MONASH University

FIT5217 NLP

Assignment 2:
Neural Methods for Morphological Reinflection
Semester 1, 2021

ID: 32039131

1. Introduction

Morphological reinflection is the task of generating a word inflection from a lemma (source word). Given the diversity of morphological systems, it presents many hurdles to computational approaches. The SIGMORPHON shared task series is devoted to this challenge. In recent years, neural networks have been without a doubt the top performer in NLP tasks and have repeatedly achieved state of the art performance on a wide variety of challenges, including morphological reinflection.

The aim of this report is to investigate modern neural-network based approaches to languages, their performance and limitations. In particular, it details the implementation, evaluation and potential methods of improvement of neural models for the task of morphological inflection. To achieve this, we utilise the SIGMORPHON shared task 'morphological inflection' challenge, and the accompanying inflection datasets, as a framework for which to base our investigation.

A brief discussion of recent work in morphological inflection is presented in Section 2. Section 3 details the implementation of a hard-attention model (Aharoni & Goldberg, 2017) and a transformer model (Wu et al., 2021), as well as the experiments conducted to improve their performance on several low-resource languages. Low-resource languages (small datasets) are chosen for our evaluation as they present a good opportunity for increasing performance, and our attempts are primarly focussed on data augmentation techniques. Finally, evaluation and error analysis of each approach are discussed in section 4.

2. Related Work

Prior to 2017, many of the proposed systems took a non-neural approach using deterministic and stochastic alignment methods often with linguistic heuristics, and had reasonable success. In recent years, neural methods have become the norm and they have been very succesful. The 2016 SIGMORPHON shared task reports on the outstanding performance gains that neural methods have over classical approaches.

(Graves, 2014) reports on the benefits of using LSTM recurrent neural networks to generate long-term dependent sequences. These have been applied to character-level transduction tasks such as morphological inflection successfully(Faruqui et al., 2016). where the problem is modeled as a simple sequence to sequence task. Future work improved on this primarily in two ways- first by making an assumption about the often- monotonic alignment of characters, and either building a system that has nearly-monotonic constraints(Aharoni & Goldberg, 2017) or exact-monotonic constraints (Wu & Cotterell, 2019). Second, the use of an attention mechanismto allow direct conditioning on the whole input sequence, mitigating the RNN bottleneck issues. Recently, a transformer (attention-only) model as been applied to the task with the intuiting that as transformer models are outperforming RNNs in many NLP tasks, they warrant investigation for character level (Wu et al., 2021). Transformer-based models currently currently hold state of the art performance (Liu & Hulden, 2020) on the SIGMORPHON challenge of morphological reinflection (Vylomova et al., 2020) and (more broadly) is a popular choice for many NLP tasks.

3. Baseline and Experimental Methods

This section is broken into two parts. The initial section introduces the two neural baseline methods' for the task of morphological inflection, with summary differences explored conceptually (section 4 compares their performance). In the second part, expriments to improve upon these initial methods are detailed. We aim to investigate the performance of the various models and potential improvements- specifically on low-resource languages. To this end, baseline experiments are conducted on Crimean-Turkish (crh), Swahili (swa), Maltese (mlt), Ingrian (izh) and Murrinh-Patha (mwf); and improved methods are applied to Maltese (mlt), Ingrian (izh) and Murrinh-Patha (mwf). Ingrian and Murrinh-Patha are chosen as low-resource data sets (n<1000), with Maltese as a type of cross-over 'control' with the baseline experiments.

3.1 Baseline Systems

Model 1: Hard-Attention model (Aharoni & Goldberg, 2017 SIGMORPHON baseline model)

This model uses a combination of bi-directional RNN for encoding/decoding sequences, with an attention mechanism with a nearly-monotonic hard alignmement constraint- called a 'hard attention' mechanism to determine what to pay attention to.

At a high level, the model's prediction process can be understood as a series of attention-pointer movements ('reads') and write operations. The encoder takes every (vectorised) character input and passes it through a bidirectional LSTM RNN to generate a hidden representation. The decoder takes as input its previous output, a set of features (e.g. corresponding to unimorph tags), and the hidden state that the attention mechanism has determined. A traditional (soft) attention mechanism is a way to create an output as a weighted sum of the input characters (weighted by how useful the network believes they are). In this implementation however, the attention control mechanism controls what is the appropriate step: to decode and write the current ('attended') input, or advance the decoders attention to the next input, akin to an alignment control.

