Multi-Task Learning for Sequence-to-Sequence Neural Models of Lemmatization

Lauren Watson

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2018

Abstract

The work presented here extends a sequence-to-sequence neural model of lemmatization to the multi-task learning setting, motivated by both potentially improving the performance of this baseline sequence-to-sequence model as well as by using the task of lemmatization as a setting in which to empirically investigate some proposed effects of multi-task learning. Mutli-task learning was implemented with the auxiliary tasks of auto-encoding and part-of-speech tagging as motivated by both analysis of the original sequence-to-sequence model as well as by related work.

The baseline model for all languages (English, French, Hindi, Turkish, Croatian and Hungarian) was improved by one or both of the auxiliary tasks for both of the dataset sizes investigated. Performance improved by up to 2.25% for models trained using 10,000 training examples and 10.46% for models trained using 1000 training examples.

Examining the behaviour of the multi-task learning model with the auxiliary task of auto-encoding input words results in the conclusion that the main task of lemmatization is not being biased towards predicting outputs which would benefit both the main and auxiliary task (by predicting the input wordform as the output lemma more frequently), which is one suggested explanation for performance gains made by multi-task models. Considering the model with the auxiliary task part-of-speech tagging instead suggests that this auxiliary task provides relevant information to the main task of lemmatization, for example by allowing it to better identify inputs which do not need to be de-inflected such as proper nouns. Aside from these two main findings, the amount of training data is found to have a significant effect on the results as well as the language being lemmatized.

Acknowledgements

I would like to sincerely thank my supervisor, Professor Sharon Goldwater, for her dedication to teaching and excellent feedback and guidance throughout. I am very grateful to have had the opportunity to learn so much about both research and writing up that research from her during the past few months.

I would also like to thank Clara Vania and Sameer Bansal for their insightful comments, discussions, suggested reading and encouragement throughout, as well as Alexander Robertson for his help during the planning of this project. I am also very thankful to both Toms Bergmanis, for generously providing his code as a starting point, and to the contributors of Nematus. The quality of their code made extending it both enjoyable and efficient.

During this project I was lucky enough to be part of a group of incredibly intelligent and hard-working students: Maria Corkery, Jessica Stringham, Shijie Yao, Faheem Kirefu, Sahand Shamal and Bogomil Gospodinov. I both learned a lot and had a lot of fun thanks to you.

Finally, I would like to thank my friends and family for their unwavering support and for reminding me at crucial times that there is life outside of courses and code.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Lauren Watson*)

List of Figures

List of Tables

Glossary of Abbreviations

NLP	Natural Language Processing
MTL	Multi-task Learning
POS	Part-of-speech
BLEU	Bilingual Evaluation Understudy
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit

Table of Contents

Chapter 1

Introduction

Many tasks in Natural Language Processing can be viewed as learning functions to correctly map input sequences to output sequences, for example tasks involving sequences of characters, words or sentences. The task of lemmatization takes as input an inflected word such as *'walking'* and outputs the canonical (dictionary) form *'walk'*. Lemmatization can therefore be viewed as learning to map character sequences to character sequences and has many important applications, including information retrieval (Kanis and Skorkovská, 2010) where correctly mapping the superficially different inflected words of *'brought'* and *'bringing'* to their common lemma *'bring'* can help to identify the relation between *'Was she bringing the cake?'* and *'She brought the cake.'*.

The main difficulty for the task of lemmatization, as with many machine learning tasks, is dealing correctly with previously unseen inputs as this necessitates learning at times complex rules as well as the exceptions to those rules. For example, in the morphologically rich language of Hungarian the plural suffix depends on the vowels present in the word being inflected (Halácsy and Trón, 2006):

$$(Singular) \rightarrow (Plural)$$
$$kar \rightarrow karok$$
$$ber \rightarrow berek$$
$$lufi \rightarrow lufik$$

There is also difficulty with respect to specific types of ambiguity. For example *'painting'* can be used as either a verb or a noun and therefore should be lemmatized to either *'paint'* or *'painting'* respectively. In this case the ambiguity lies in the part-of-speech (POS) tag of the word which has an effect on the resulting correct lemma.

There have been several statistical models of lemmatization proposed (Chrupała, 2006; Chrupała et al., 2008; Müller et al., 2015) and recently Bergmanis and Goldwater (2018) showed improved performance for the lemmatization of over 20 languages with a sequence-to-sequence neural model of lemmatization, Lematus.

Generally, neural networks map input vectors of a fixed size to output vectors of a fixed size. This has limitations for sequence-to-sequence tasks, for example in machine translation where the input *'I want to go'* in English translates to an output sequence of a different length, *'Je veux aller'*, in French. Sequence-to-sequence neural models (Sutskever et al., 2014) instead map variable length input sequences (of fixed size vectors) to variable length output sequences and have been shown to perform well for many sequence-to-sequence tasks including traditionally word-level tasks such as machine translation (Bahdanau et al., 2014) and character level tasks such as lemmatization (Bergmanis and Goldwater, 2018).

Sequence-to-sequence neural models usually consist of an encoder which builds an intermediate representation of the input sequence and a decoder which returns an output sequence given the intermediate representation as input. Despite attaining impressive performance, Lematus still under-performs for input tokens which were either unseen or ambiguous (appeared with more than one lemma) during training, in comparison to those which were seen during training. A promising extension of the single encoder-decoder architecture which may allow these issues to be addressed is extending the model to the framework of multi-task learning (MTL) (Caruna, 1993). Generally, MTL involves training a model to perform more than one prediction task from a given input. For example, one possible implementation would be using a single encoder and then feeding the intermediate representation obtained by this encoder to two or more separate decoders which perform different output tasks. In this case the encoder is therefore influenced by both tasks during training. As discussed by Caruna (1993):

> *'MTL uses the information contained in the training signal of related tasks to bias the learner to hypotheses that benefit multiple tasks.'*

MTL therefore incorporates related tasks which can serve as sources of relevant information for each other during training.

Apart from this intuitive reason in favour of MTL, there has been convincing empirical evidence in favour of extending sequence-to-sequence models to MTL for some sequence tasks such as machine translation. For example Dong et al. (2015) encoded En-

glish sentences which were then translated to Spanish, Dutch, Portuguese and French via separate decoders, obtaining improved BLEU scores for the translation to each target language. However, the reasons for the improvement caused by MTL are often less clear with fewer sources of empirical evidence (Lipton and Steinhardt, 2018). Apart from the motivation of potential improved performance, the task of lemmatization therefore also provides a convenient setting in which to analyse the effects of a specific MTL architecture in an attempt to provide insight into why MTL improves performance.

In the case of extending a model with the main task of lemmatization to a multi-task learning framework, two secondary (auxiliary) tasks appear to be particularly relevant, motivated by both our error analysis of Lematus and also by related work. The first proposed auxiliary task is auto-encoding, whereby the model simply learns to copy the input sequence as the output sequence. Bergmanis et al. (2017) demonstrated the usefulness of auto-encoding as an auxiliary task in a low resource setting for the opposite task of morphological re-inflection, which maps from an input lemma *'walk'* and part-of-speech (POS) tag *'Verb Present Participle'* to the inflected word *'walking'*. Together with the fact that our error analysis of Lematus (Section 3.4) indicated that many incorrect predictions were made when Lematus did not copy the input wordform as the output lemma but should have, either because it de-inflected a wordform which should not have been changed or because its prediction had a few consonants predicted incorrectly, this points towards the auxiliary task of auto-encoding with the hypothesis that increased bias towards copying the input sequence would improve performance. For example for the task of English lemmatization 60% of errors made by Lematus occurred when the input was not copied but should have been.

The second proposed auxiliary task is part-of-speech tagging, which labels words with their relevant tag such as *noun*, *verb*, *adjective* etc... This is primarily motivated by a second pattern which emerged in our error analysis of Lematus, when the model erroneously applied lemmatization rules to nouns. This was sometimes because the word is ambiguous and can either be a verb or a noun, for example incorrectly lemmatizing *'ruling'* to *'rule'*. It also occurred when the noun ended in a sequence commonly used as a suffix, for example lemmatizing *'Mohammed'* to *'Mohamm'*. Using part-of-speech tagging as an auxiliary task is further motivated by the fact that previous non-neural models of lemmatization make use of POS tagging information (Müller et al., 2015; Chrupała et al., 2008).

The first goal of this project was therefore to extend Lematus to the multi-task learning setting with the auxiliary tasks of auto-encoding and POS tagging in order to potentially improve the performance of this model by providing another source of training signal. The overall goal of this work was then to empirically investigate specific questions relating to changes in model behaviour caused by MTL, motivated by obtaining a better understanding of the effects of MTL in this setting:

1. Does MTL with the auxiliary task of auto-encoding bias the model towards returning the input wordform as the output lemma *(copying)*, therefore benefiting languages exhibiting a higher number of lemmas being equal to their inflected wordform?

2. Does the auxiliary task of POS tagging appear to provide information relevant to the main task of lemmatization?

3. Does the MTL framework benefit lower resource models, which use less training data, more than it benefits higher resource models, which use more training data?

4. In cases where both auxiliary tasks separately improved performance, does the use of both auxiliary tasks together further improve performance?

In order to provide as comprehensive an overview as possible given the time available these questions were investigated by observing the effects in both a medium resource setting of 10,000 training examples and a lower resource setting with 1,000 training examples for 6 languages (English, French, Croatian, Hindi, Hungarian and Turkish) which had different proportions of cases where the desired output lemma is identical to the input word, and diverse distributions of POS tags, for example with significantly different proportions of nouns and verbs in comparison to other tags. These languages also include agglutinative languages which commonly combine unchanged morphemes to produce inflected words (Hungarian/Turkish) as well as fusional languages which can have spelling changes when morphemes are combined (English/Hindi/Croatian/French) as this could have an effect on the difficulty of the lemmatization task.

The following conclusions were reached, among others, as supported by the remainder of this document:

- Performance for all languages in both the medium and low resource settings could be improved by extending the baseline sequence-to-sequence model to

one of the auxiliary tasks.

- There is little evidence that the auxiliary task of auto-encoding biases the model towards copying, however it improves performance regardless by around 1% in the medium resource setting and by up to 10% in the low resource setting.
- The auxiliary task of POS tagging improves the model to a larger degree than auto-encoding in the medium resource setting. It improves performance for every language by at least 0.83% and at most 2.25%. For all languages except English and Hindi the largest improvements are for unseen tokens.
- There is evidence that tagging information helps to correct errors in lemmatization which could be helped in theory by knowledge of the part-of-speech tag in the medium resource setting.
- In the low resource setting, POS tagging improves performance for all languages except English, although the multi-task learning models often appeared to be more unstable with higher levels of variance in performance than the baseline models.
- When both individual auxiliary tasks improved performance individually, usually using both auxiliary tasks together in addition to the main task results in slightly higher accuracy than using either task individually. However this is not unanimously true and in some cases it can be detrimental to performance.

