Bootstrapping a Neural Morphological Generator from Morphological Analyzer Output for Inuktitut

Jeffrey Micher
US Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783
jeffrey.c.micher.civ@mail.mil

Abstract

We present a method for building a morphological generator from the output of an existing analyzer for Inuktitut, in the absence of a two-way finite state transducer which would normally provide this functionality. We make use of a sequence to sequence neural network which "translates" underlying Inuktitut morpheme sequences into surface character sequences. The neural network uses only the previous and the following morphemes as context. We report a morpheme accuracy of approximately 86%. We are able to increase this accuracy slightly by passing deep morphemes directly to output for unknown morphemes. We do not see significant improvement when increasing training data set size, and postulate possible causes for this.

1 Introduction

Morphological generation is the process of transforming an underlying sequence of morphemes (for example, a lemma or stem, plus inflections) into a surface realization of those morphemes. For many languages which have complex morphology, morphological generation is a necessary step in certain language processing applications such as machine translation. Finite state transducers (FSTs) have been the main technology used in morphological analysis and generation. Most finite state transducers can operate in both directions, the analysis, or upward, direction and the generation, or downward, direction (Beesley & Karttunen, 2003). The Uqailaut morphological analyzer (Farley, 2009), which provides an analysis of Inuktitut words, however, only operates in the "analyze" direction, because this analyzer was hard-coded, not using currently widely used tools such as *xfst* (Beesley & Karttunen, 2003). To make up for this lack of functionality, we present a method for bootstrapping the output of the morphological analyzer to build a morphological generator that will work in the opposite direction, inspired by the analyzer bootstrapping technique in Micher (2017). We do this using a sequence to sequence neural network architecture based on encoder-decoder networks to "translate" from the deep morpheme sequences to their corresponding surface forms, and we present accuracy results. We show that this technique has promise when building morphological generators when the associated analyzer does not operate in reverse. We envision the use of this generator as a post-process to an English-Inuktitut machine translation system which translates English into sequences of underlying Inuktitut morphemes, to convert those deep morphemes to surface forms.

2 Inuktitut Morphophonemics

Inuktitut is a polysynthetic language spoken in the Canadian territory of Nunavut and other regions of Arctic Canada. Inuktitut words tend to be very long because many morphemes can be added onto roots iteratively, often generating words which would correspond to full clauses in other languages. In general, Inuktitut words consist of a root followed by zero or more lexical postbases, followed by a grammatical suffix and an optional clitic (Dorais, 1990). The surface realization of each morpheme is based on specific morphophomemic rules which are unique to each morpheme and not conditioned wholly on their phonetic environment. For example, the underlying morpheme sequence for the word "mivviliarumalauqturuuq" meaning "he said he wanted to go to the landing strip" is "mit+vik+liaq+juma+lauq+juq+guuq." The spelling rule for each morpheme must be learned individually, and also the final surface spelling

Proceedings of the 3rd Workshop on the Use of Computational Methods in the Study of Endangered Languages: Vol. 2 Extended Abstracts, pages 37–41,
Honolulu, Hawai'i, USA, February 26–27, 2019.

will be affected by the previous and following morphemes. We work from the end to the beginning to understand this phenomenon in this example. The morpheme ‘guuq’ is a UVULAR ALTERNATOR [1], which means that the first phoneme ‘g’ will change according to what the previous morpheme ends with. In this case, it surfaces as ‘r’ because of the uvular ‘q’ before it. The rule also deletes a previous consonant, so the ‘q’ of ‘juq’ gets deleted. Next, ‘juq’ is a CONSONANT ALTERNATOR, which means the first consonant gets spelled based on the ending of the previous morpheme. In this case, it comes out as ‘t’ because the previous morpheme ends with a consonant. Next, ‘lauq’ is NEUTRAL, so there are no changes (but if the following morpheme was a UVULAR ALTERNATOR, it would have lost its final ‘q’). Next, ‘juma’ is like ‘guuq’ so it gets spelled ‘ruma’ and deletes the preceding ‘q’. Next, ‘liaq’ is a DELETER, so it deletes the previous morpheme’s final consonant ‘k’ and recall its ‘q’ was already deleted. Next, ‘vik’ is a VOICER, which causes the previous ‘t’ to completely assimilate to ‘v’.

3 The Uqailaut Analyzer and Sample Output

The Uqailaut morphological analyzer takes a single input word and produces an analysis or set of possible analyses for words those words. An analysis consists of a sequence of morphemes in curly braces. Each morpheme consists of its surface form, deep form, and relevant morphological information such as person and number in the case of verbs, as is depicted below:

{<surface form>:<deep form>/<morphological analysis information>}{..}{..}..etc.