Model 2: Transformer model applied to character-level transduction (Wu et al., 2020 baseline model)

The second model implemented has a transformer-based architecture. Traditionally, transformer models have performed very well on word-level tasks (Vaswani et al., 2017)b. It turns out that they work well on character-level tasks as well, even when training examples are limited, providing the model implementation is tweaked a bit (notably the batch size). In contrast to model 1 above, the transformer architecture uses a soft-attention mechanism.

Transformer models differ from RNN models in that they rely on a self-attention mechanism, consequently having attractive computational/parallelizable characteristics and being better at capturing capture long-range dependencies. This implementation uses a relatively small version of the transformer under the intuition of low-resource data sets creating potential for overfit. Other notable modifications from the 'vanilla' transformer is adapting for feature positional invariance and running the decoder left-right greedily.

3.2 Further experiments and improvements

Supervised-learning neural models traditionally perform the best when there are large amounts of training data available. When the data sets are small, overfitting/specialisation is a risk. Low-resource languages present a challenge in this regard, and data augmentation can be a very worthwhile technique to improve predictive performance. There have been promising results within morphological inflection with low-resource languages in particular (Anastasopoulos & Neubig, 2019).

The existing 'data hallucination' process as outlined in the 2020 baseline approach is outlined as follows:

- Take as input the original training data
- Find 1 or more 'stem' sequences (defined as 3 or more consecutive characters common to both source and inflected form)
- Replace these stems with 'hallucinated' (e.g. fabricated) character sequences
- Repeat until desired data set size is achieved.

This process has some potential limitations. First, the replacement characters are sampled uniform-randomly from the languages alphabet. This may result in unrealistic sequences, potentially impacting on the training of the model as it seeks to find patterns within the hallucinated data.

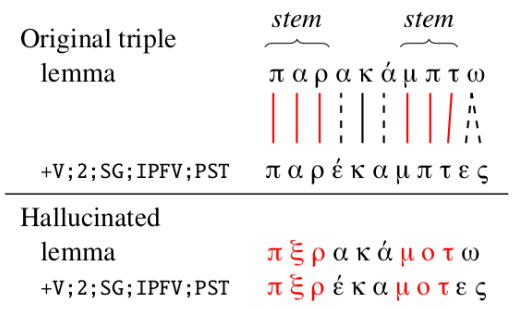

Figure 3.2.1 – Random sampled hallucination (Anastasopoulos & Neubig, 2019

Second- the replacement length is strictly matching that of the original length. This limits the variety of word forms present in the training data, which may result in influencing the types of patterns the model learns to focus on, i.e. if there are now 10x the number of lemma→ inflected form pairs for a given rule, but they are all the same length, then the model will be biased to words of that length. This could limit the overall benefit that the augmentation has on the model.

To attempt to remedy these potential shortfalls, we propose the following:

- Sampling of sequences will be without replacement from a categorical distribution utilising a markov model (over observed sequences in the original training data), where the probability of a replacement is conditioned on the preceding n-characters (i.e. ngram)

- Replacement sequences can have the following flexible characterstics:
  - They can be k +/- m characters long, where k is the original sequence length, and m <= n
  - They can be 'fixed' or 'compositional'; the former is where the sequence must have been observed in the training data in its entirety, the latter is where the sequence can be made up of recursive sampling
  - (providing it does not conflict with the above choices) a maximum sequence length may be specified (e.g. when compositional, we may implement a strict character-level unigram model if desired)
- As with the previous method, multiple 'stems' can be replaced, however in this implementation it is possible to specify the minimum length of the stem (default is 2)