Chapter 2

Background

2.1 Lemmatization

Lemmatization is the task of mapping an inflected wordform to its dictionary (base) form. This can be naively done by compiling a dictionary of words with their corresponding lemmas from training data and then using this dictionary to select the most frequently occurring lemma for a given input word. However, building such a dictionary and thus aiming to obtain an exhaustive list of words with their corresponding lemma(s) is both time consuming and expensive. It also has obvious limitations for out-of-vocabulary words, which are an inevitable consequence of ever evolving languages. For example, when testing a child's ability to grasp morphological rules for new nonsense words the 'Wug Test' (Berko, 1958) found that 91% of children tested correctly answered that the plural of one *'wug'* was two *'wugs'* and 90% replied that a man who knew how to *'zib'* was *'zibbing'*. This type of knowledge is not covered by a simple dictionary approach. An obvious next step would be to build templates of lemmatization rules, for example removing *'s'* from the end of inflected nouns in English. However, identifying these rules can become prohibitive, particularly in morphologically rich languages.

2.1.1 Morphology and Lemmatization

The morphology of a given language and the difficulty of the lemmatization task for that language are related. More complex morphology, when morphology is defined in

terms of the number of possible inflected forms of a root wordform (Jeong et al., 2010), where each could pertain to a slightly different meaning of the root form, potentially implies more difficulty in the lemmatization task as evidenced by the fact that Bergmanis and Goldwater (2018) report strong negative correlation between the performance of their system for lemmatization and the proportion of 'unseen' inflected wordforms in a new set of data for the language in comparison to the set that was used to train the model.

For example, consider the following phenomena used to inflect base wordforms for gender, case, time etc... in both English, which is not considered morphologically complex, and Hungarian, which is:

- Suffixes: Adding morpheme(s) to the end of a wordform
  'cat'+'s' → *'cats'*
  'sing'+'ing' → *'singing'*
- Prefixes: Adding morpheme(s) to the beginning of a wordform
  'un'+'happy' → *'unhappy'*
- Circumfixes: Adding two morpheme(s) either side of a wordform
  'leg' + *'nagy'* (big) + *'bb'* → *'legnagyobb'*(biggest) (Dressler and Kiefer, 1990)
- Spelling changes at morpheme boundaries
  'alternate' → *'alternating'*
  'permit' → *'permitting'*
- Irregular Verbs
  *'dr**i**ve'* → *'dr**o**ve'*
  'go' → *'went'*
- Vowel Agreement: When the suffix depends on vowels already present in the word and the other properties of the base wordform (Halácsy and Trón, 2006)
  'kar'+'ok' → *'kar**ok**'*
  'bér'+'ok' → *'bér**ek**'*
  'fa'+'ok' → *'fá**k**'*

Lemmatization is therefore a task which inherently combines memorization in the form of recognizing special cases and irregular forms, as well as the task of learning and applying common rules and patterns governed by the morphology of the language.

2.1.2 Statistical Models of Lemmatization

Note: from here until the end of Section 3.3 some of the content (although not necessarily the wording) is similar to that of the relevant sections in the proposal for this project.[1] *In particular, many of the diagrams used are similar to those of the proposal.*

A significant amount of work has been done in building statistical models of lemmatization. For example the models of Chrupała et al. (2008) and Müller et al. (2015) both approach lemmatization as a supervised classification task which chooses the most probable of a set of pre-defined edit trees mapping an inflected input word to its lemma. Notably, both of these models jointly model the part-of-speech tag and lemma of the input word.

Recently, Bergmanis and Goldwater (2018) proposed a neural model of lemmatization, Lematus, which was adapted from a model of machine translation (Sennrich et al., 2017) discussed in Section 2.2. Lematus outperformed the models of Müller et al. (2015), Chrupała et al. (2008) and Chrupała (2006) as well as a dictionary based approach for 19 of the 20 languages reported. The largest improvements were in terms of accuracy on input wordforms which were not seen during training, although there was also improvement for ambiguous inputs which were seen with more than one candidate lemma during training. Despite this, both unseen and ambiguous accuracies under-performed in comparison to overall accuracies and those of words which were seen during training, leaving room for improvement.

2.2 A Neural Model of Lemmatization: [LN]ematus

2.2.1 Sequence-to-Sequence Models

As outlined in the Introduction, sequence-to-sequence neural models allow neural networks to map between variable length sequences of vectors and are therefore well suited to the task of lemmatization when it is viewed as mapping between sequences of characters. As proposed by Cho et al. (2014b) the variable length input sequence of vectors is *encoded*, using a Recurrent Neural Network (RNN), to a fixed-size intermediate representation which is then *decoded* to a variable length output sequence of

[1] Informatics Project Proposal: *Investigating Multi-Task Learning for Neural Models of Lemmatization* Lauren Watson

vectors, using another RNN. These models are therefore referred to encoder-decoder models, as depicted in Figure 2.1.

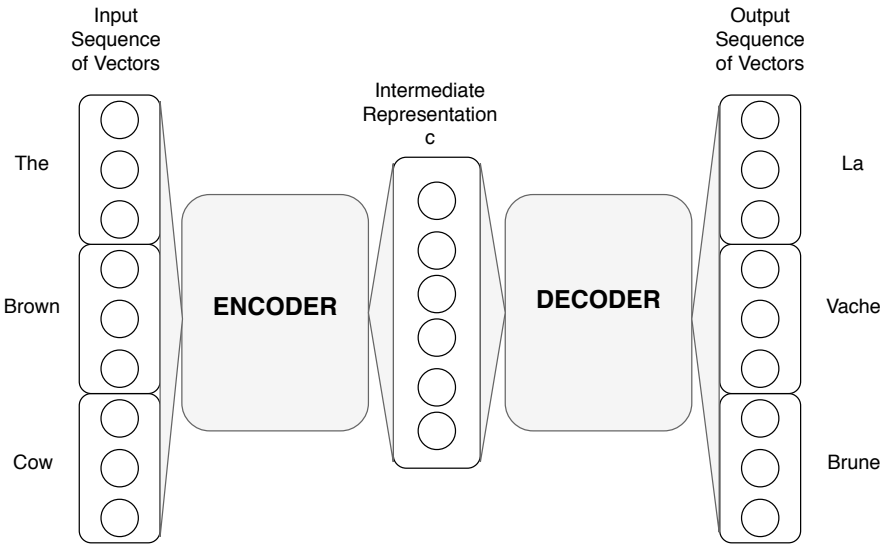


Figure 2.1: Example of a sequence-to-sequence model for machine translation

In examining how the encoder component works, a Recurrent Neural Network takes the input sequence of vectors, $i_1, ...i_T$, and builds the intermediate representation c in a step-by-step manner.

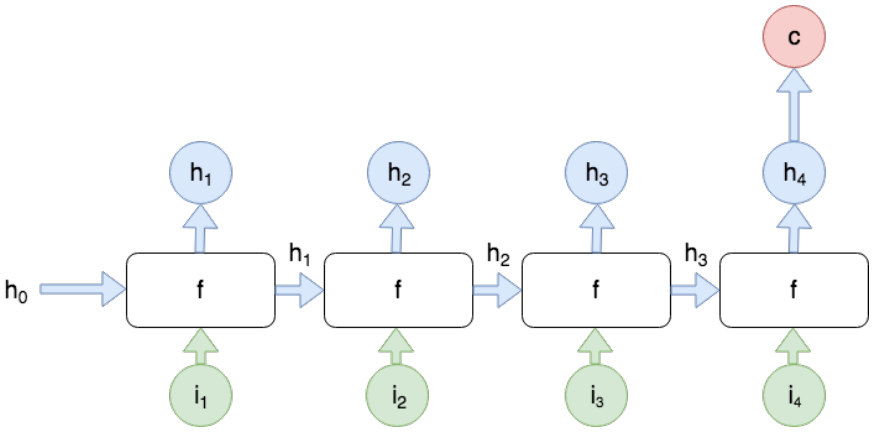


Figure 2.2: Example of an RNN architecture where f represents a so-called 'Activation Function' such as tanh(x)

As shown in Figure 2.2, this is done by building a sequence of hidden states, $h_1, ...h_T$ where each depends on the previous hidden state and the input at that time :

$$h_1 = f(h_0, i_1)$$

$$\mathrm{h}_2 = f(h_1, i_2)$$
$$\vdots$$
$$\mathrm{h}_T = f(h_{T-1}, i_T)$$

The intermediate representation c is then is then often defined as the final hidden state when all inputs have been seen.

$$\mathrm{c} = \mathrm{h}_T$$

A similar model is used to decode this intermediate representation into an output sequence, using the previously output vectors at each timestep.

There have been numerous extensions to this architecture proposed addressing weaknesses found in the performance of these models. For example, Bahdanau et al. (2014) added a mechanism known as *'attention'* which essentially allows the model to learn which part of the input sequence to pay attention to for a given part of the output sequence. For example, in translating the phrase *'The cat and dog'* to French, attention is intended to allow the model to focus on the input *'cat'* when outputting the relevant part of the translation, *'chat'*. Essentially, there is a different intermediate vector c_j used for each output time step j. If the output sequence is $o_1, ...o_J$ then for each output timestep $j = 1...J$ the context vector is defined as:

$$\mathrm{c}_j = \sum_{i=0}^{T} \alpha_{ij} h_i$$

where h_i is the hidden representation obtained as before from the first i inputs. The weights α_{ij} are learned during training and can be interpreted as how heavily to weight the representations obtained for each of the input timesteps $i = 1, ...T$ for the output at time j. This thus addresses the issue of the network potentially 'forgetting' the input sequence by the time it is several timesteps into the output sequence.

Sutskever et al. (2014) proposed a similar architecture to that of Bahdanau et al. (2014) for machine translation which used a variant on the type of neural network used as the encoder and decoder, named a Long Short-Term Memory Network (LSTM). LSTMs add a memory component to RNNs addressing the issue of learning long-term dependencies in sequences by providing access to this memory component as well as the previous hidden state at each time-step which was was found to improve translation quality for English-French translation in comparison to both the models of Cho et al. (2014b) and Bahdanau et al. (2014).

2.2.2 Nematus to Lematus

Sennrich et al. (2017) provided a toolkit implementing these ideas as a framework for machine translation, Nematus, which was essentially the model of Bahdanau et al. (2014) with some extensions. For example, Nematus uses another variant of RNNs which also adds a memory component although in a slightly different way than LSTMs and is based on Gated Recurrent Units (GRUs) (Cho et al., 2014a). Nematus is therefore referred to as an attentional encoder-decoder model.

Some of the numerous alterations made by Nematus to the architecture of Bahdanau et al. (2014) include:

- Initializing the decoder hidden states in a slightly different manner
- Add further regularization to the model to avoid overfitting in the form of recurrent Bayesian dropout (Gal and Ghahramani, 2016)

As a model of machine translation, Nematus mapped between sequences of words, as demonstrated in Figure 2.2. Lematus (Bergmanis and Goldwater, 2018) instead applies Nematus to character sequences, as opposed to sequences of words, as shown in Figure 2.3. Therefore Lematus uses the Nematus model to translate sequences of characters.

