete syntax with a neural network for natural language understanding

Laurette Marais
Voice Computing, NGEI, CSIR
laurette.p@gmail.com

Abstract

Multilingual Grammatical Framework (GF) domain grammars have been used in a variety of different applications, including question answering, where concrete syntaxes for parsing questions and generating answers are typically required for each supported language. In low-resourced settings, grammar engineering skills, appropriate knowledge of the use of supported languages in a domain, and appropriate domain data are scarce. This presents a challenge for developing domain specific concrete syntaxes for a GF application grammar, on the one hand, while on the other hand, machine learning techniques for performing question-answering are hampered by a lack of sufficient data. This paper presents a method for overcoming the two challenges of scarce or costly grammar engineering skills and lack of data for machine learning. A Zulu resource grammar is leveraged to create sufficient data to train a neural network that approximates a Zulu concrete syntax for parsing questions in a proof-of-concept question-answering system.

1 Introduction

In any Grammatical Framework (GF) domain grammar, typically called an application grammar, the abstract syntax defines the domain, in the sense that it provides the concepts of the domain and the ways in which they can be combined to construct meanings. The semantics is modelled, and hence restricted, by the abstract syntax, which leads each concrete syntax to be a controlled natural language (CNL) of a specific natural language, due to its limitation of semantics, syntax and vocabulary (Kuhn, 2014). The GF runtime enables both parsing of text strings into abstract trees, which is a kind of language understanding, and linearisation of abstract trees into text strings, which is a kind of language generation (Ranta et al., 2020).

GF-based CNLs have proven useful for developing language technology applications for low-resourced languages.[1] For example, Marais et al. (2020) presents a multilingual CNL used for speech-to-speech translation to enable healthcare providers to communicate with patients in their own language. Coverage of the application is restricted via a CNL in order to achieve high-quality speech translation in a high risk setting. In Marginean (2017), a GF domain grammar is used to perform question-answering (QA) in the medical domain.

The effort required to develop application grammars is reduced if a GF resource grammar (RG) exists for a language, since it can be used as a software library for the morphology and syntax of the language. Using an RG to develop an application grammar essentially involves mapping the semantic categories and functions in the application grammar to the syntactic categories and functions in the RG.

In this paper, we show how such a mapping can, in a sense, be learned by a neural network so that a workflow that relies heavily on grammar engineering skills can be replaced with one that requires machine learning skills. The two approaches are compared by considering the suitability of the resulting artifacts for the use case.

In Section 2 we present the application context for this work, namely a QA system where the concrete syntax we attempt to approximate is responsible for enabling language understanding. Then, in Section 3 we describe what a typical workflow for developing a Zulu concrete syntax would look like in an under-resourced development context. Section 4 presents the technique whereby the Zulu RG is leveraged alongside a domain abstract syntax to generate a dataset for training a neural network to

[1] For a recent audit of human language technology resources for Zulu, see Moors et al. (2018)

approximate a Zulu concrete syntax. A description of the neural networks trained and a comparison of them is given in Section 5. We discuss and contextualise the results in Section 6, before concluding with closing remarks in Section 7.

2 Overview of a grammar-based spoken QA system

The grammar-based system that provides the context for this work is a proof-of-concept spoken QA system that answers questions related to weather conditions for various locations in South Africa. The most important semantic concepts are place names, weather conditions, weather elements and time. The `WeatherFact` abstract syntax is centered around the notion of a weather fact, and for each supported language, two concrete syntaxes are included for expressing questions about weather facts and answers about weather facts, respectively. The OpenWeatherMap One Call API[2] is used to acquire up-to-date information with which to answer questions. It provides current weather information, historic weather data for 5 days, minute information for the next hour, hourly information for 48 hours and daily information for 7 days for any geographical coordinates. The semantics of this domain is by nature constrained, making a semantically limited CNL an appropriate choice for presenting a language interface to it.

The `WeatherFact` grammar allows questions about the weather for a fixed list of 33 locations in South Africa, including the biggest cities, provincial capitals, and cities and towns with airports. Furthermore, users can ask about the general weather conditions, or about specific weather metrics, such as cloud coverage or temperature, and time can be indicated by referring to a number (between 1 and 99) of minutes, hours or days in the past or future, or the concepts of yesterday and tomorrow. Typical English questions included in the grammar are 'What is the weather like in Johannesburg?' and 'What will the wind speed be in Cape Town in two days?'.