Implementation Outline

- First, a mapping vocabulary is built from all source & inflected words in the training data set:
  - each word has '#' and '$' affixed to mark the beginning and end of a word
  - for each character/sequence of characters, a mapping is constructed to all observed succeeding sequences <= max_len length (greedy), along with their frequency of occurance
  - frequencies are converted to a cumulative sum to allow for sampling from the categorical distribution.

```
------
Chars succeeding # in source word or a in target word:
[('n', 573), ('r', 705), ('m', 835), ('b', 986), ('y', 1049), ('i', 1075), ('e', 1097), ('l', 1170), ('a', 1194), ('k', 1256), ('t', 1378), ('d', 1398), ('p', 1543), ('w', 1554),
('n', 2670), ('m', 3376), ('r', 4111), ('b', 4847)]
------
```

Fig. 3.2.2a: Example distribution of 1-char (max_len = 1) sequences succeeding the chars '#' or 'a'

```
------
Chars succeeding # in source word or a in target word:
[('ngi', 82), ('rra', 108), ('mi', 133), ('me', 170), ('ntj', 196), ('bu', 224), ('ye', 252), ('nu', 280), ('bi', 302), ('rdi', 324), ('nth', 351), ('rri', 382), ('buy', 412), ('b
a', 440), ('mu', 469), ('na', 497), ('ya', 523), ('nga', 580), ('i', 606), ('rdu', 634), ('e', 681), ('rru', 706), ('la', 759), ('a', 783), ('be', 812), ('li', 866), ('ngu', 1003)
, ('kir', 1008), ('kum', 1014), ('men', 1018), ('num', 1033), ('ner', 1037), ('tha', 1052), ('kun', 1058), ('nub', 1079), ('nun', 1097), ('kud', 1100), ('dir', 1103), ('pur', 1109
), ('pun', 1120), ('pub', 1142), ('thu', 1208), ('puy', 1216), ('ne', 1221), ('pir', 1232), ('nud', 1240), ('pum', 1256), ('kam', 1257), ('den', 1260), ('nuy', 1273), ('bam', 1275
), ('nin', 1293), ('par', 1301), ('nge', 1314), ('nam', 1316), ('pil', 1327), ('tji', 1362), ('pim', 1363), ('kar', 1366), ('kuy', 1369), ('pin', 1385), ('nul', 1391), ('pan', 139
8), ('dil', 1402), ('ku', 1405), ('nem', 1406), ('pib', 1407), ('pi', 1410), ('yun', 1416), ('nir', 1423), ('ka', 1427), ('de', 1430), ('wur', 1438), ('bem', 1439), ('kan', 1443),
 ('pe', 1444), ('tje', 1450), ('kin', 1453), ('kub', 1461), ('pu', 1463), ('ke', 1467), ('nar', 1473), ('bim', 1474), ('kil', 1475), ('pa', 1476), ('min', 1478), ('pam', 1481), ('
nil', 1488), ('pud', 1495), ('ben', 1497), ('nim', 1498), ('din', 1499), ('mem', 1501), ('pul', 1504), ('kur', 1507), ('kul', 1510), ('nur', 1514), ('dam', 1517), ('wul', 1520), (
'mun', 1524), ('bin', 1526), ('bun', 1528), ('ban', 1532), ('ki', 1534), ('nan', 1536), ('yin', 1539), ('per', 1546), ('muy', 1548), ('nen', 1550), ('nib', 1551), ('dem', 1552), (
'dan', 1553), ('duy', 1554), ('ntj', 2542), ('m', 3114), ('ngi', 3712), ('na', 4353), ('n', 5015), ('rra', 6348), ('nga', 7721), ('rne', 8420), ('rde', 9122), ('rni', 9829), ('nye
', 10539), ('nge', 11252), ('nyi', 11969), ('mam', 12690), ('rri', 13415), ('rdi', 14145), ('rna', 14877), ('bim', 15610), ('ne', 16344), ('nam', 17080)]
------
```

Fig. 3.2.2b: Example distribution of <= 3-char (max_len = 3) sequences succeeding the chars '#' or 'a'