The following shows a typical analysis for the word ‘maligarmut,” meaning “bill, law; something that one follows,” in the dative case. For words with ambiguous analyses, each analysis is given on a separate line:

{maligar:maligaq/1n}{mut:mut/tn-dat-s}
{mali:malik/1v}{gar:gaq/1vn}{mut:mut/tn-dat-s}

The Nunavut Hansard data set (Martin, Johnson, Farley, & Maclachlan, 2003), derived from Nunavut legislative proceedings, was processed with the Uqailaut (Micher, 2018a) analyzer to provide data for morphological analysis and machine translation experiments (Micher, 2018b), and is used again in this current set of experiments.

4 Our Model

We consider the task of morphological generation for Inuktitut as a sequence to sequence processing task similar to that of machine translation, and as such, we model our generator after current designs for machine translation and follow the work of (Kann & Schütze, 2017) and (Faruqui, Tsvetkov, Neubig, & Dyer, 2015). Specifically, we use an encoder-decoder architecture with attention (Bahdanau, Cho, & Bengio, 2015) to encode input sequences of morphemes into a hidden state, then decode them into surface characters. The encoder is a bidirectional LSTM (Hochreiter & Schmidhuber, 1997) and the decoder is a character RNN. Different from MT models, however, we limit the encoder to consider only the current, previous, and following morphemes, which we essentially “translate,” letting the attention mechanism figure out that the sequence of three morphemes is focused on the central morpheme. The limiting of the context to the previous and following morpheme reflects the linguistic context for the operation of the morphophonemic rules of Inuktitut discussed above. The figure below depicts the architecture.

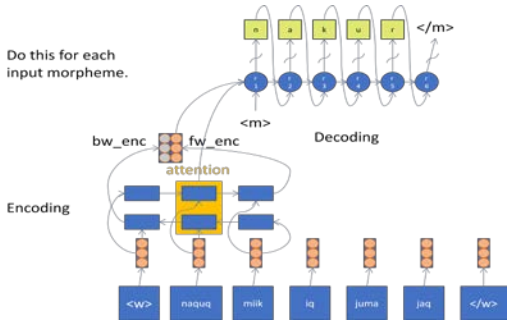

Figure 1: Architecture

[1] The names of the rules are those of (Mallon, 2000)

5 Experiments

For data, we use the same set of morphologically analyzed Inuktitut words as used in (Micher, 2017) having a single morphological analysis, and hold out two sets of 1000 items (labeled 'test' and 'dev'), leaving approximately 23,000 training items.

Our baseline system makes use of the architecture described above. Morpheme embeddings are sized at 128, LSTM hidden states at 128, with 2 layers. We completed 35 epochs with SGD and a simple learning rate. We did not use mini-batching because of the small size of training data. As is common in neural models, we replace infrequent deep morphemes with an <unk> symbol for those having fewer than two instances in the data, and verify that all of the morphemes present in the test sets are found in the training set.

Our second system, Baseline+Backoff, makes use of a simple concept: when an unknown morpheme is encountered, rather than having the system try to generate a correct character sequence, we copy the deep morpheme directly to the surface, with no changes. Because surface forms often contain many of the same character sequences as deep forms, this allows us to "guess" at the right form for those morphemes which may not be sufficiently represented in the data to accurately learn their behavior.

6 Results

We report a full word accuracy score: i.e. the percentage of test items completely correct; a morpheme accuracy score, i.e. how many morphemes are correct, on average, per word; an average Levenshtein edit distance, normalized over word length, by dividing the edit distance score per word by the number of characters in the original word; and a character BLEU-4 (Papineni, Roukos, Ward, & Zhu, 2002) score. While the BLEU-4 score is traditionally used to compare MT system output between different systems, we felt that, as a modified precision score, which accounts for length, we could use it to capture a character-level accuracy. The table below displays the systems and the scores obtained over the results from running the 'test' set through the model.

	Full Word Accuracy	Morpheme Accuracy	Ave. Leven-shtein Distance	Character BLEU-4
Baseline	60.46	86.24	0.036	91.89
Baseline +Backoff	61.50	87.65	0.030	92.64

Table 1: Results

As can be seen, the Baseline+Backoff model performs better on all metrics. It is interesting to note that, even in the non-character based metrics, we get an improvement by simply copying over unknown deep morphemes to the surface. What this reflects is that a certain percentage of morphemes are identical in their deep and surface forms and the model is making mistakes on these, likely, rare morphemes.

We further experimented with adding more training data. From the morphologically analyzed words in the Nunavut Hansard, we used those that had 2, 3, and 4 analyses in addition to the data already used.