Figure 4 shows the basic architecture of the text-based part of the QA system. The `WeatherFact` multilingual GF grammar is responsible for parsing questions into semantic trees and linearising semantic trees into answers. Any given semantic tree can be linearised as a question or an answer, but parsing a question will result in an incomplete

[2]https://openweathermap.org/api/one-call-api

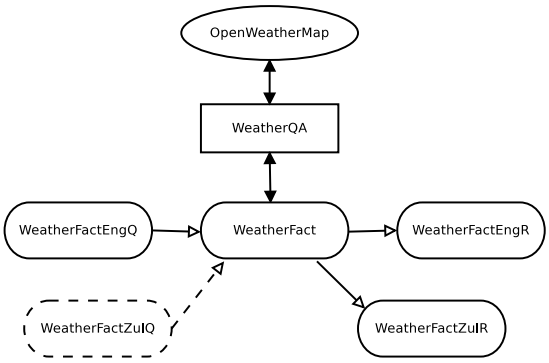

Figure 1: QA architecture

semantic tree – a missing sub-tree must be created and inserted by the QA system, so that the complete tree can be linearised as an answer. Figure 2 shows an example of a semantic tree, with the sub-tree that would be missing if it was created by parsing a question using the `WeatherFactEngQ` concrete syntax indicated in bold.

The incomplete semantic tree contains all the information necessary to inform the QA system of which query to send to the weather service, as well as how to extract the relevant information from the result in order to complete the tree. The root node informs the QA system what kind of information to expect in the tree. The OpenWeatherMap One Call API accepts geographical coordinates and returns a structured description of the current, past and future weather conditions as mentioned above. For example, in the case shown here, the QA system must use the `Location` information (in the form of a place name) in the tree to look up relevant geographical coordinates, which is used to send a query. The `TimeInstant` and `WeatherElement` information is then used to extract the appropriate information from the query result in order to supply the missing sub-tree.

3 Developing a question-parsing concrete syntax

For language generation in this context, it is entirely acceptable to provide a single way of expressing a certain meaning. However, a spoken QA system should allow some variation in how meaning is expressed by the user. Training data-driven models for this purpose typically requires large datasets of question-answer pairs (Bao et al., 2016), which are not available for many under-resourced languages, such as Zulu. Instead, a grammar-based language understanding system, developed for the relevant

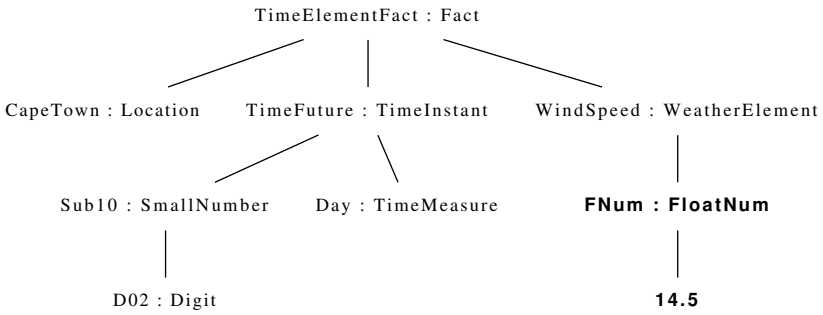

Figure 2: Semantic tree corresponding to the question 'What will the wind speed be in Cape Town in two days?' and the answer 'In two days, the wind speed in Cape Town will be 14.5 kilometers an hour.'

domain, remains a feasible approach to making spoken QA systems available for under-resourced languages. It then falls to the grammar engineer to produce a concrete syntax with an acceptable amount of flexibility for user input.

For Zulu, this is not usually as simple as bootstrapping from an existing concrete syntax for, for example, English, since the two languages are very dissimilar in terms of their linguistic characteristics and lexicon. For example, consider the English utterances 'Is it hot in Johannesburg?' and 'Is it windy in Johannesburg?', which concern the `WeatherElement` concepts of `Temperature` and `WindSpeed`, respectively. These utterances were translated by a Zulu language practitioner in order to develop an idiomatic Zulu concrete syntax, and the translations were given as *Kuyashisa eGoli?* and *Kunomoya eGoli?*. At first glance, the difference between the two Zulu utterances may appear to mirror that of the English utterances, but this is not the case.

Kuyashisa contains the verb root *-shis-*, which means to be hot, and it is used here in the present tense with the subject concord for noun class 17 (a locative class), to mean 'it is hot'. *Kunomoya*, on the other hand, contains the noun root *-moya*, meaning 'wind', and it is used in the present tense associative copulative construction, along with the subject concord for noun class 17, to mean 'it is with wind'. A concrete syntax developer would have to know that the two concepts are expressed using different syntactic constructions in Zulu, so that the Zulu linearisation category of `WeatherElement` in the application grammar reflects this variability. Ideally, the concrete syntax developer needs a tool to analyse the translated utterances to detect the syntactic and morphological characteristics in the domain data.