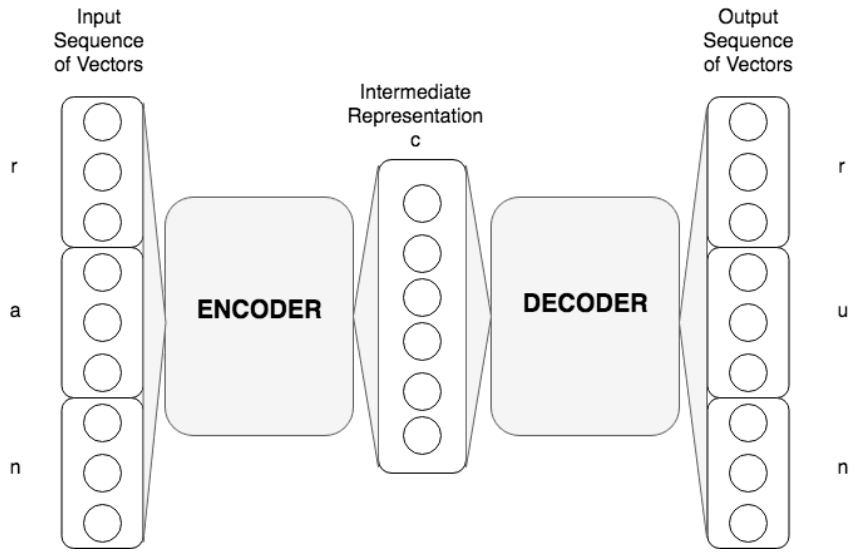


Figure 2.3: Example of a sequence-to-sequence model for machine translation

The input sequence is in the form of the characters of the wordform to be lemmatized with the surrounding characters of the sentence that wordform occurred in, for example

a window of 8 characters either side of the input wordform. The output was the lemma of the wordform. Special characters indicate the beginning and end of the input phrase or output word <w> and <\w> respectively, the start of a new word (<s>), the end of the context characters to the left of the target word (<lc>) and the beginning of the context characters to the right of the target word (<rc>). The characters of the word were separated by a space.

Input: <w> a m <s> n o t <s><lc> k i d d i n g <rc> <s> a n d <s> n o <\w>
Output: < w> k i d <\ w>

Bergmanis and Goldwater (2018) found that a context length of 20 characters had the best performance for lemmatization. Therefore, at test time Lematus requires only input words in context.

2.3 Multi-Task Learning

Multi-task learning (MTL) is a general framework within machine learning introduced by Caruna (1993) which involves performing more than one task using a given model with the motivation of training signals from related tasks helping each other in some way. For example, a common paradigm would be to perform more than one task for a given input, say translating and English input sentence to both French and Spanish output sentences.

2.3.1 A Framework for Sequence-to-Sequence Multi-Task Learning

MTL has been studied for many tasks involving neural networks (Zhang et al., 2014; Wu et al., 2015) and was introduced to Natural Language Processing by Collobert and Weston (2008) who used a convolutional neural network to map input English sentences to POS tags, semantic roles, chunks etc... . Dong et al. (2015) extended MTL to sequence-to-sequence tasks for machine translation, proposing using one encoder and several separate decoders to translate from English into several output languages.

This setting therefore facilitated mapping a single input to several outputs and essentially trained the same encoder but a separate decoder for each desired output task, as shown in Figure 2.4.

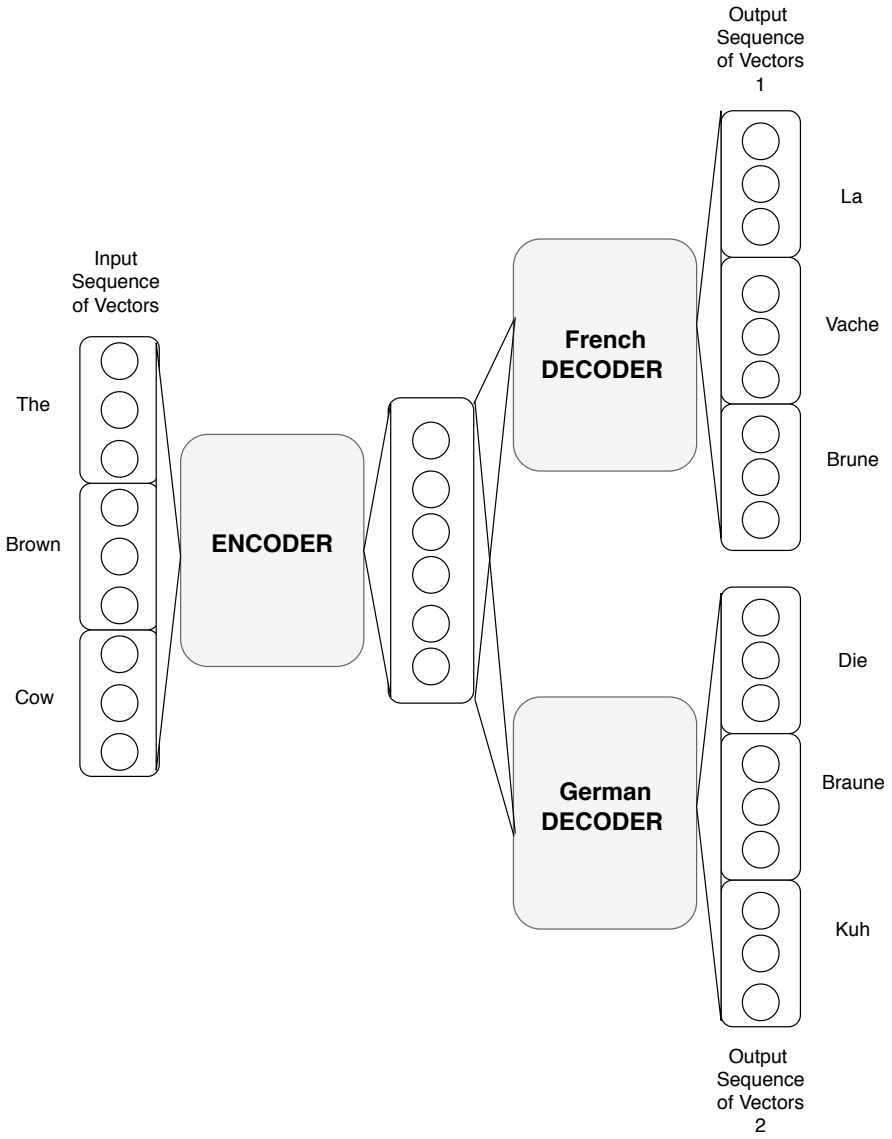


Figure 2.4: Example of a multi-task sequence-to-sequence model for machine translation

Luong et al. (2015) then extended the work of Dong et al. (2015) by examining three potential architectures for extending encoder-decoder models to MTL for sequence-to-sequence tasks. The first architecture was identical to that of Dong et al. (2015) as it involved a single encoder and multiple decoders and was named the *one-to-many* setting. They also examined the *many-to-one* and *many-to-many* settings with multiple decoders and one encoder and multiple encoders and decoders respectively. There are many alternative architectures for MTL, the above methods are often referred to as *hard parameter sharing* as part of the model is explicitly used by all tasks, for example

the encoder in the one-to-many setting. In contrast *soft-parameter sharing* (Liu et al., 2008) is where specific layers of the model are not explicitly shared between models but instead connected in some other way, for example by penalizing the distance between the weights in the models.

2.3.2 Empirical Findings

Both the work of Dong et al. (2015) and Luong et al. (2015) was focused on improving the quality of machine translation using related output tasks. For example, Dong et al. (2015) reported improved performance by between 0.67 and 1.64 BLEU points when this setup was used to translate English input to several related languages, namely Spanish, Dutch, Portuguese and French. Using a similar one-to-many setup, Luong et al. (2015) reported still more improvement over the reported BLEU scores of Dong et al. (2015) using parsing of the Penn Treebank as an auxiliary task, even when this auxiliary task was allowed very little training time in comparison to that of the main translation task.

Although machine translation and the encoder-decoder architectures are the most relevant to this project, there has been a significant amount of work investigating MTL for NLP in recent years. In particular, although the reasons for improved performance remain generally quite unclear from the reported results to date (Alonso and Plank, 2016; Bingel and Søgaard, 2017), there has been recent work attempting to isolate why MTL improves performance when it does. For example, Alonso and Plank (2016) found that while MTL is not always helpful for the main task of semantic sequence labelling, when it is the distribution of classes for the target labels of the auxiliary task is important. For example there is increased improvement for auxiliary tasks such as POS tagging which have quite small numbers of possible labels and these label classes are relatively uniformly distributed with mid-entropy and low kurtosis. Luong et al. (2015) report that a smaller ratio of main task to auxiliary task benefits performance. While impressive findings, these examples pertain to ‘when’ multi-task learning may help performance, not always answering ‘why’ multi-task learning helps in these scenarios. Bingel and Søgaard (2017) find that main tasks which quickly plateau during training benefit from tasks with which do not plateau as quickly, conjecturing that in this case the auxiliary task may stop the main task from becoming stuck in local minima during training and therefore identifying both a ‘when’ and ‘why’.

Apart from this relative shortage of empirically supported explanations for improved performance in comparison to the strength of some of the results in favour of MTL, there are several other conjectured benefits of MTL:

- Training signal from related tasks can bias the model towards choices which benefit both tasks (Caruna, 1993)
- Regularizing the model (Søgaard and Goldberg, 2016)
- Helping the model to identify which features in noisy training data are most relevant (Ruder, 2017)
- There is also the fact that MTL implicitly provides more training data for a given input, even if not all of that data is for the main task there is still more data in total.

2.3.3 Relation to the Research Questions

In light of the empirical evidence discussed in the previous section, the assertion in the Introduction that lemmatization provides a convenient setting in which to investigate some proposed effects of MTL, becomes more clear. The first research question of whether or not the auxiliary task of auto-encoding biases the main task of the model towards predicting the input wordform as the output predicted lemma (thus auto-encoding the input wordform) would shed light on whether in this specific MTL architecture (Section 3.1), Caruna (1993)'s hypothesized effect of MTL biasing the model towards hypotheses that benefit both tasks occurs. This behaviour would potentially improve a specific type of error found in the neural model of lemmatization, Lematus, which occurs when the input wordform should have been copied but was not. Although this is a very simplistic view of the linguistic reasons for this 'copying' behaviour, it is nevertheless a feature of the task which therefore allows this question to be investigated.