- Each word pair (lemma & inflected) in the training set has LCS (longest common subsequence) computed. The LCS is then split into substrings of characters that are consecutive in both words.
- For each LCS substring that meets minimum length criteria:
  - the string is replaced R times, by sampling without replacement from the vocabulary, either compositionally or as a fixed sequence (defined above). R = (target set size - existing size)/(existing size *(#lcs_substrings)
  - If the vocabulary is exhausted during replacement, move on to the next word pair
- This process is repeated for all word pairs in the training data set, and the resulting data-set is added to the original data-set to create an augmented set of the desired size.

```
Old words: ngi, ngungi
 New words: ['miri', 'rdii', 'ntji', 'birii', 'rraii', 'nuyi', 'menii', 'nuyyi', 'menni', 'buyi'],
['ngumiri', 'ngurdii', 'nguntji', 'ngubirii', 'ngurraii', 'ngunuyi', 'ngumenii', 'ngunuyyi', 'ngumenni', 'ngubuyi']
```

Fig. 3.2.3: Example hallucinated words for the pair ngi→ ngungi

Evaluation and analysis of both the baseline methods and the improvements noted above are detailed in the next section.

It is worthwhile to note that there has also been strong evidence showing the benefits of cross-lingual training, where a model is trained on languages from the same language family (e.g. niger-congo) as they often contain very similar morphological structure. This obviously is not possible when there is only one language resource available for the given family (e.g. Southern Daly: Murrinh-Patha).

4. Evaluation and Error Analysis

Evaluation has been carried out utilising the following language data sets:

Language	ISO 693-3	Family	Genus	Train #	Dev #
Ingrian	izh	Uralic	Finnic	763	112
Murrinh-Patha	mwf	Southern-Daly	Murrinh-Patha	777	111
Maltese	mlt	Afro-Asiatic	Semitic	1233	176
Crimean Tatar;	crh	Turkic	Turkic	5215	745
Swahili	swa	Niger-Congo	Bantu	3374	469

Overall baseline performance is measured for all 5 languages, whereas the improvements are evaluated on only the first 3 languages due to focus (low-resource languages) and time.

4.1 Baseline Performance

We compare the two neural baseline methods' (described in section 3.1) performance in terms of overall accuracy and Levenshtein distance on a subset of data sets from the SIGMORPHON 2020 shared task challenge task 1. (Refer to the appendix for the parameters chosen)

Overall Performance

Development Set

Model	Ingrian		Murrrinh-Patha		Maltese		Swahili		Crimean-Turkish	
	Acc	Lev	Acc	Lev	Acc	Lev	Acc	Lev	Acc	Lev
2017-Hard	70.54	0.87	0.87	0.34	86.55	0.2	100	0	98.12	0.27
2020-Trm	83.04	0.34	91.89	0.14	94.89	0.07	100	0	99.06	0.01

Test Set

Model	Ingrian		Murrrinh-Patha		Maltese		Swahili		Crimean-Turkish	
	Acc	Lev	Acc	Lev	Acc	Lev	Acc	Lev	Acc	Lev
2017-Hard	62.5	1.23	82.05	0.31	86.46	0.2	100	0	96.29	0.58
2020-Trm	79.91	0.37	87.84	0.28	93.77	0.09	100	0	98.72	0.02

The transformer model outperforms the hard-attention RNN model in every test set except for Swahili, where both models achieve 100% accuracy.

The two models have a similar relative performance with respect to the language, i.e. their ranked performance of language is the same: Swahili, Crimean-Turkish, Maltese, Murrinh-Patha, Ingrian(in descending accuracy order).

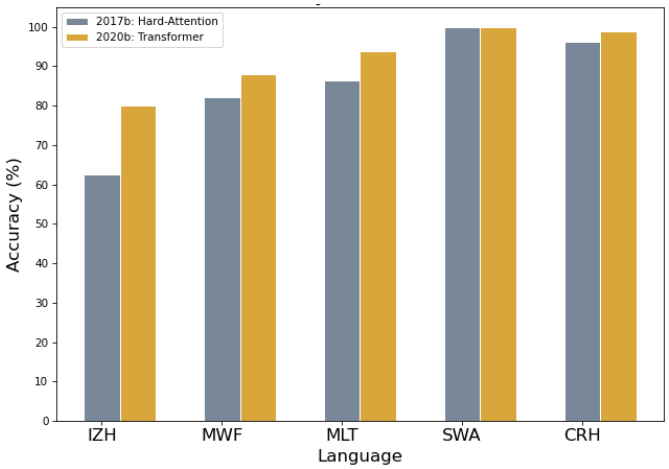

Fig. 4.1.1: Two model Baseline performance

The lowest performing languages for both models are Ingrian and Murrinh-Patha. Both have quite small data sets (<800 training samples). Inspecting this further, we see a trend with the size of the data-set and each models' performance (figure 4.1.2 below). The ranking of languages by their data-set size is the same as above. This hints at the performance dependency of data-set size, and potential benefits of augmenting training data.