# Analyses	# Words	# Training Items
2	28,841	57,682
3	18,519	55,557
4	21,275	85,100

Table 2: Additional Training Data Set Sizes by Number of Analyses

From these data, we created training sets in increments of 25K items over the baseline set, up to 200K items. Each incremental set contained all of the training items from all of the smaller sets, including the baseline data set, using up first the 2-analyses set, then the 3-analyses, and then the 4-analyses sets. As such, the divisions in the training sets do not correspond exactly to the divisions based on number of analyses. We trained using the baseline system described earlier, and tested using the same held out sets as reported for the baseline system. As of the writing of this paper, we are not seeing any significant improvements from the addition of training data over the baseline. All systems are converging at roughly 86% morpheme accuracy, once comparable amounts of data have been seen over several epochs of training. This result may seem counter to general trends in neural network training, in which greater amounts of training data produces better results. However, it should be noted that the different analyses that are provided

by the Uqailaut analyzer are probably noisy: Uqailaut produces all possible analyses whether they are likely or semantically plausible or not. Also, the Uqailaut analyzer is built to account for dialectal spelling variation, which is frequent in the Nunavut Hansard, so learning a single, unambiguous application of a spell out rule of an underlying morpheme sequence from these data may be impossible. A thorough error analysis could shed some light on what is happening with these training data and why additional data are not producing a more accurate system.

7 Future work

Ideally, we would like to refine this approach and get higher accuracy scores, within the 90% and above range, but as accurate as possible since we envision using this model in a downstream machine translation system, in which we hope to minimize the cascading of errors that is often seen in pipelined approaches. Thus, in future work, we will conduct an error analysis to see why there is not more improvement with greater amounts of training data, and we will use an alternative source of data which can be vetted for accuracy and restricted to a single dialectal variant. Also, we will try a comparable system which uses full-word morpheme history instead of only the previous and following morpheme to account for any possible long distance dependencies that may be present in the data. Finally, we will experiment with an unknown morpheme backoff to a character-level encoder, which may show further improvement as specific characters in an unknown morpheme will become salient for purposes of morphophonemic rule application.

8 Related work

Much work has been done on morphological generation. Recent work has focused on morphological reinflection (Cotterell, et al., 2016), (Cotterell, et al., 2017) in which an inflected form is given, and a desired (different) inflected form should be produced. Faruqui et al. (2015) show that a character-level neural model can predict surface forms from base forms plus morphological inflection information. In our work, however, we investigate how well this technique works when only morphological context is provided and no explicit morphological rules or inflection information is given.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *CORR*. Retrieved from http://arxiv.org/abs/1409.0473

Beesley, K. R., & Karttunen, L. (2003). *Finite State Morphology.* Palo Alto: CSLI Publications.

Cotterell, R., Kirov, C., Sylak-Glassman, J., Walther, G., Vylomova, E., Xia, P., . . . Hulden, M. (2017). CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection in 52 Languages. In M. Hulden (Ed.), *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection* (pp. 1-30). Vancouver: Association for Computational Linguistics.

Cotterell, R., Kirov, C., Sylak-Glassman, J., Yarowsky, D., Eisner, J., & Hulden, M. (2016). The SIGMORPHON 2016 Shared Task—Morphological Reinflection. In M. Elsner, & S. Kuebler (Ed.), *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology* (pp. 10-22). Berlin: Association for Computational Linguistics.

Dorais, L.-J. (1990). The Canadian Inuit and their Language. In D. R. Collins, *Arctic Languages An Awakening* (pp. 185-289). Paris: UNESCO.

Farley, B. (2009). *The Uqailaut Project*. Retrieved from Inuktitut Computing: http://www.inuktitutcomputing.ca/Uqailaut/info.php

Faruqui, M., Tsvetkov, Y., Neubig, G., & Dyer, C. (2015). Morphological Inflection Generation Using Character Sequence to Sequence Learning. Retrieved from http://arxiv.org/abs/1512.06110

Hochreiter, S., & Schmidhuber, J. (1997). Long Short Term Memory. *Neural Computation, 9*(8), 1735-1780.

Kann, K., & Schütze, H. (2017). Unlabeled Data for Morphological Generation With Character-Based Sequence-to-Sequence Models. *CoRR*. Retrieved from http://arxiv.org/abs/1705.06106

Mallon, M. (2000). *Inuktitut Linguistics for Technocrats*. Retrieved from Inuktitut Computing: http://www.inuktitutcomputing.ca/Technocrats/ILFT.php

Martin, J., Johnson, H., Farley, B., & Maclachlan, A. (2003). Aligning and Using an English-Inuktitut Parallel Corpus. Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel

Texts: Data Driven Machine Translation and Beyond - Volume 3 (pp. 115-118). Stroudsburg, PA, USA: Association for Computational Linguistics.

Micher, J. (2017). Improving Coverage of an Inuktitut Morphological Analyzer Using a Segmental Recurrent Neural Network. *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages* (pp. 101-106). Honolulu, HI: Association for Computational Linguistics.

Micher, J. (2018a). Provenance and Processing of an Inuktitut-English Parallel Corpus, Part 1. Adelphi, MD: U.S. Army Research Laboratory.

Micher, J. (2018b). Using the Nunavut Hansard Data for Experiments in Morphological Analysis and Machine Translation. *Proceedings of the Workshop on Computational Modeling of Polysynthetic Languages* (pp. 65-72). Santa Fe, NM: Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 311-318). Stroudsburg, PA, USA: Association for Computational Linguistics.