The second question of whether errors are decreased by the auxiliary task of part-of-speech tagging is related to another feature of the performance of Lematus, that of the lower performance on ambiguous tokens, for example the many cases where lemmatization rules are mistakenly applied to ambiguous nouns whose wordform could also be used as a verb. It is therefore related to Caruna (1993)'s statement that the auxiliary task may provide training signal for the main task, although in this case in a

more nuanced way by potentially providing information about the part-of-speech of the input.

Both of these research questions thus comprise two parts, whether the behaviour occurs and whether this behaviour leads to the hypothesized increase in accuracy of the model. By proxy, the effect of the ratio of the main task to auxiliary task will also be investigated as well as that of the effect of original dataset size, which forms the third research question. The effect of the distribution of target tags in the auxiliary task will also be observed in this specific setting, however there is not a large variance in the distribution of POS tags between languages in comparison to the variance in distribution of labels between tasks such as POS tagging and Named Entity Recognition and so this is not a main motivating factor of the investigation.

Finally, after the proposal and motivations for this project were formed, Subramanian et al. (2018) reported improved sentence representations using an almost identical one-to-many multi-task learning setup using one encoder with multiple decoders, concluding that future work would be directed towards understanding the specific inductive biases provided by the auxiliary tasks. Thus, this work also extends the work of Subramanian et al. (2018) by investigating some hypothesized biases, although for character level sequences.

Chapter 3

Methodology and Implementation

3.1 Multi-Task Learning with Lematus

The encoder-decoder architecture as implemented by Nematus (Sennrich et al., 2017) and adapted to the task of lemmatization for Lematus (Bergmanis and Goldwater, 2018) was extended to the one-to-many architecture as depicted in Figure 3.1. The implementation of these models is discussed in Section 3.5.

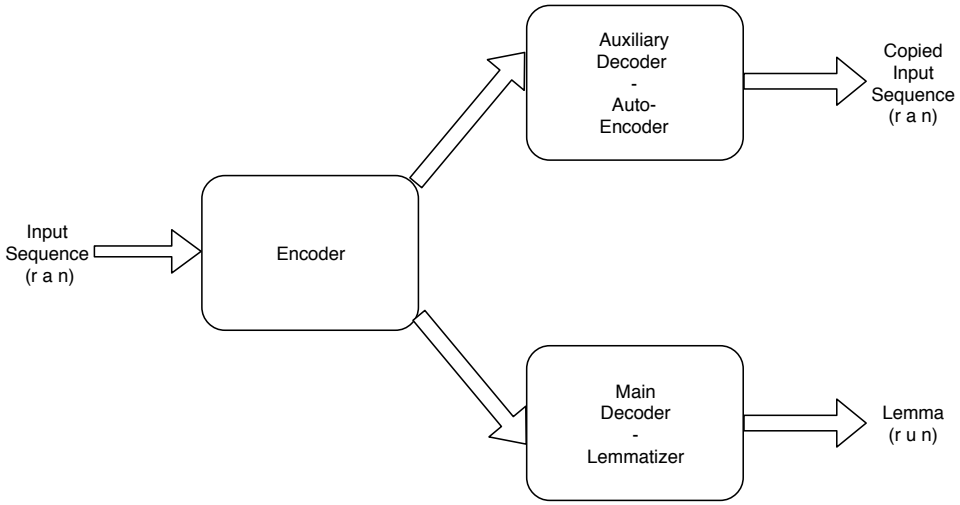


Figure 3.1: Multi-Task Lematus with the auxiliary task of auto-encoding

3.1.1 Training

Using a Tensorflow (Abadi et al., 2016) implementation of this architecture, there are two obvious ways to train the model. The first is to alternate between training decoders for a given batch of inputs at some predefined rate. Therefore for each batch only one task is trained.

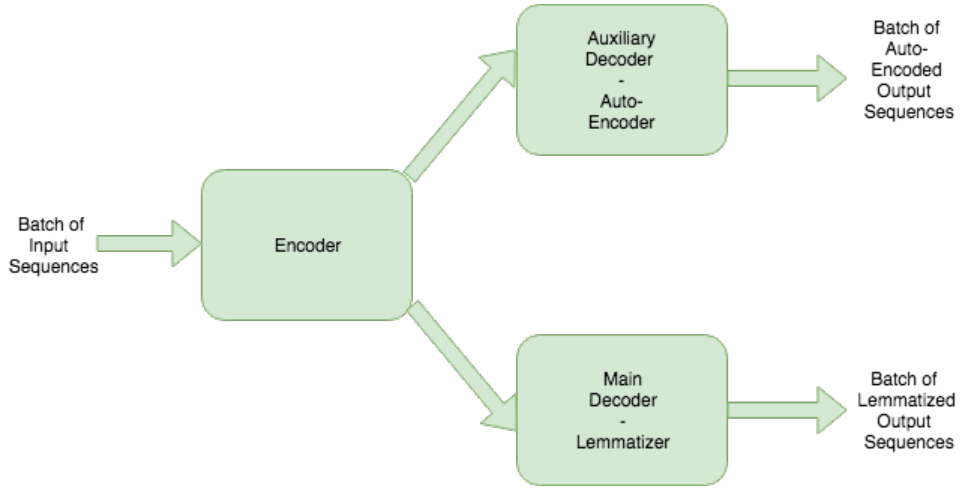


Figure 3.2: Batch of input sequences propogated through the model

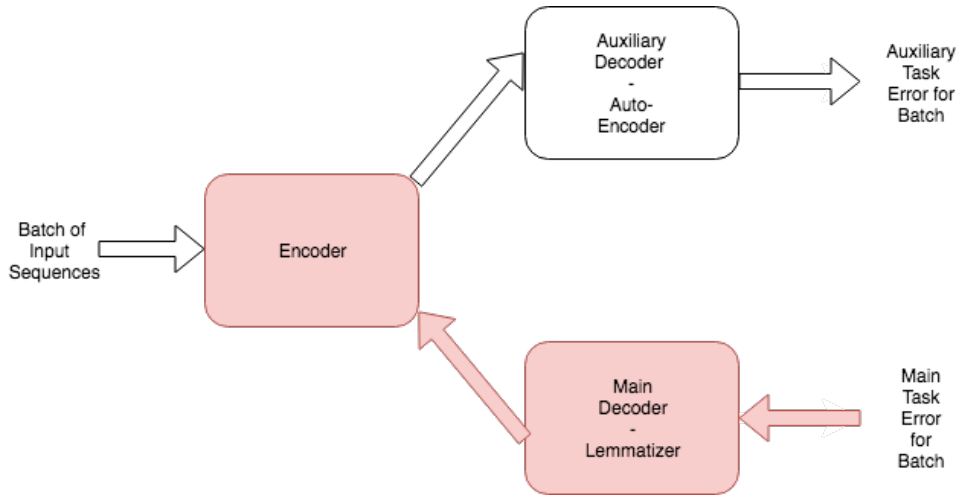


Figure 3.3: Backpropogating error for a batch training the main task

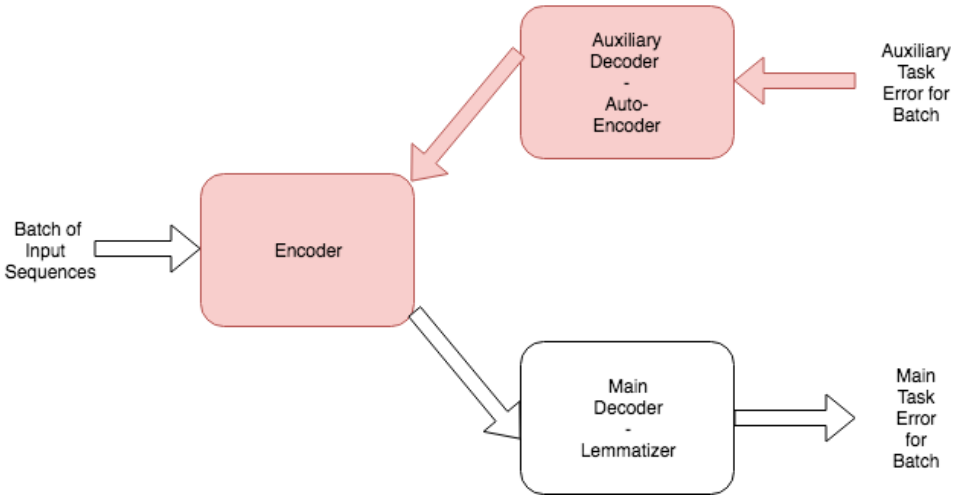


Figure 3.4: Backpropogating error for a batch training the auxiliary task

As outlined in Figure 3.2 a batch of inputs is first propogated through the model. Then the error from either the main or auxiliary decoder for that batch is backpropogated through their respective decoders and the rest of the model as depicted in Figures 3.3 and 3.4 respectively. This method is referred to as *'Alternating Training'* from this point onwards where *'Alternating Training 10:1'* means training with a 10:1 ratio of batches training the main task versus the auxiliary task.

The second method is to add both errors weighted by a factor of $\alpha \in [0,1]$, where $error_{main}$ is the error from the main task and $error_{aux}$ is the error from the auxiliary task:

$$\text{Total Error} = \alpha \cdot error_{main} + (1-\alpha) \cdot error_{aux}$$

This error is then backpropogated through the model, as shown in Figures 3.5and 3.6. This is similar to Alternating Training with the weighted error contributed by the specific parts of the model backpropogated with a 1:1 ratio of main task to auxiliary task.