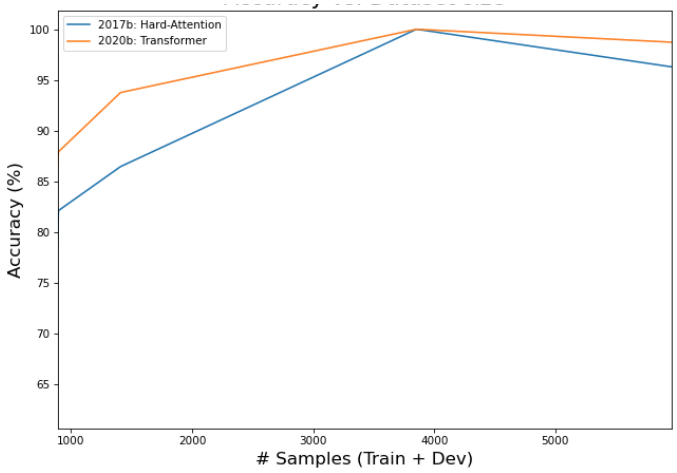

Figure 4.1.2: Accuracy vs data set size

4.2 Improvement Evaluation

From section 4.1 and the existing literature (Vylomova et al., 2020) It is clear that there is a relationship between a neural model's performance and the size of the training data available to it. Therefore, we attempt to improve upon the best performance achieved in section 4.1 (with that of the transformer model) by implementing the modified data augmentation methods discussed in section 3.2.

Overall Performance

For initial comparisons, we attempted to include the baseline 'uniform random' augmentation methods as implemented by Anastasolous & Neubig. The augmentation was implemented following default parameters and guidelines outlined in their approach (from their github repo). However, there was obviously an error in the process as the best performance achieved was very poor. Preliminary troubleshooting was performed with marginal difference, and a thorough troubleshooting was beyond the scope of this assignment. The results have been included for completeness, however should not be relied upon.

(Note: Levenshtein distances have been normalised)

Development Set

Model	**Ingrian**		**Murrrinh-Patha**		**Maltese**	
	Acc	Lev	Acc	Lev	Acc	Lev
Unaugmented	83.03	0.34	91.89	0.14	94.88	0.07
*Random-Augmented**	30.35	1.18	19.9	2.1	52.84	0.94
Markov-Augmented	84.82	0.27	92.79	0.13	95.45	0.08

Test Set

Model	**Ingrian**		**Murrrinh-Patha**		**Maltese**	
	Acc	Lev	Acc	Lev	Acc	Lev
Unaugmented	79.91	0.36	87.83	0.28	93.76	0.09
*Random-Augmented**	**26.78**	2.07	13.96	2.58	58.07	0.89
Markov-Augmented	**83.48**	0.37	88.29	0.26	94.62	0.07