A Zulu resource grammar (RG) is currently under development[3], and in conjunction with the GF runtime and an appropriate lexicon, it provides the ability to parse Zulu utterances in order to inspect their linguistic structure. A typical workflow for developing a domain specific concrete syntax using the Zulu RG is as follows:

1. Develop the abstract syntax and an English concrete syntax.
2. Identify a representative subset of utterances from the grammar and render them in English.
3. Obtain a set of translations for the English utterances.
4. Analyse the syntactic constructions used in the Zulu translations (with the help of the RG), and extrapolate from them in order to implement the Zulu concrete syntax.
5. Generate Zulu utterances from the concrete syntax for review.

The resulting concrete syntax models the linguistic structures found in the translated domain data. The last step in the workflow requires specialised grammar engineering skills. In South Africa, where such skills are not taught at tertiary level, this workflow cannot be implemented widely. On the other hand, machine learning is widely taught and a growing community of natural language processing researchers in Africa (Orife et al., 2020) could exploit workflows that rely on data-driven techniques.

4 Data for domain specific language understanding

The obvious requirement for developing any data-driven solution is sufficient and appropriate data.

[3]https://github.com/LauretteM/gf-rgl-zul

Many efforts have been made and are currently underway to gather corpora for the under-resourced languages of Africa and South Africa (Barnard et al., 2014; Moors et al., 2018). In this work, we present a technique for *generating* a domain specific corpus. The use case admits of a semantically restricted CNL – in essence, we intend to develop a model for a Zulu CNL which is, in addition, implicitly restricted linguistically, rather than explicitly as with a GF concrete syntax.

4.1 What to learn?

A concrete syntax enables the GF parser to accept strings and to produce typed semantic tree structures, but this need not be the only options to consider as input and output for a neural network.

When considering Zulu text input, a first option is to use space separated tokens, but this presents some problems for machine learning. Zulu is an agglutinating language with a conjunctive orthography, resulting in tokens that often consist of long sequences of morphemes and which typically leads to sparsity in datasets. Alternation rules govern sound changes between adjacent morphemes, which makes the task of segmenting morpheme sequences non-trivial. However, Kotzé and Wolff (2015) have shown that segmenting on syllable boundaries, which are easily identified, can improve the performance of natural language translation systems. Furthermore, one could also consider character-level segmentation (Lee et al., 2017).

The Zulu RG provides additional options for input to a neural network, although it introduces a pre-processing step that may be subject to failure. Specifically, Zulu strings could be parsed by the GF runtime using the Zulu RG to produce syntax trees. Syntax trees, however, are hierarchical, and it may not be necessary to retain this structure in order to benefit from the pre-processing step. In fact, transformer sequence-to-sequence models use multi-head attention to learn the most relevant relationships between tokens in a sequence (Vaswani et al., 2017). In order to exploit existing transformer model implementations, the trees could be flattened into sequences of syntax function names. Additionally, the syntax trees could be mined for relevant information directly by, for example, extracting the lexical nodes to produce a lemma sequence.

A similar flattening might also be useful for the target of the neural network. Developing a concrete syntax that parses semantic function name sequences into semantic trees is trivial: each linearisation category is defined as a string, and each linearisation function is implemented to produce its own name followed by the strings of its children. The GF parser can then be exploited to restore the typed hierarchical structure.[4] Simpler sequences could be implemented by letting non-terminal functions produce an empty string followed by the strings contributed by its children. Such a concrete syntax essentially defines a natural language agnostic keyword question language that could be used by the QA system in exactly the same way as a concrete syntax for natural language questions.

4.2 What to learn from?

In the workflow discussed in Section 3, the representative subset of utterances from the domain grammar that is rendered in English in order to be translated, is chosen to be minimal and repetitive. For each semantic category, a template utterance is identified and the chosen category varied, so that its effect on the utterance can be seen. This kind of repetition is also a useful way to discourage a translator from introducing spurious variability in their translations that make it difficult to isolate the effects of the various semantic functions on the Zulu utterances. Variability, if required, is more effectively elicited by procuring utterances from different translators and by including variability in the source language.