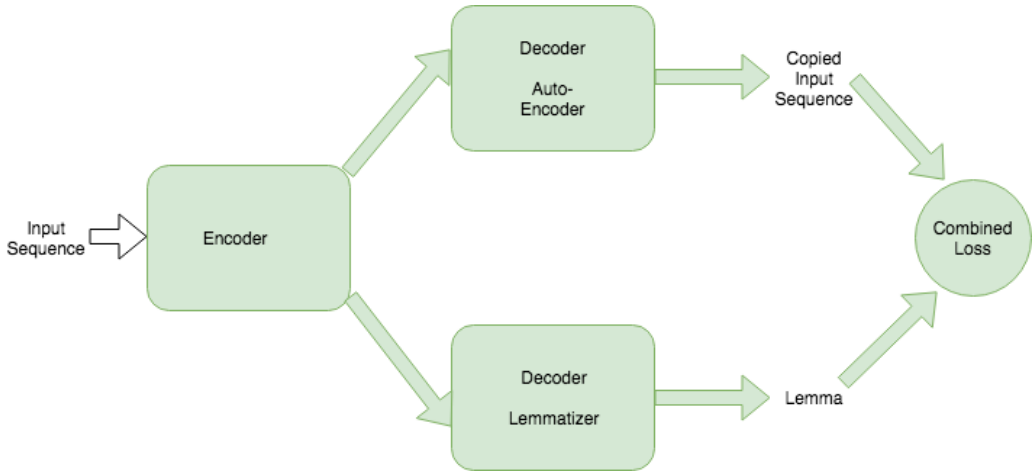


Figure 3.5: Batch of input sequences propogated through the model

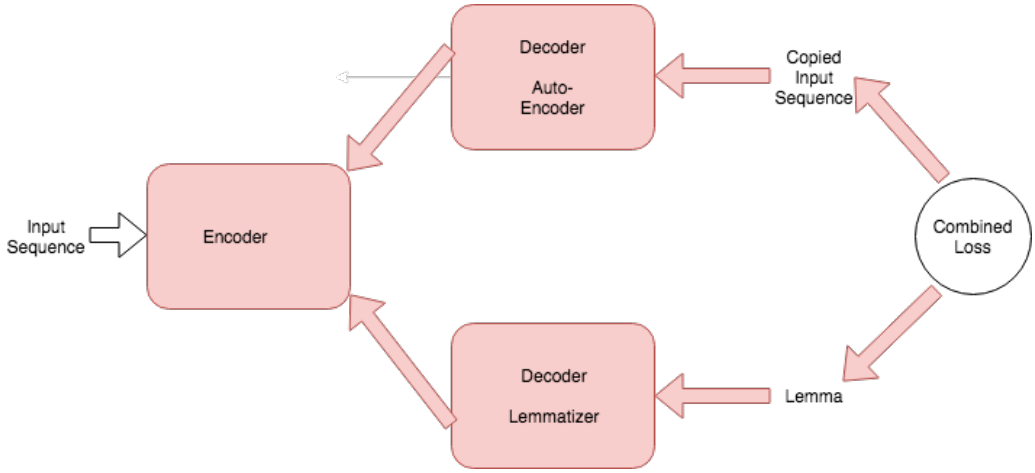


Figure 3.6: Error from batch of input sequences backpropogated through the model

As previous results in favour of MTL have favoured alternating training with smaller ratios of auxiliary task to main task, alternating training is expected to outperform this type of training. Specifically alternating training with a 10:1 ratio of main task to auxiliary task is expected to outperform the other types of training, as this is the training setup examined in this report with the lowest proportion of auxiliary task in comparison to main task training.

3.2 Datasets

The datasets used were those of the Universal Dependencies Project V2 (Nivre et al., 2017) which consist of corpora of treebanks for over 60 languages and provide sentences annotated with both lemmas and POS tags.

Due to time constraints, only 6 languages were chosen for experimentation. In order to make the performance more comparable across languages given the different dataset sizes available for each, the dataset sizes were restricted to 10,000 training tokens, 8,000 validation tokens and 8,000 test tokens. This is referred to as the *'medium resource setting'*. A *'low resource setting'* was also investigated with 1000 training tokens, 800 validation tokens and 800 test tokens. The 6 languages were chosen to vary by 3 properties, as demonstrated in Figures 3.7 and 3.8.

1. The amount of copying behaviour demonstrated in the training data, namely the % of training examples where the input word equalled the desired output lemma. This property is related to the morphological complexity of the language as highly morphologically complex languages generally have many inflected forms of a given base form, with additional suffixes, prefixes and circumfixes denoting case, tense, gender, number, voice markers etc...

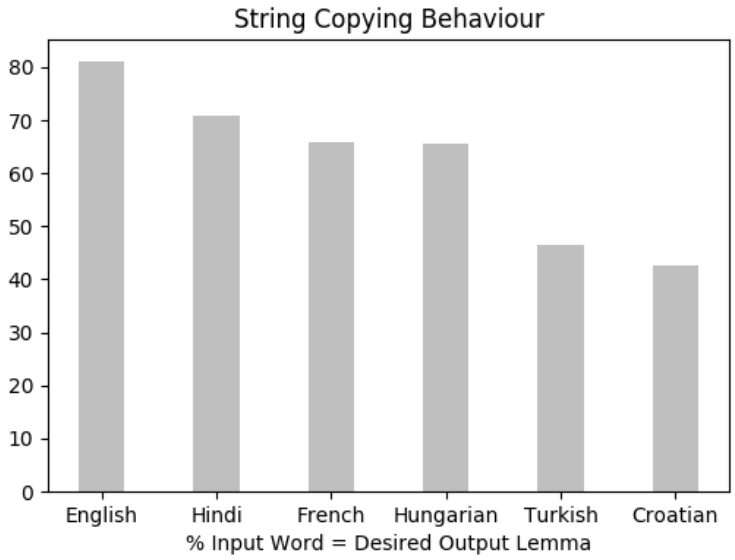


Figure 3.7: Percentage of training cases where the input inflected word is identical to the desired output lemma, calculated from 10,000 training tokens

2. The distribution of Part-of-Speech tags, for example languages such as Hindi which contained almost 20% prepositions/postpositions in comparison to languages such as Hungarian with only 2% of the same.

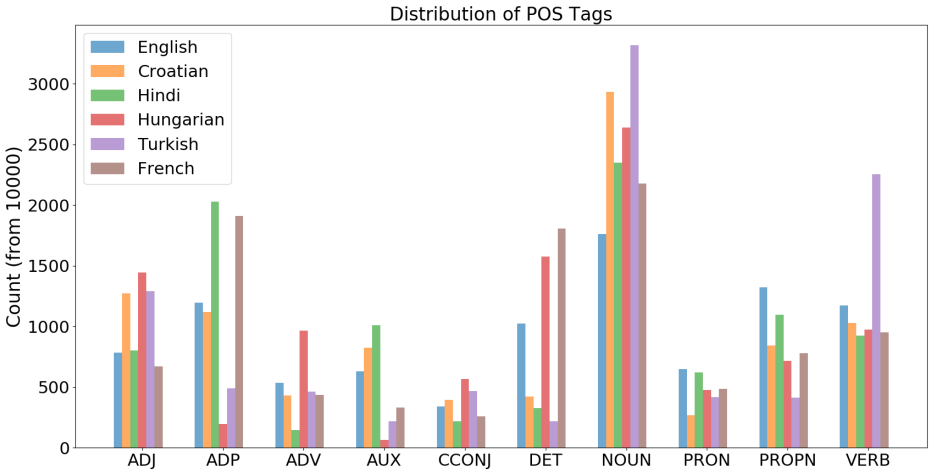


Figure 3.8: ADJ (adjective), ADP (preposition/postposition), ADV (adverb), CCONJ (coordinating conjunction), DET (determiner), NOUN (noun),PRON (pronoun), PROPN (proper noun), VERB (verb)

3. Agglutinative versus fusional languages. Agglutinative languages such as Hungarian and Turkish form words by combining mostly unchanged morphemes whereas fusional languages such as English, French, Hindi and Croatian may have changes at the boundaries of combined morphemes. This may have an effect on the difficulty of the task. For example in the fusional language of English *'changing'* is lemmatized by removing *'ing'* and appending an *'e'* to become *'change'* and similarly *'dating'* is lemmatized to *'date'* whereas *'salting'* is not lemmatized to *'salte'* but instead to *'salt'*.

The reported results are split into overall, unseen and ambiguous token statistics. Unseen input tokens in the validation set are those which were not seen in the training set.

Obtaining representative counts of ambiguous tokens was more challenging than that of unseen tokens. For example, if ambiguity is defined as an input which appears with more than one target lemma in the training data (Bergmanis and Goldwater, 2018), this raises the issue of cases such as *l'* in French which appears with two different lemmas

Language	Types (10k train)	% Unseen Tokens (8k dev)	% Amb. Tokens (8k dev)
English	3994	26.65%	1.24%
French	5465	28.38%	19.93%
Croatian	6263	39.23%	5.16%
Hungarian	7340	43.38%	15.39%
Turkish	8525	42.50%	1.53%
Hindi	3967	31.10%	11.81%

Table 3.1: Statistics calculated using the 10,000 training set tokens and 8,000 validation set tokens. Tokens are input words, types are the set of unique input words e.g. {'Cat', 'Dog', 'Cat'} has 3 tokens and 2 types.

in the training data

$$l' \rightarrow le \quad \text{(225/232 occurrences)}$$
$$l' \rightarrow l' \quad \text{(7/232 occurrences)}$$

These are not inherently different lemmas, they instead represent a potential misunderstanding by an annotator. This is different than inputs such as the French word *fait* which is either an inflected form of the verb *faire* (*to do*) or a noun meaning *a fact* and therefore correctly lemmatized to either *faire* or *fait* respectively.

$$fait \rightarrow faire \quad \text{(14/19 occurrences)}$$
$$fait \rightarrow fait \quad \text{(5/19 occurrences)}$$

The question of if a model can correctly lemmatize ambiguous inputs could therefore be split into two categories: those potentially ambiguous from the model's perspective due to a small percentage of disagreement (or errors) in the training data, and those which are ambiguous with respect to meaning and therefore have more than one correct lemma depending on the context which the input appears in. In total from 1593 ambiguous tokens in the 8000 validation set tokens for French, approximately 1531 (96%) are the first case and 62 are the second case, calculated by manually identifying the number of each case for each type within the ambiguous tokens (there were only 32 types from 1593 tokens. 30% of the tokens were either *la*, *le* or *l'*).

Despite this issue, ambiguous tokens are reported based on those which occur with more than one lemma in the training data regardless of the reason for this occurrence in order to make results comparable to those of Bergmanis and Goldwater (2018).

Where relevant the different cases are discussed.

3.2.1 Pre-Processing

Following Bergmanis and Goldwater (2018), each inflected word was extracted with a context window of 20 characters and markers were added indicating the beginning and end of the input phrase or output word, the start of a new word, the end of the context characters to the left of the target word and the beginning of the context characters to the right of the target word, as discussed in Section 2.2.2. For the auxiliary task of POS tagging the beginning and end of POS tags were marked by <pos> and < \pos> respectively. The characters of the word were separated by a space and examples containing numerical digits 0-9 and hyphens or backslashes were removed. The input was therefore of the format:

<w> t h e <s> l i n k e d <s> a r t i c l e <s> <lc> c o m m i t s <rc>
<s> j u s t <s> a b o u t <s> e v e r y <s> s </w>

with the lemmatization target of:

<w> c o m m i t </w>

and POS tag target of:

<pos> V E R B </pos>

No other pre-processing was done in order to have datasets which were as comparable as possible to that of Bergmanis and Goldwater (2018)

3.3 Evaluation

Following Bergmanis and Goldwater (2018), evaluation is made on exact match lemmatization accuracy where only predicted lemmas which are identical to the target lemma are counted as correct. An alternative measure of accuracy would be using a distance metric such as Levenstein distance however this could conceivably favour models with all predictions close to correct (for example with many predictions including an ‘s’ which should have been removed) over models with mostly perfectly correct lemmas and otherwise significantly incorrect lemmas. It is not clear which would be preferable for the ultimate downstream use of lemmatizers, therefore the less complex measure

of exact match accuracy was used. A potential extension of the analysis included here would be to also analyse the incorrect predictions using a distance metric and take these figures into account as well.