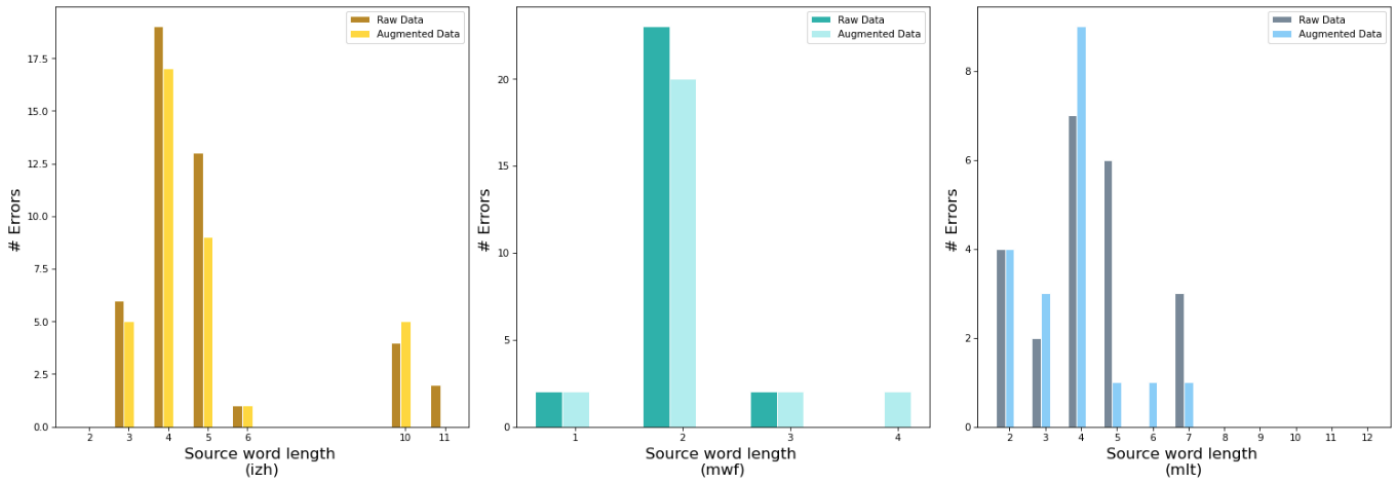

Figure 4.2.1 – Impact of augmented data on prediction errors, broken down by word length

Analysis

Inspecting the raw results, we see a marginal improvement in performance of the transformer model using the augmented data, with the exception of Ingrian which sees a relatively large boost of around 3.5% accuracy. Inspecting this further, we see that the number of errors relative to the word length (figure 4.2.1) reduces in most cases where there were errors originally; however the most common errors remain the most common, and a small number of errors have also been introduced where there were none previously. A possible interpretation for the results above, in terms of the languages chosen, might be as follows:

- Ingrian and Murrinh-Patha are both very low resource datasets, wheras Maltese has roughly twice as many examples. The performance on Maltese is already quite good, and the model does not seem to benefit much from augmenting further with this technique.
- Ingrian contain many words of length > 4 characters and often a rich combination of character sequences, in theory this might lead to a good source of augmented data. Wheras Murrinh-Patha's words are, on average, much shorter, and see less variation in the character combinations. This means a relatively restricted set of data for which to build up the augmentation vocabulary.

Perhaps the largest impact that the data augmentation has is on the training speed. Figure 4.2.2 shows the marked increase in speed at which the model achieves its (approximately) best performance, somewhere around the 40th epoch, compared to the unaugmented data set requiring 250 epochs.

This could be very beneficial when training more complex architectures as the practicalities of computation is often a limiting factor.

We can also see the rather arduous training journey the model takes with the uniform-augmented data. For now this remains a mystery.

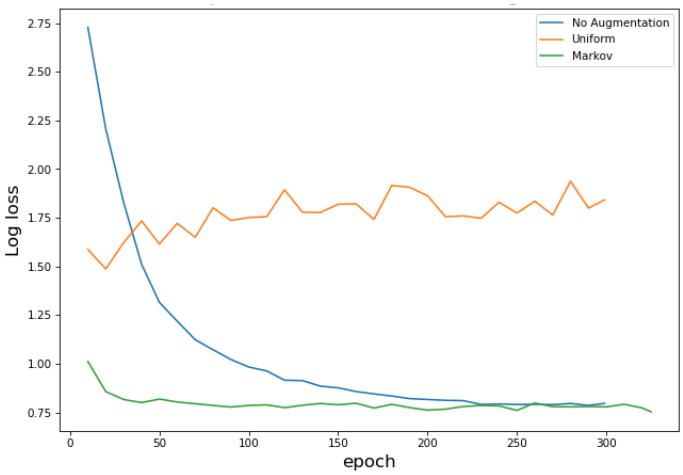

Figure 4.2.2 Effect on dev loss vs. training iteration (IZH)

Parameter Tuning

There appears marginal difference with the different parameter settings. The best performing settings do outperform the raw data set marginally (please refer to the appendix for the table of results). Reducing the LCS minimum seems to have the largest impact. In fact, initially the implementation had no minimum length setting, so it would match a subsequence of 1 character (providing it existed in a larger LCS match). This created many errors; in particular with Murrinh-Patha given its shorter word lengths.
This result confirms a reasonable choice for a consecutive sequence replacement of 2-3 characters. Longer than this would yield little benefit as it would limit the number of words that could be augmented.