For the `WeatherFact` grammar, 76 questions were generated via the English concrete syntax, which were translated by two language practitioners, resulting in 152 Zulu utterances. The English utterances included variations of utterances where it seemed natural in the English, such as 'How hot is it in Johannesburg?' for 'What is the temperature in Johannesburg?'. Although, Zulu numerals "are not a coherent morphosyntactic class" (Zerbian and Krifka, 2008), their behaviour is not domain specific information that requires elicitation from a translator, and hence the semantic functions for numbers were not varied in the elicitation data in the same way as other semantic functions.

The translations were parsed using the Zulu RG and a domain specific lexicon. Since the Zulu RG has not been completed yet, for the purpose of this experiment, it was extended, especially with re-

[4] Of course, this would only be possible if the function name sequence is defined by the concrete syntax. We discuss this caveat in Section 5.

gards to question words and their accompanying syntactic functions, in order to be able to achieve parses for 148 of the 152 translations, with at least one parsed Zulu utterance for each English utterance. The result is a set of 148 semantic-syntactic tree pairs.

4.2.1 Augmenting data within a CNL

The constraints on a domain of utterances imposed by the definition of an abstract syntax presents an opportunity to perform data augmentation that is likely to be semantically reliable. Augmentation can be done based on semantic categories, while the Zulu RG can be leveraged to produce grammatically correct Zulu utterances. If it is known that a certain change from one semantic tree to another is accompanied by a certain change from the corresponding syntax tree to another, this can be used to derive so-called augmentation rules. A rule would have the following form:

$T_A, T_B \rightarrow t_a, t_b$: Given a semantic tree T_1 and corresponding syntax tree t_1, if a semantic tree T_2 is acquired by substituting T_A in T_1 for T_B, and a syntax tree t_2 is acquired by substituting t_a in t_1 for t_b, then t_2 is the corresponding syntax tree of T_2.

A simple example from the `WeatherFact` domain is the following,

$$\{\text{Hour,Minute}\} \rightarrow \{\text{hora\_5\_6\_N,zuzu\_3\_4\_N}\}$$

which essentially states that whenever the noun stem *-hora* is used to express the meaning of `Hour`, the noun stem *-zuzu* can be used to express the meaning of `Minute` instead. With this rule, for example, the Zulu sentence *Bekushisa eMbombela emahoreni amathathu adlule?* ('How hot was it in Mbombela three hours ago?') gives rise to a new sentence, namely *Bekushisa eMbombela emizuzwini emithathu edlule?* ('How hot was it in Mbombela three minutes ago?'). Note the effect that the change in the class of the noun has on the modifying adjective *-thathu* ('three'), namely that *amathathu* becomes *emithathu*, as well as the relative clause based on the verb *-dlule* ('passed'), where *adlule* becomes *edlule*.

Rules are not limited with regards to the complexity of the sub-trees they contain. Figure 3 shows the rule which states that whenever the adverb *izolo*, which expresses `Yesterday`, is used, an adverbial phrase, which is linearised as *emahoreni amabili adlule*, can be used to express the notion of two hours ago (or `TimePast (Sub10 D02) Hour`).

A set of augmentation rules were developed with reference to the semantic-syntactic tree pairs. This was done by hand, but in principle, rules could also be derived from the semantic-syntactic tree pairs automatically, provided that the seed data is sufficiently representative. For example, we did not attempt to elicit examples of all numerals in the seed data in order to derive appropriate augmentation rules for numerals 1 to 99. Instead, these rules were defined by consulting linguistic texts, and could easily be reused for different domains. The augmentation rules were exhaustively applied to the seed data, which resulted in 341 254 unique semantic-syntactic tree pairs.

Next, the Zulu RG was used to linearise each syntax tree in order to obtain a Zulu utterance. Then, both the semantic and syntax trees were flattened and simplified to sequences of keywords and abstract lexical functions (effectively lemmas), respectively. The result was a dataset of 5-tuples (as shown in Table 1) that could be used to train sequence-to-sequence neural networks.

4.2.2 Review of augmentation algorithm

The semantic trees generated by the augmentation algorithm were compared to all those defined by the `WeatherFact` abstract syntax to determine if any semantic trees for questions were missing. The only trees not generated were those that contain a digit sequence starting with a zero. Given the semantics of the domain, this meant that all meaningful numbers (and therefore semantic trees) were generated.

A random sample of 100 5-tuples from the augmented data was selected and the Zulu linearisations, alongside their English equivalents, were presented to a language practitioner for review. The only errors discovered in the augmented data were the incorrect use of the immediate future and past tenses where the remote future and past tenses were required. This was deemed not to be a problem for this application, given the One Call API's time frame, for which the use of the remote tenses is unlikely. If it had presented a problem, the solution would have been to procure more translations that contain the remote tenses in Zulu, and to refine the augmentation rules accordingly.