Results are split into overall, seen, unseen and ambiguous tokens. Seen tokens are those input tokens in the validation set which were present in the training set. Similarly, unseen tokens are those input tokens in the validation set which were not present in the training set. As previously discussed, ambiguous tokens are those input tokens which occur with more than one lemma in the training data.

The performance of the model on unseen tokens is of particular interest as the main purpose of models of lemmatization which are not dictionary based is ultimately to deal with out-of-vocabulary words. Ambiguous tokens are seen during training by definition but are also of interest as they more clearly necessitate knowledge beyond that of a dictionary.

3.4 Baseline Models

Three baselines were considered:

1. **Baseline 1**: Copy the input. This is arguably not a model of lemmatization at all, however it gives an indication of how much more a model of lemmatization is learning over this simple rule

2. **Baseline 2**: Most Frequent Lemma. If the input was seen during training, output the most frequently seen lemma. If the input was not seen during training, copy it as the lemma.

3. **Baseline 3**: Lematus (Bergmanis and Goldwater, 2018)

Baselines 1 and 2 are intended as an indication of the difficulty of the task, they provide a ‘sanity check’ as the models take seconds to ‘train’ (e.g. building the dictionary) and run as opposed to the hours needed to train a neural model for the same amount of data. The results of Baselines 1 and 2 are constant, the results of Baseline 2 were averaged over 3 training runs. All other results for neural models reported are also averaged over 3 training runs.

Figures 3.9 to 3.12 show the differences in performance between the baselines.

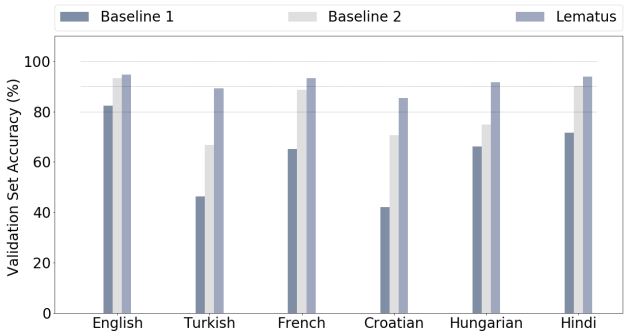


Figure 3.9: Overall Validation Accuracy

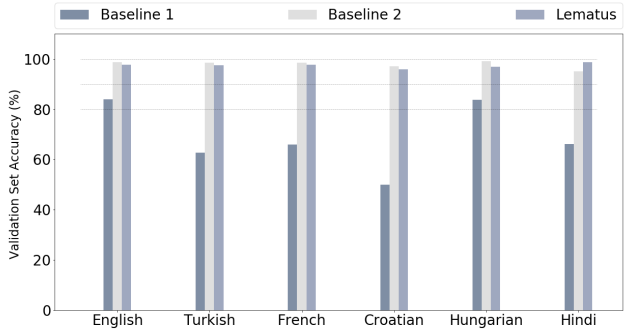


Figure 3.10: Seen Validation Accuracy

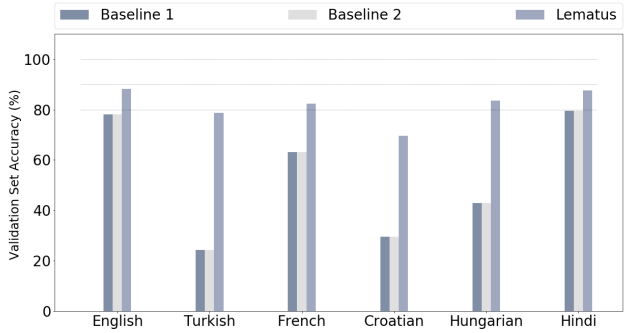


Figure 3.11: Unseen Validation Accuracy

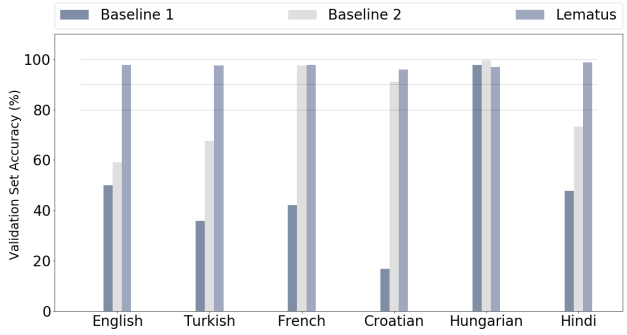


Figure 3.12: Ambiguous Validation Accuracy

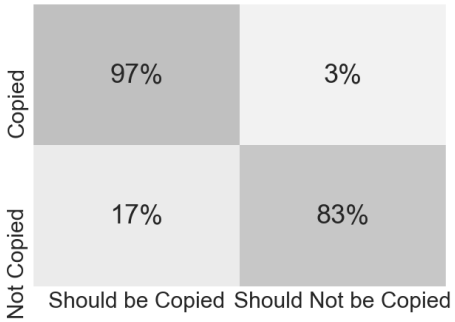
Notably, Baseline 2 performs very well for English, Hindi and French. Most of the performance gain in overall accuracy of Lematus over Baseline 2 is for unseen and ambiguous tokens. Surprisingly, Baseline 2 outperforms Lematus on seen inputs for some languages such as Hungarian.

There was a clear dip in performance across all languages between seen accuracy and unseen accuracy for both Baseline 2 and Lematus. For example, English accuracy for Baseline 2 falls from 98.76% for seen inputs to 78.10% for unseen inputs. For the baseline model of Lematus English accuracy falls from 97.13% for seen inputs to 88.31% for unseen inputs.

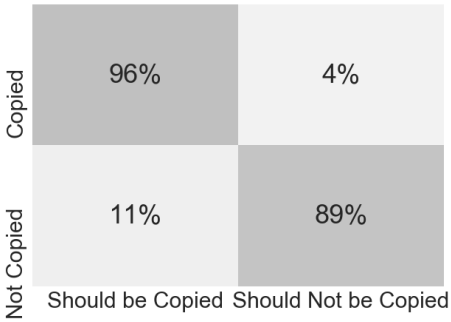
When examining the performance of Lematus more closely, there were some notable patterns:

1. Unseen accuracy is consistently lower than ambiguous, overall or seen accuracy.

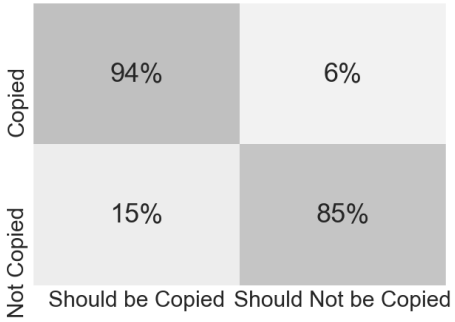
2. Many errors with respect to overall accuracy involved inputs where the lemma should equal the input, as demonstrated in Figures 3.14 and 3.13.



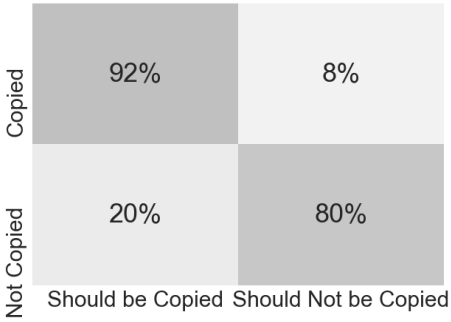
(a) English



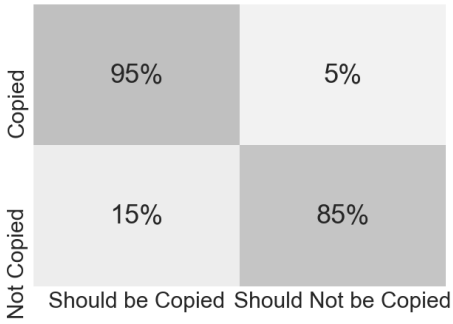
(b) French



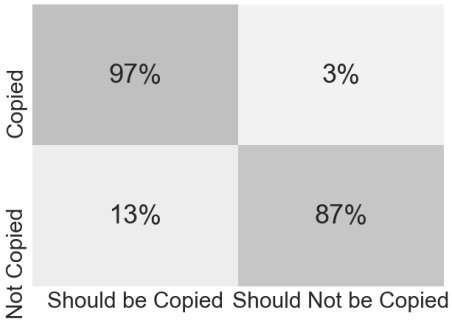
(c) Turkish



(d) Croatian



(e) Hungarian



(f) Hindi

Figure 3.13: Confusion Matrices for Overall Copying Behavior in Lematus

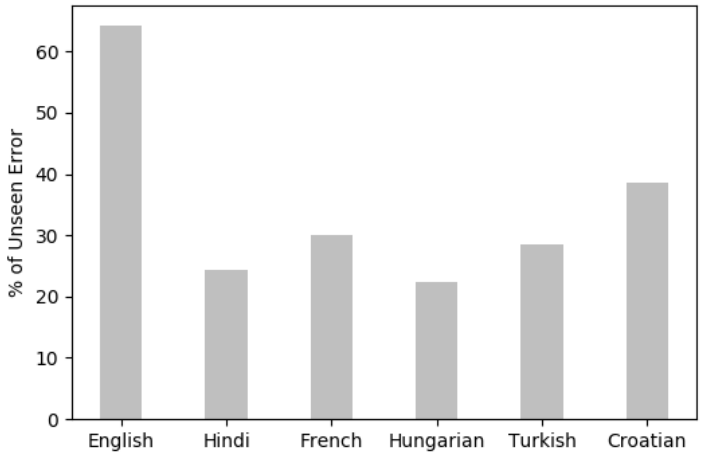


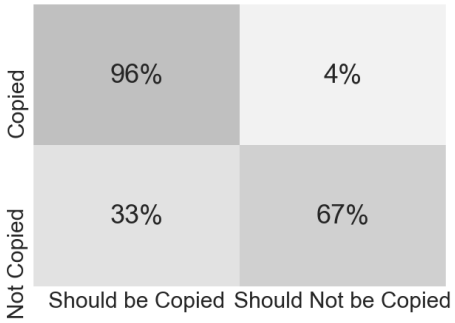
Figure 3.14: Percentage of incorrectly predicted unseen tokens which should have been copied but were not

3. For all languages, except Turkish and Hungarian, there is a higher level of unseen inputs which should have been copied but were not than vice versa (Figure 3.15).

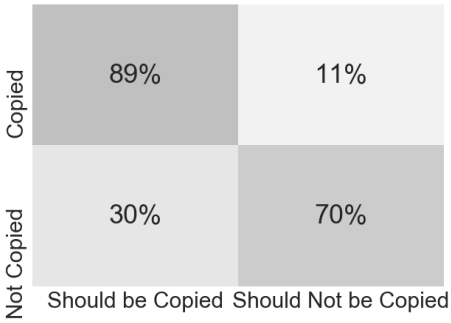
4. For languages such as Turkish, Hungarian, English and Croatian there is a much higher proportion of seen inputs copied than unseen inputs (Table 3.2).