Whereas the parameters above had a relatively small impact on performance, the amount of augmented data (target data-set size) had a large impact on performance (figure 4.2.3 below)- mainly due to the extremely negative impact on the mwf data-set. This could be attributed to the sequence invariance in the training data (as mentioned above). A size of 2000 was found to be consistently optimal; it tells us that the transformer model is quite good with moderately low training examples and to bolster it with more hallucinated data hinders more than helps. (Potentially if with future work the augmentation techniques became a bit more flexible, an increase in size could be warranted).

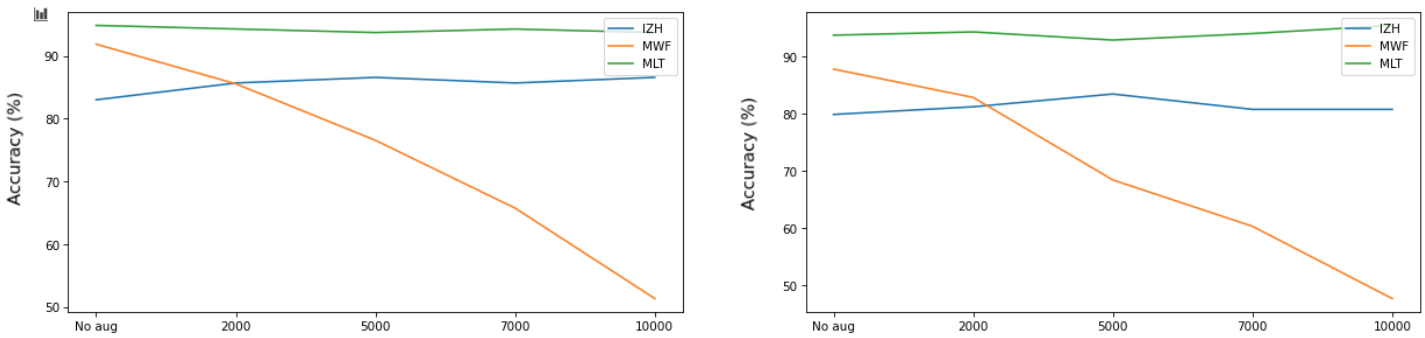

Figure 4.2.3 Effect of augmentation amount on prediction accuracy

Conclusion

Morphological reinflection is the task of generating a word inflection from a lemma (source word), and is an important and challenging task of Natural Language Processing. Over the past several decades there have been many advancements in approaches to tackling morphological reinflection, with the most modern and substantive of these with the use of neural networks.

The goal of this assignment and report was to investigate and develop an understanding of neural network approaches to NLP tasks. To achieve this, two baseline neural models differing in architecture were implemented, and an attempt at improving upon their performance was undertaken through the augmentation of training data. Two low resource languages (Ingrian and Murrinh-Patha) that have had poor relative performance in the 2020 shared task challenge were chosen to highlight the challenge.

We demonstrated that augmenting the training data using a categorical sampling distribution based on a Markov model can yield some level of improvement, and that there is benefit to augmenting data specific to the requirements of a language family and it's morphological paradigms. Unfortunately we were unable to gain a meaningful comparison to existing augmentation techniques due to issues with implementation. Future work could investigate alternative methods such as the use of unsupervised neural methods to augment data (via auto encoders / variational auto encoders) and self-supervised training techniques (such as word completion or Unimorph tag to word generation).