{Yesterday,
TimePast (Sub10 D02) Hour} → {izolo_Adv,LocNPAdv (DetCN (DetQuant IndefArt NumPl)
(RelCN (AdjCN (PositA bili_A) (UseN hora_5_6_N))
(UseRCl TPerfTemp PPos (RelVP IdRP (UseV dlul_V)))))}

Figure 3: Example of an augmentation rule

Semantic tree	TimeElementFact Mbombela (TimePast (Sub10 D03) Minute) (Temperature ?)
Keywords	Mbombela TimePast D03 Minute Temperature
Syntax tree	PhrUtt NoPConj (UttS (UseCl TPerfPresTemp PPos (PredVP (UsePron (ProDrop it15_Pron)) (AdvVP (UseV shis_V) (LocNPAdv (AdvNP (DetCN (DetQuant IndefArt NumSg) (UseN Mbombela)) (LocNPAdv (DetCN (DetQuant IndefArt NumPl) (RelCN (AdjCN (PositA thathu_A) (UseN zuzu_3_4_N)) (UseRCl TPerfTemp PPos (RelVP IdRP (UseV dlul_V)))))))))))) NoVoc
Lemmas	it15_Pron shis_V Mbombela thathu_A zuzu_3_4_N dlul_V
Linearisation	bekushisa eMbombela emizuzwini emithathu edlule

Table 1: Example of a 5-tuple data point

4.2.3 Balancing the data

The presence of especially numbers in the grammar requires that the dataset be balanced to some extent. Since some kinds of utterances can contain any of the numbers below 100, these utterances far outnumber other kinds. The kind of utterances in this domain happens to correlate well with the length of the keyword sequence it is represented by, and so balancing was done on the length of the target keyword sequences by duplicating shorter utterances in the dataset. The final dataset contains 954 324 entries, and includes at least one example of every semantic question tree defined by the domain abstract syntax.

Usually, such duplication is not done when training machine learning systems, but in this case, we are aiming to approximate a concrete syntax. We will have achieved our goal if exactly those utterances which would have been modelled by a concrete grammar are handled correctly by the model, that is, those utterances that are in our implicit Zulu CNL. In effect, we are introducing drastic sample bias because our use case allows it.

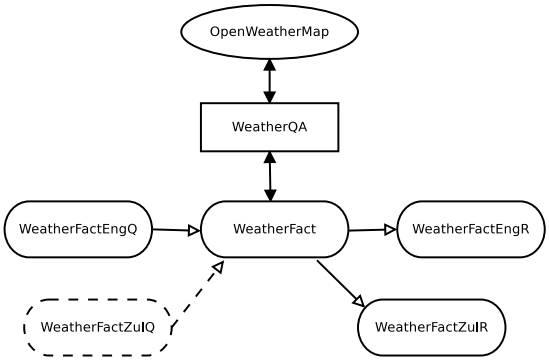

Figure 4: QA architecture

Model	F score	P score
token2key	1.00000	100%
syllable2key	1.00000	100%
char2key	0.99537	97.84%
syntax2key	1.00000	100%
lemma2key	0.99966	99.67%

Table 2: Comparison of different transformer models on a test set

5 Sequence-to-sequence models for language understanding

The PyTorch[5] implementation of a transformer sequence-to-sequence model of Vaswani et al. (2017) was used as the basis for a number of neural networks with varying input types, namely Zulu text strings, syntax function sequences and lemma sequences. For input based on the Zulu text strings, tokenisation was done on white space, syllable boundaries, and characters. Keyword sequences were used as the target. In total, five different models were trained on the augmented data, which was split into training, validation and test sets. Training was done over 3 epochs, which turned out to be sufficient for the models to converge. Table 2 lists the results obtained.

[5]https://pytorch.org

F score was calculated using the NLTK[6] implementation of ChrF score (with the default parameter values) on each respective test set. We have included the percentage of perfect predictions, which we have called the P score, because this is an indication of the number of times the QA system will be able to provide the user with exactly the information that was asked for, with no need to engage the user further.

The perfect or near perfect F for all the models show that they all succeed in learning to translate the controlled Zulu language to keyword sequences. Furthermore, the perfect or near perfect P scores mean that almost all output from the models can be parsed successfully using the keyword sequence concrete syntax. As such, the models could be used, in conjunction with a keyword concrete syntax, to approximate a Zulu concrete syntax for language understanding.

