The role of morphosyntactic similarity in generating related sentences

Michael Hammond
Dept. of Linguistics
U. of Arizona
Tucson, Arizona 85721, USA
hammond@arizona.edu

Abstract

In this paper we describe our work on Task 2: AmericasNLP 2024 Shared Task on the Creation of Educational Materials for Indigenous Languages. We tried three approaches, but only the third approach yielded improvement over the baseline system. The first system was a fairly generic transformer model. The second system was our own implementation of the edit tree approach from the baseline system. Our final attempt was a version of the baseline system where if no transformation succeeded, we applied transformations from similar morphosyntactic relations. We describe all three here, but, in the end, we only submitted the third system.

1 Introduction

The nature of the task was to transform one sentence into another in three languages based on a specification of the morphosyntactic differences between the input and output sentences (Chiruzzo et al., 2024). The three languages are Bribri, Guarani, and Maya. We give sample data in Table 1.

Glosses or translations were not provided. In addition, we do not know what morphosyntactic features might be appropriate for the input sentence.

We were provided with definitions for the different tags. In the first example in Table 1 we convert the absolutive argument to a plural. In the second example, we convert to an affirmative. Finally, in the third case, we switch one of the arguments to a first person plural subject.

We tried three different approaches: a simple transformer, our own implementation of edit trees in terms of a single regular transduction, and using related morphosyntactic tags when possible. In the following sections, we describe these three attempts.

2 Transformer

The first model we tried was a simple transformer (Vaswani et al., 2017).[1] We first concatenate the input and morphological tags to serve as input.

These are fed into the encoder which first creates embeddings, applies drop-out, and feeds these to a GRU layer (Cho et al., 2014).

The output and hidden weights are then fed to an attention-based decoder with two layers of GRUs and a simple linear layer. Attention was Bahdanau (Bahdanau et al., 2014).

Batch size varied, but was typically around 32. The dimensions for all hidden layers was either 512 or 1024 for different runs. Dropout for the encoder was set at 0.1. Loss was negative log likelihood and the Adam optimizer was used. We tried a variety of different configurations, but best performance was at 600 epochs with 512 hidden nodes at all layers. See Table 2 for performance with dev data.

The data are extremely limited and this surely impaired performance. Our sense is that simply concatenating the input and morphosyntactic tags was also not the best choice.

3 Edit trees as a single transduction

The baseline system for the task is based on the notion of *edit trees*. The basic idea is to build a tree representation of changes that the input must undergo to be converted to the output (Chrupała, 2008).

Chrupała gives the edit tree in Figure 1 for the Polish word pair *najtrudniejszy* 'hardest' and *trudny* 'hard'. The basic logic is that we identify the largest shared span, in this case characters 3 through 6. To the left of that, we replace *naj* with

[1] All of our code can be obtained at https://github.com/hammondm/americasnlp24task2. Our specific transformer architecture is an adaptation from https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html.

Proceedings of the 4th Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP), pages 221–223

Language	Input	Features	Target
Bribri	Pûs kapë'wa̲	ABSNUM:PL	Pûs kapë'ulur
Guarani	Ore ndorombyai kuri	TYPE:AFF	Ore rombyai kuri
Maya	Janalnajen tu k'íiwikil koonol	PERSON:1_PL	Janalnajo'on tu k'íiwikil koonol

Table 1: Example data

Language	Accuracy	BLEU	ChrF
Bribri	0.00	3.14	12.51
Guarani	0.00	0.29	4.56
Maya	0.67	14.17	40.08

Table 2: Performance with transformer model on dev data

Language	Accuracy	BLEU	ChrF
Bribri	3.30	12.93	39.14
Guarani	0.00	22.19	72.63
Maya	1.34	30.68	71.95

Table 3: Performance with our implementation of edit trees on dev data

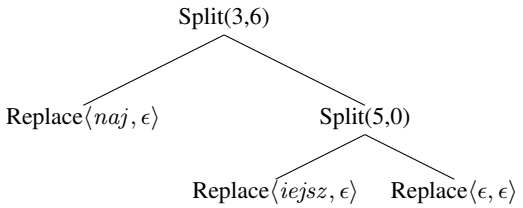

Figure 1: Example edit tree (Chrupała, 2008, p.127)

ϵ. To the right, we repeat the process and identify the longest shared span: *y*. To the left of this, we replace *iejsz* with ϵ. To the right, we do nothing: Replace(ϵ,ϵ).

Formally, Chrupała defines a function lcs from two strings $(\Sigma^* \times \Sigma^*)$, specifically $w_{1...n}$ and $w'_{1...m}$, to four natural numbers $(\mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N})$, (i, j, k, l), representing indices into the strings where the shared string is indexed as $w_{i...n-j} = w'_{k...m-l}$.

