The role of morphosyntactic similarity in generating related sentences

Michael Hammond. U. of Arizona

Overview

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.

1. A fairly generic transformer model.
2. Our own implementation of the edit tree approach from the baseline system.
3. 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. Here are some sample data:

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

Table: Sample data

Transformer

The first model we tried was a simple transformer (Vaswani et al., 2017). Our specific transformer architecture is an adaptation from `https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html`.

Specifics:

- We 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.

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: Performance with transformer model on dev data

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 below 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 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).

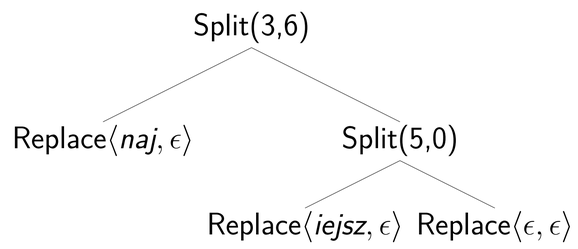

Figure: Example edit tree (Chrupała, 2008, p.127)

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:

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

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 above would be represented as:

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

To convert these to a transduction, 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 above, 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 below.

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: Performance with our implementation of edit trees on dev data

Morphosyntactic similarity

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.

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 below.

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: Performance with morphosyntactic tag similarity on test data

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Chiruzzo, L., Denisov, P., Canul Yah, S., Hau Ucán, L., Agüero-Torales, M., Alvarez, A., Fernandez Sabido, S., Molina Villegas, A., Ebrahimi, A., Pugh, R., Oncevay, A., Rijhwani, S., Coto-Solano, R., von der Wense, K., and Mager, M. (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.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chrupała, G. (2008). *Towards a machine-learning architecture for lexical functional grammar parsing*. PhD thesis, Dublin City University.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

All code available here

`https://github.com/hammondm/americasnlp24task2`