5.1 Bias and generalisation

We have knowingly introduced sample bias into our models by training exclusively on synthetic data. The next step would be to try to determine if any of the models generalise beyond the implicit Zulu CNL represented by the training data to a more semantically constrained CNL. In other words, can the model(s) accurately understand independently sourced Zulu utterances that express meanings defined in the abstract syntax?

To investigate this, the original set of English seed utterances were translated by two new Zulu translators. The new translations only contain a 13% overlap with the original translations, which seems to confirm the need for allowing variability in user input for a Zulu QA system. As before, the translations were parsed using the Zulu RG and in this case, we did not extend the grammar to parse the new translations. However, in order to compare the different models, we only used those new translations that could be parsed using the RG as is: we used 71 utterances, and discarded 81. As the RG becomes more complete, the former number should rise and the latter should drop. For now, we note that this step introduces some bias in the evaluation towards the syntactic structures currently covered by the RG.

The seed data was designed to be minimal and repetitive, as noted earlier, which makes it an unsuitable evaluation set. In order to achieve a more balanced evaluation set, we augmented the 71 parsed utterances (resulting in 138 896 utterances) and sampled it so that the final evaluation set contained 100 independently generated 5-tuples, balanced according to keyword sequence length. It should be noted that this augmentation step is only possible on utterances that can be parsed using the RG.

[6] http://www.nltk.org/

Model	F score	P score
token2key	0.65185	33%
syllable2key	0.83098	**54%**
char2key	0.81897	52%
syntax2key	0.81894	52%
lemma2key	**0.83629**	53%
lemma2key*	0.18702	0%

Table 3: Comparison of different transformer models on independently generated data

The bias introduced by limiting our evaluation to such utterances becomes clear when inspecting the evaluation set: although all the weather elements included in the domain grammar appear in the set of newly parsed utterances, this is only true for the present tense. The syntactic constructions used in the new translations to express the notion of *temperature* in the past and future tenses are the only ones currently covered by the RG. As a result, the `Temperature` keyword occurs in 52 of the 100 keyword sequences.

From Table 3, we can see that the *lemma2key* model performs the best in terms of F score, with *syllable2key*, *syntax2key* and *syntax2key* achieving comparable scores. In fact, the outlier is the *token2key* model trained on space separated tokens, which seems to confirm that the orthography of Zulu presents a problem for machine learning, and that, presumably, any attempt to segment the text systematically produces an improvement.

For reference, we have included the F and P scores for a *lemma2key** model trained only on the original seed data. Table 4 gives a comparison of some examples in the evaluation set for each of the two lemma-based models. The predictions have been post-processed to include only the tokens appearing before the first end-of-sequence symbol. It is evident that the model trained on the seed data has simply learned to produce sequences that mostly consist of the token 'Johannesburg', hence its P score of 0%.

Target	Augmented model prediction
NorthernCape Yesterday	NorthernCape Yesterday
EasternCape WindSpeed	EasternCape Clouds
Durban Yesterday Temperature	Durban Temperature
Limpopo TimePast D01 Hour	Limpopo TimePast D01 Hour
Target	**Seed model prediction**
NorthernCape Yesterday	Johannesburg Johannesburg
EasternCape WindSpeed	Johannesburg Temperature
Durban Yesterday Temperature	Johannesburg
Limpopo TimePast D01 Hour	Johannesburg Johannesburg Johannesburg

Table 4: Comparing the *lemma2key* model trained on the augmented data and the seed data

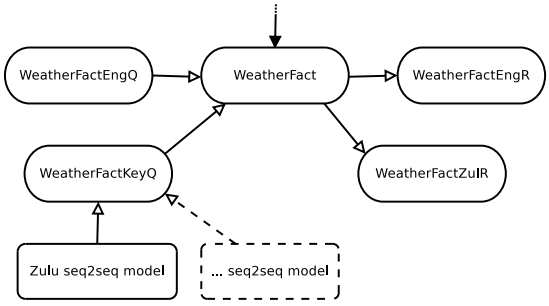

Figure 5: Grammar component of the QA architecture, adapted to include a Zulu sequence-to-sequence model for questions

6 Discussion

We have shown that the Zulu RG can be leveraged in conjunction with a domain abstract syntax to augment a very small set of manually translated data to a sufficiently large set of synthetic data, which essentially represents all the utterances a concrete syntax would be expected to cover. Using this dataset, a variety of different transformer models can be trained and incorporated into a pipeline that would perform almost exactly like the GF runtime enabled by a Zulu concrete syntax, with regards to parsing natural language into typed semantic tree structures. From this we conclude that it is possible to approximate a Zulu GF concrete syntax with a neural network to perform language understanding. Figure 5 shows the adapted configuration.