There is then a function *split* which maps from a string and indices to three strings $(\Sigma^* \times \mathbb{N} \times \mathbb{N}) \rightarrow (\Sigma^* \times \Sigma^* \times \Sigma^*)$, taking a string $w_{1...n}$ and indices i and j and returning the triple $(w_{1...i}, w_{i+1...n-j}, w_{n-j+1...n})$.

An edit tree is then either a Replace node with two strings or it is a Split node with two indices and two daughter nodes that are themselves edit trees.

An edit tree is built with the function et which is defined as follows with respect to strings w and w'. If there is no lcs span, then the tree is simply Replace(w, w').

Otherwise it is defined as:

$$\begin{array}{l} \text{Split}(i_w, j_w), \\ et(w_{prefix}, w'_{prefix}), \\ et(w_{suffix}, w'_{suffix}) \end{array} \quad (1)$$

In our implementation, these operations are mapped to a single regular transduction with backreferences. For convenience, we use python for this.

First, Split nodes are represented as a list of three elements: the left daughter, the two indices, and the right daughter. Replace nodes are represented as a pair of strings. The tree in Figure 1 would be represented as:

$$[(naj, \epsilon), (3, 6), [(iejsz, \epsilon), (5, 0), (\epsilon, \epsilon)]] \quad (2)$$

We traverse the tree from left to right converting each node into a pair of strings. All string pairs map to themselves. Pairs of indices i, j are translated into maps from `.{1,k}`, where $k = j - i$, to the next available backreference. In the case of the tree in Figure 1, we have the translation `naj.{1,3}iejsz.{1,1}` mapping to `\1\2`. This mapping is executed in our code using the python `re.sub` function.

This approach approximates the baseline system, but does not perform as well. See Table 3.

4 Morphosyntactic similarity

Our final model, and the one we submitted, was an addition to the baseline system.

The baseline system records the edit trees that are associated with specific morphosyntactic tag combinations along with the relative frequency of each tree.

At inference stage, one selects the edit trees associated with the tags for the test item, sorts these by how frequently they're used in training, and try them one by one starting from the most frequent. If a rule succeeds, the output of that rule is returned.

Language	Accuracy	BLEU	ChrF
Bribri	9.38	17.13	55.07
Guarani	25.81	50.36	79.46
Maya	14.84	22.55	73.18

Table 4: Performance with morphosyntactic tag similarity on test data

If no rule succeeds or the specific tag combination is not in the training data, then the input is returned as the output.

Our adaptation here was to add additional options to the list. If the procedure above produced no distinct output form, we then applied additional rules. These additional rules were generated from the full list of rules, sorted by how similar the tag sequence is to the test item tag sequence. If the above procedure results in no change, we then turn to these rules, going through them one by one. This procedure is terminated the same way as the baseline system: when a rule produces a change, that is the final output and further rules are not considered. If no rule produces a distinct output, the input itself is returned.

This change results in a modest overall improvement in the baseline as seen in Table 4.

5 Conclusion

To summarize, we tried three different techniques: transformer, edit trees as transductions, and exploiting morphosyntactic similarity in selecting edit trees.

As implemented, the transformer performed the worst. Systems built on edit operations seem to perform much better in these character-to-character mapping domains, so this is not really a surprise.

Translating edit trees into regular transductions did not reach the level of the baseline, but is not an unreasonable approach to pursue further. Edit operations are clearly useful. The question is what is the scope of those operations. Are they separate operations as in edit trees as originally developed or can some amalgam of those operations be more successful.

Finally, using morphosyntactic similarity is successful and this is thus clearly an approach worth pursuing further.

One very obvious way to go further is to build a model of the morphosyntactic structure of the input. We do not know what the words mean, but perhaps we can get some mileage toward identifying parts of speech from the meanings of the tags. With this in hand, we could exploit that part of speech information in our edit trees.

This last approach is purely speculative, but it seems like a fairly obvious way to go (in hindsight!) given the nature of the task.

Acknowledgments

Thanks to Diane Ohala for useful discussion. All errors are my own.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Luis Chiruzzo, Pavel Denisov, Samuel Canul Yah, Lorena Hau Ucán, Marvin Agüero-Torales, Aldo Alvarez, Silvia Fernandez Sabido, Alejandro Molina Villegas, Abteen Ebrahimi, Robert Pugh, Arturo Oncevay, Shruti Rijhwani, Rolando Coto-Solano, Katharina von der Wense, and Manuel Mager. 2024. Findings of the AmericasNLP 2024 shared task on the creation of educational materials for indigenous languages. In *Proceedings of the 4th Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP)*. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.