Language	Train Size	Overall Copied	Seen Copied	Unseen Copied
English	**10k**	81.98%	85.13%	73.31%
	1k	79.87%	87.05%	71.32%
Turkish	**10k**	46.80%	63.06%	24.80%
	1k	49.58%	80.47%	37.35%
French	**10k**	66.25%	66.76%	64.99%
	1k	60.17%	67.38%	52.27%
Hungarian	**10k**	67.04%	83.62%	38.24%
	1k	60.71%	92.86%	43.40%
Croatian	**10k**	42.02%	50.76%	28.77%
	1k	35.63%	52.07%	25.44%
Hindi	**10k**	71.44%	68.12%	78.79%
	1k	50.54%	67.66%	30.04%

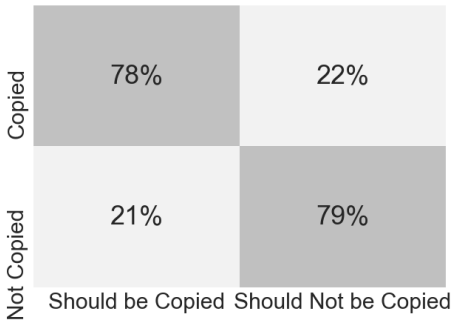
Table 3.2: Copying Behaviour in Baseline Lematus



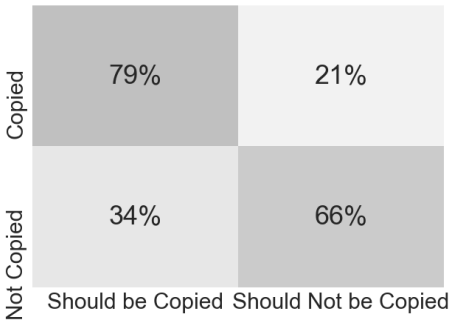
(a) English



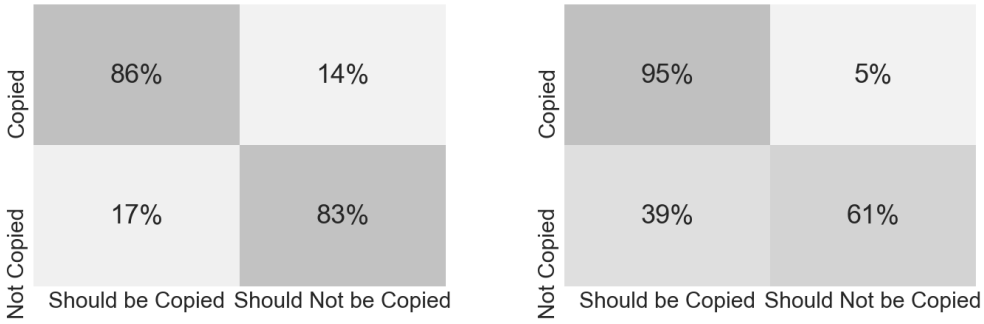
(b) French



(c) Turkish



(d) Croatian



(e) Hungarian (f) Hindi

Figure 3.15: Confusion Matrices for Unseen Copying Behavior in Lematus

5. As previously discussed by Bergmanis and Goldwater (2018), there is a clear relationship between the percentage of unseen tokens for a given language, which is a measure of how morphologically productive that language is (Bergmanis and Goldwater, 2018) and therefore can act as a measure of morphological complexity, and the performance of the Lematus baseline as demonstrated in Figure 3.16. Lematus appears to perform best for languages which are not morphologically complex as measured in this way. Increasing accuracy for unseen tokens would therefore improve performance for morphologically complex languages.

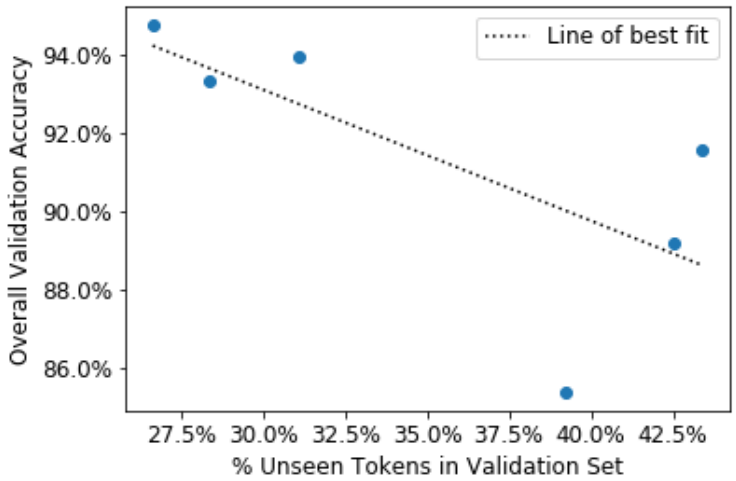


Figure 3.16: Percentage of unseen tokens versus overall validation accuracy

6. Our error analysis of English output for Lematus indicated that common rules for

de-inflecting verbs (such as removing '*ed*') are sometimes erroneously applied to nouns which end in a common suffix. For example, the English baseline Lematus model appears unsure of whether or not to de-inflect *'Mohammed'* to *'Mohamm'*. Between 3 training runs of the Lematus baseline, two models chose *'Mohamm'* while the third output *'Mohammed'*. Knowledge that *'Mohammed'* is a noun, not a verb, would indicate that de-inflecting *'ed'* is inappropriate, or from the perspective of the model, that it is statistically unlikely.

Examining the probabilities associated with these decisions, as shown in Table 3.3, demonstrates that up until *'Mohamm'* the probability for each of the characters output is above 96% for each training run, meaning that the model believes that there is a 96% probability or greater that the character in question is the correct one. After that point the models begin to differ. One model favours continuing to output *'e'* with a probability of 84% while another favours ending the word with probability 86% (or continuing to *'e'* with probability 13%). The third model has a probability of 69% of ending the word and 29% of continuing to *'e'*. Apparently, there is not enough information in the training data for the model to reliably learn what to do in this situation.

Output	**Run 1**
<**w**>	100%
M	97.36%
o	98.97%
h	99.97%
a	99.99%
m	99.89%
m	99.78%
<**w**>	69.05%

Output	**Run 2**
<**w**>	100%
M	96.32%
o	99.61%
h	99.97%
a	99.98%
m	99.94%
m	99.94%
<**w**>	86.55%

Output	**Run 3**
<**w**>	99.99%
M	97.21%
o	99.34%
h	99.93%
a	99.99%
m	99.89%
m	99.89%
e	83.47%
d	99.50%
<**w**>	99.99%

Table 3.3: Probability of outputting the given output symbol for each training run of the baseline Lematus model

7. Examining example outputs of the baseline Lematus model for French verbs, there were two further types of error made. The first was when the verb was

incorrectly de-inflected. For example changing the input *'croyait'* to *'croyer* instead of the correct *'croire'*. Deleting *'ait'* and appending *'er'* is a common transformation between inflected wordform and lemma for regular verbs in French, however *'croire'* is an irregular verb. Some other examples of attempted but ultimately incorrect de-inflections made by the baseline Lematus model were:

'couvrent' → *'couvre'* (instead of *'couvrir'*)
'fallait' → *'faller'* (instead of *'falloir'*)
'suivirent' → *'suivirer'* (instead of *'suivre'*)

8. The other common type of error with respect to verbs would appear to be the inverse of the case discussed for nouns previously, which is when no apparent attempt is made by the model to de-inflect the input at all, perhaps indicating a lack of awareness that the input is a verb. For example:

'donne' → *'donne'* (instead of *'donner'*)
'serrent' → *'serrent'* (instead of *'serrer'*)
'explora' → *'explora'* (instead of *'explorer'*)

This is despite similar transformations being correctly done by the baseline Lematus model for many other verbs, including:

'reste' → *'rester'*
'adressent' → *'adresser'*
'adopta' → *'adopter'*

9. The above are all observations made when the models were trained with 10,000 training tokens, in the low resource setting with only 1000 training tokens the patterns in errors above are less common although there are still some examples of both. Instead, there were some other types of common errors. Firstly the baseline Lematus models make many mistakes with consonants, for example the French baseline model outputs the following:

'film' → *'fily'*
'zone' → *'jone'*
'cot' → *'coyt'*

10. The second type of frequent error in the low resource setting occurs when the model predicts a common lemma seen during training, instead of something related to the input wordform. For example the French baseline model frequently predicts 'être', which is one of the most common verbs in French, for seemingly random input words:

'etape' → *'être'*

'kerr' → *'être'*
'zones' → *'être'*

The observations made about the behaviour of Lematus above therefore empirically motivate both the use of auto-encoding and POS tagging as auxiliary tasks. In the medium resource setting it is possible that auto-encoding could correct some of the errors discussed in points 2, 3 and 4 above, while errors of the types discussed in points 6, 7 and 8 could be helped by knowledge of the POS tag of the input word. In the low resource setting a bias towards auto-encoding the input could help to correct both types of errors discussed.

3.5 Practical Details

3.5.1 Implementation and Experiment Setup

An implementation of Nematus is available via Github (Sennrich et al., 2017) and was used as the basis for this project. The training setup used originally for Lematus was generously provided by the authors (Bergmanis and Goldwater, 2018), although they originally used the Theano (Al-Rfou et al., 2016) implementation of Nematus [1] which is no longer supported. This work extends the current Tensorflow version[2] instead.

Models were trained using NVIDIA 1060 GTX 6GB GPUs and due to the variation in training runs each model was retrained 3 times. It would have been preferable to retrain models more than 3 times, however this was not possible due to the availability of both time and access to shared resources.

There are 3 variations of each of the two training setups (Section 3.1) investigated as it was not immediately clear what an appropriate training setup would be in terms of the ratio of auxiliary to main task training:

- **Alternating 10:1** Alternating training as discussed in Section 3.1.1 with a 10:1 ratio of batches used to train the main task in comparison to the auxiliary task
- **Alternating 2:1** As above with a 2:1 ratio of batches used to train the main task in comparison to the auxiliary task

[1] https://github.com/EdinburghNLP/nematus/tree/theano
[2] https://github.com/EdinburghNLP/nematus

- **Alternating 1:1** As above with and equal split of batches used to train the main task and the auxiliary task
- **Joint 0.2** Joint training as outlined in Section 3.1.1 with $\alpha = 0.2$
- **Joint 0.5** As above with $\alpha = 0.5$
- **Joint 0.8** As above with $\alpha = 0.8$

Each auxiliary task was therefore investigated for 6 languages and two dataset sizes (Section 3.2) using these MTL training settings. This allows the hypothesized effects of the respective auxiliary tasks to be observed for different languages and dataset sizes, therefore giving an indication of to what extent the conclusions are generally applicable across languages and resource levels.