The language that is correctly "understood" by the models (i.e. the set of Zulu strings that lead to perfect predictions) has also been shown to be somewhat larger than what a concrete syntax might include. While it does not make sense to talk about the language accepted by a neural model, the model is integrated into the QA system in conjunction with a language agnostic keyword concrete syntax. Hence, those Zulu sentences that cause the neural model to produce strings that are in the keyword sequence language can be thought of as being "accepted". And while it is not possible to know exactly which Zulu sentences are accepted in this way, it is known which keyword sequences can be reconstructed into typed trees defined by the abstract syntax. In the long definition of a CNL, with regards to the C in CNL, Kuhn (2014) proposes that a CNL must be restricted in terms of "lexicon, syntax and/*or* semantics" (our emphasis) and also that it be "explicitly and consciously defined". It is certainly the case that the domain abstract syntax, as well as the keyword concrete syntax, is explicitly and consciously restricted in terms of semantics, while the input to the system is natural Zulu. Hence, we contend that the conjunction of a neural model and a keyword concrete syntax as described here does indeed implement a controlled natural language.

This work differs in some important ways from other text augmentation attempts. The goal of text augmentation is usually to *improve* machine learning by supplementing existing data (Duwairi R, 2021). The seed data is typically enough to train a reasonable system, but significant improvements can be made via augmentation. In this work, due to the absence of suitable data, seed data was generated via manual translation. This is an expensive way of obtaining data, and so the goal was to start with a minimal seed corpus, which we showed to be woefully insufficient to train a useful model.

Although the effective gain in data size for text augmentation techniques is not often reported, with authors instead reporting on improvements in system performance (Sharifirad et al., 2018; Kobayashi, 2018; Rizos et al., 2019; Şahin and Steedman, 2018), an increase of 5 times the original dataset is reported by Wang and Yang (2015),

while Duwairi R (2021) report a tenfold increase. This is in sharp contrast to the more than 2000-fold increase achieved here, from 148 to 341 254 unique utterances.

Furthermore, text augmentation has often focused on text classification tasks (Sharifirad et al., 2018; Kobayashi, 2018; Rizos et al., 2019), as opposed to sequence generating tasks, such as POS-tagging (Şahin and Steedman, 2018). Our work is similar to the latter in that a sequence of labels is generated, but, to the best of our knowledge, our work differs from any previous work with regards to the novelty of the target sequences generated by the augmentation process. In classification tasks, augmentation techniques have been aimed at producing more examples that preserve (or at most flip) the labels of the existing data (such techniques are "meaning preserving" (Rizos et al., 2019) in different senses, depending on the task), while the work of Şahin and Steedman (2018) either reduces or rearranges POS tags of existing target sequences (along with their corresponding source sequences) by cropping and rotating Universal Dependency trees.

Our augmentation technique, in contrast, leverages a domain grammar that models the semantics of a domain to produce entirely new target sequences. In addition, it leverages a Zulu resource grammar that models the linguistics of the natural language to produce corresponding source sequences. A useful connection has been explored between Universal Dependency trees and the GF RG library (Kolachina and Ranta, 2019), and in this sense also, our work is most similar to that of Şahin and Steedman (2018). However, the addition of a semantic domain abstract syntax has been the key to generating pairs of new source and target sequences for the task of language understanding. For example, substituting the notion of 'yesterday' with that of 'two hours ago', as per the rule in Figure 3, produces a new data point where the source sequence (via the syntax tree) *and* the target sequence (via the semantic tree) contain new tokens.

The domain abstract syntax increases the factor by which data can be augmented, while also imposing limitations on the structure of new data points that are generated. Future work will include experimenting with the complexity of domain grammars to better understand the ability of the augmentation technique to scale to larger and more complex domains, as well as to study the ability of neural networks to deal with increasingly complex constructs.[7]

7 Conclusion

The workflow described in this paper does not require any grammar engineering skills. Instead, it relies on the ability to use the Zulu RG and GF runtime to parse Zulu text and linearise syntax trees. This is a significant advantage in contexts where grammar engineering skills, especially in conjunction with knowledge of Zulu and its use in any given domain, is costly or scarce.