References

Aharoni, R., & Goldberg, Y. (2017). Morphological Inflection Generation with Hard Monotonic Attention. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2004–2015. https://doi.org/10.18653/v1/P17-1183

Anastasopoulos, A., & Neubig, G. (2019). Pushing the Limits of Low-Resource Morphological Inflection. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 984–996. https://doi.org/10.18653/v1/D19-1091

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *ArXiv:1607.04606 [Cs]*. http://arxiv.org/abs/1607.04606

Faruqui, M., Tsvetkov, Y., Neubig, G., & Dyer, C. (2016). Morphological Inflection Generation Using Character Sequence to Sequence Learning. *ArXiv:1512.06110 [Cs]*. http://arxiv.org/abs/1512.06110

Graves, A. (2014). Generating Sequences With Recurrent Neural Networks. *ArXiv:1308.0850 [Cs]*. http://arxiv.org/abs/1308.0850

Liu, L., & Hulden, M. (2020). Leveraging Principal Parts for Morphological Inflection. *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 153–161. https://doi.org/10.18653/v1/2020.sigmorphon-1.17

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (n.d.). *Attention is All you Need*. 11.

Vylomova, E., White, J., Salesky, E., Mielke, S. J., Wu, S., Ponti, E. M., Hall Maudslay, R., Zmigrod, R., Valvoda, J., Toldova, S., Tyers, F., Klyachko, E., Yegorov, I., Krizhanovsky, N., Czarnowska, P., Nikkarinen, I., Krizhanovsky, A., Pimentel, T., Torroba Hennigen, L., … Hulden, M. (2020). SIGMORPHON 2020 Shared Task 0: Typologically Diverse Morphological Inflection. *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, 1–39. https://doi.org/10.18653/v1/2020.sigmorphon-1.1

Wu, S., & Cotterell, R. (2019). Exact Hard Monotonic Attention for Character-Level Transduction. *ArXiv:1905.06319 [Cs]*. http://arxiv.org/abs/1905.06319

Wu, S., Cotterell, R., & Hulden, M. (2021). Applying the Transformer to Character-level Transduction. *ArXiv:2005.10213 [Cs]*. http://arxiv.org/abs/2005.10213

Appendix

Model Parameters

Hard-Attention (2017 Baseline)	**Transformer (2020 Baseline)**
Input embedding dimensions: 100 Hidden Layers:100 Feature embedding dimensions: 20 Epochs: 50 LSTM layers: 2 Optimization: ADADELTA Regularization: 0 Learning Rate: 0.0001 Epochs: 50 (CPU intensive, no gains noted after 50)	Batch size = 125 Input/Output embedding size (dmodel) = 1024 Embedding dimension: 256 Attention heads: 4 Regularization: 0.03 (Dropout) Encoder/Decoder stack layers (transformer): 4 Epochs: 300+

Augmentation Implementation - Optimal Parameters

Development set	Test set
Dataset size: 2000 Sequence Offset: +/- 1 LCS_min: 3 Compositional/Fixed replacement: Fixed Max_len replacement: 10	Dataset size: 2000 Sequence Offset: 0 LCS_min: 3 Compositional/Fixed replacement: Fixed Max_len replacement: 10

Augmentation Implementation - Optimal Parameters

A selection of parameters where chosen for assessment, initially with a datasize of 2000. Their results are given in table 4.2.3 below:

	Dev		**Test**	
Data set	**Avg. Accuracy**	**Avg. Lev**	**Avg. Accuracy**	**Avg. Lev**
Unaugmented	89.94	0.18	87.17	0.25
Offset 0, comp, LCS-2	88.16	0.26	84.89	0.31
Offset 0, fixed, LCS-2	88.43	0.22	84.48	0.30
Offset 1, comp, LCS-2	88.65	0.24	85.05	0.32
Offset 1, fixed, LCS-2	89.14	0.22	85.77	0.30
Offset 0, comp, LCS-3	89.94	0.20	88.27	0.23
Offset 0, fixed, LCS-3	**91.02**	**0.16**	87.43	0.25
Offset 1, comp, LCS-3	89.45	0.19	88.06	0.24
Offset 1, fixed, LCS-2	90.34	0.20	**88.80**	**0.24**

Table 4.2.3 – Parameter impact on performance