In addition to approximating a concrete syntax, we have shown that certain transformer sequence-to-sequence models, trained on synthetic augmented data, have some ability to generalise beyond the linguistic structures found in the seed data. This is an improvement on the use of a concrete syntax, especially since the ability to deal with variability in user input is important in spoken QA systems. An attempt was made to evaluate the extent of this kind of generalisation, although a more accurate assessment will only be possible once the Zulu RG is complete, since it forms the basis for generating a balanced evaluation set from seed data.

The neural networks developed in this work are unidirectional, as opposed to a concrete syntax which enables the GF runtime in both the parsing and linearising directions. Future work will include training neural networks in the opposite direction, namely to generate Zulu utterances from semantic trees. Since the language generation aspect of the QA system only requires one way of expressing meaning, it is expected that the technique presented here would achieve similar success.

The data augmentation step, which centres on a GF RG for the language, forms the core of the technique. In principle, therefore, it could be applied to any under-resourced language for which a GF RG exists.

Acknowledgements

The author would like to thank three anonymous reviewers for their insightful comments towards improving this manuscript.

[7] Thanks to an anonymous reviewer for this suggestion.

References

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514.

Etienne Barnard, Marelie H Davel, Charl van Heerden, Febe De Wet, and Jaco Badenhorst. 2014. The NCHLT speech corpus of the South African languages. Workshop Spoken Language Technologies for Under-resourced Languages (SLTU).

Abushaqra F. Duwairi R. 2021. Syntactic- and morphology-based text augmentation framework for arabic sentiment analysis. *PeerJ Computer Science*, 7.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Prasanth Kolachina and Aarne Ranta. 2019. Bootstrapping UD treebanks for delexicalized parsing. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 15–24, Turku, Finland. Linköping University Electronic Press.

Gideon Kotzé and Friedel Wolff. 2015. Syllabification and parameter optimisation in Zulu to English machine translation. *South African Computer Journal*, (57).

Tobias Kuhn. 2014. A Survey and Classification of Controlled Natural Languages. *Computational Linguistics*, 40(1):121–170.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully Character-Level Neural Machine Translation without Explicit Segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Laurette Marais, Johannes A. Louw, Jaco Badenhorst, Karen Calteaux, Ilana Wilken, Nina van Niekerk, and Glenn Stein. 2020. AwezaMed: A Multilingual, Multimodal Speech-To-Speech Translation Application for Maternal Health Care. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–8.

Anca Marginean. 2017. Question answering over biomedical linked data with grammatical framework. *Semantic Web*, 8(4):565–580.

Carmen Moors, Ilana Wilken, Karen Calteaux, and Tebogo Gumede. 2018. Human language technology audit 2018: Analysing the development trends in resource availability in all south african languages. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, SAICSIT '18, page 296–304, New York, NY, USA. Association for Computing Machinery.

Iroro Orife, Julia Kreutzer, Blessing Sibanda, Daniel Whitenack, Kathleen Siminyu, Laura Martinus, Jamiil Toure Ali, Jade Abbott, Vukosi Marivate, Salomon Kabongo, Musie Meressa, Espoir Murhabazi, Orevaoghene Ahia, Elan van Biljon, Arshath Ramkilowan, Adewale Akinfaderin, Alp Öktem, Wole Akin, Ghollah Kioko, Kevin Degila, Herman Kamper, Bonaventure Dossou, Chris Emezue, Kelechi Ogueji, and Abdallah Bashir. 2020. Masakhane – Machine Translation For Africa.

Aarne Ranta, Krasimir Angelov, Normunds Gruzitis, and Prasanth Kolachina. 2020. Abstract Syntax as Interlingua: Scaling Up the Grammatical Framework from Controlled Languages to Robust Pipelines. *Computational Linguistics*, 46(2):425–486.

Georgios Rizos, Konstantin Hemker, and Björn Schuller. 2019. Augment to prevent: Short-text data augmentation in deep learning for hate-speech classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 991–1000, New York, NY, USA. Association for Computing Machinery.

Gözde Gül Şahin and Mark Steedman. 2018. Data augmentation via dependency tree morphing for low-resource languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5004–5009, Brussels, Belgium. Association for Computational Linguistics.

Sima Sharifirad, Borna Jafarpour, and Stan Matwin. 2018. Boosting text classification performance on sexist tweets by text augmentation and text generation using a combination of knowledge graphs. In *Proceedings of the 2nd workshop on abusive language online (ALW2)*, pages 107–114.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR*, abs/1706.03762.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, Lisbon, Portugal. Association for Computational Linguistics.

Sabine Zerbian and Manfred Krifka. 2008. Quantification across bantu languages. *Quantification: A cross-linguistic perspective*, 64:383–414.