3.5.2 Hyperparameter Settings

The hyperparameter settings used by Lematus were not intentionally altered for any of the models, however some changes were necessitated due to functionality not yet being available in the Tensorflow version at the time of beginning this work:

1. The gradient descent method Adam (Kingma and Ba, 2014) was used instead of AdaDelta (Zeiler, 2012)
2. Validation error was used as the early stopping criteria as opposed to validation accuracy
3. Weight normalization (Salimans and Kingma, 2016) was not supported

As 4 of the 6 languages overlap with the results reported by Bergmanis and Goldwater (2018) and the obtained baseline results using this setup are similar to those results, time was not spent extending the implementation to include these features.

The following hyperparameters were found to have good performance across languages by Bergmanis and Goldwater (2018) and were not changed for any model discussed in this report in order to allow for a direct comparison of the effects of MTL without the results being confused by differences caused by hyperparameter tuning in this way. This follows a similar approach used in related work such as that of Bingel and Søgaard (2017) and Alonso and Plank (2016). Ideally, given enough time and resources, it may have been preferable to instead tune the hyperparameters for every

model and therefore compare each model's best possible performance. This would give a more comprehensive idea of which model would ultimately have the best performance for use in real-world applications however is impractical for this work as it is both time and resource intensive. There would undoubtably be performance gains if the model were tuned further for specific languages.

- Word Embedding Dimension: 300
- Hidden State Dimension: 100
- Layers in Encoder: 2
- Layers in Decoder: 2
- Early Stopping with patience 10
- Batch Size: 60
- Optimizer: Adam
- Learning Rate: 0.0001
- Validation Burn In: 10 epochs
- Validation Frequency: 10 epochs
- Maximum output length: 75 characters
- Dropout (Gal and Ghahramani, 2016) 0.2
- Loss Function: Cross-Entropy Loss
- Activation: Tanh
- Beam Search with a beam size of 12

Chapter 4

Results Part 1: Auto-Encoding and Copying

There are three key questions to be answered by these results:

1. Does MTL with the auxiliary task of auto-encoding improve the performance of Lematus as hypothesized?
2. Does MTL with the auxiliary task of auto-encoding bias the model towards returning the input wordform as the output lemma (*copying*), therefore benefiting languages exhibiting a higher number of lemmas being equal to their inflected wordform?
3. Are these conclusions different in a low-resource setting?

4.1 Accuracy in the Medium Resource Setting

Initially, exact-match accuracy was considered for overall, seen, unseen and ambiguous tokens by averaging the performance of each model over 3 training runs with 10,000 training examples, as discussed in more detail in Chapter 3. As demonstrated in Table 4.4, for each language the best overall and unseen accuracies were obtained by a MTL model, although in some cases by a small margin. In many cases there is a larger improvement in unseen than seen accuracy, for example a 1.57% improvement in unseen accuracy for Hungarian in comparison to a 0.5% improvement in seen accuracy.

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	94.77%	93.34%	85.40%	91.57%	89.21%	93.95%
Best MTL Model	95.78%	93.38%	85.65%	92.38%	90.12%	94.77%
Improvement	+1.01%	+0.04%	+0.25%	+0.81%	+0.91%	+0.82%

Table 4.1: (Auxiliary Task: Auto-Encoding) Average Overall Validation Accuracy figures. The 'Best MTL Model' is the MTL model with the best overall accuracy from the 6 variations of training setup described in Section 3.5.1

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	97.13%	97.67%	95.53%	97.62%	96.96%	96.78%
Best MTL Model	98.54%	97.67%	95.77%	98.19%	97.57%	97.64%
Improvement	+1.41%	-	+0.24%	+0.5%	+0.61%	+0.86%

Table 4.2: (Auxiliary Task: Auto-Encoding) Average Seen Validation Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	88.26%	82.36%	69.69%	83.65%	78.71%	87.64%
Best MTL Model	88.41%	82.64%	70.30%	85.22%	80.04%	88.41%
Improvement	+0.15%	+0.28%	+0.61%	+1.57%	+1.33%	+0.77%

Table 4.3: (Auxiliary Task: Auto-Encoding) Average Unseen Validation Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	63.27%	97.59%	90.94%	99.30%	75.76%	86.93%
Best MTL Model	79.59%	97.91%	91.91%	99.43%	69.42%	89.80%
Improvement	+16.32%	+0.31%	+0.97%	+0.13%	-6.32%	+2.97%

Table 4.4: (Auxiliary Task: Auto-Encoding) Average Ambiguous Validation Accuracy figures

The largest changes for languages such as English and Turkish are for ambiguous accuracies, however the amount of ambiguous tokens in the languages with considerable changes is relatively low. For example English and Turkish both have less than 1.6% (160) ambiguous tokens. In terms of overall accuracy MTL appears to improve performance for English, Hungarian, Turkish and Hindi. However, this result is weakened by the fact that trying 6 variations of MTL, as outlined in Section 3.5.1, in comparison

to one version of the baseline Lematus model is affording a level hyperparamter tuning (of the proportion of auxiliary task training to main task training) to the MTL models which is not offered to the baseline. Considering the accuracies of all of the MTL models in Figure 4.1 gives a more comprehensive review.

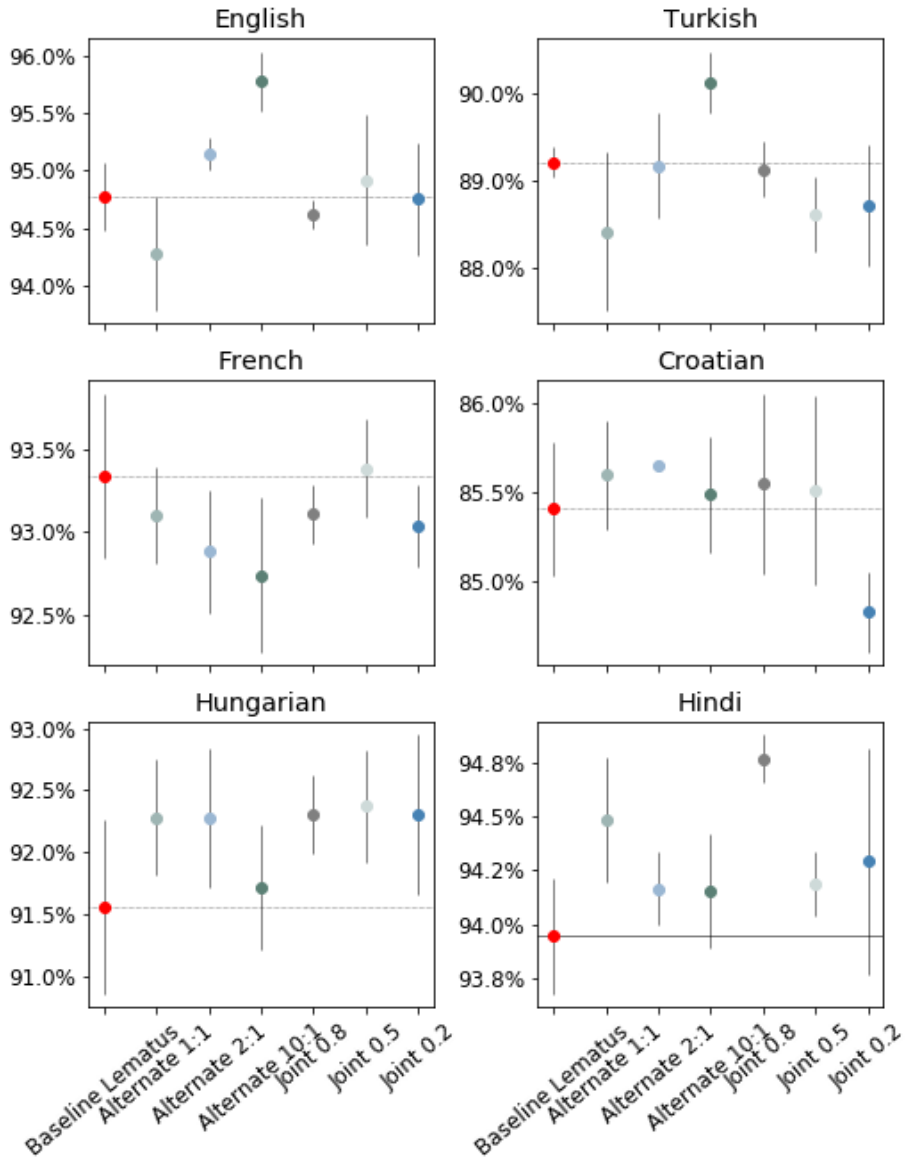


Figure 4.1: Overall Validation accuracies for 10,000 training examples with the auxiliary task of auto-encoding. *Alternate 1:1* represents alternating training with a 1:1 ratio of main task : auxiliary task, as discussed in Chapter 3. Similarly, *Joint 0.50* represents joint training with $\alpha = 0.5$. The vertical lines represent $\pm$ 1 standard deviation.

For Hungarian and Hindi there is a clear improvement in performance. Croatian appears unchanged, particularly when the standard deviations are taken into account. English and Turkish have varied results, with one specific MTL model significantly outperforming all others. In contrast, the French lemmatization model appears to be made worse by the addition of auto-encoding as an auxiliary task. Similar plots for seen and ambiguous tokens are included in Appendix B.

Considering the at times substantial standard deviations in these results, which equate to different outputs for a given input across different training runs of the same model, accuracies were also considered when an ensemble of the three training runs for each model was compared instead of the average performance across those three training runs. Given three training runs for each model, an ensemble output was obtained for a given input as follows:

1. If two or more training runs agreed on an output, then this output was used.
2. If all three training runs disagreed, one of their outputs was chosen at random.
3. The performance for the ensemble for a model was then averaged over 100 runs of that ensemble in order to account for the random element introduced in 2.

The conclusions from these results are largely the same. Appendix C contains an example of the accuracies from the ensemble, however the remainder were excluded for the sake of brevity as the conclusions reached were the same as those of average accuracies for all of the results outlined in this document.

Overall auto-encoding appears to benefit some languages but not all languages, often by margins of around 1% or less. There is no obvious link between the characteristics of these languages as discussed in Chapter 3 and the magnitude of these improvements. For example the fact that French performance did not improve indicates that languages with a higher proportion of lemmas equal to inflected wordforms in the training data (Figure 3.7) do not necessarily improve more. For languages such as Turkish and Hungarian much of this improvement is for unseen tokens while in English, Croatian and Hindi there is a clear improvement for ambiguous tokens. English and Hindi also have improved performance for seen tokens, however this is less impressive when the performance of Baseline 2, which chooses the most frequent lemma seen during training for seen inputs and simply copies unseen inputs, is considered (Chapter 3). The only language which has any neural model of lemmatization which improves performance for seen tokens above that of Baseline 2 is Hindi. Therefore, similarly to the fact that

the main benefit over the baselines for Lematus is with respect to unseen and ambiguous tokens, the valuable improvements in this case over Lematus are with respect to unseen and ambiguous tokens.

4.2 Copying in the Medium Resource Setting

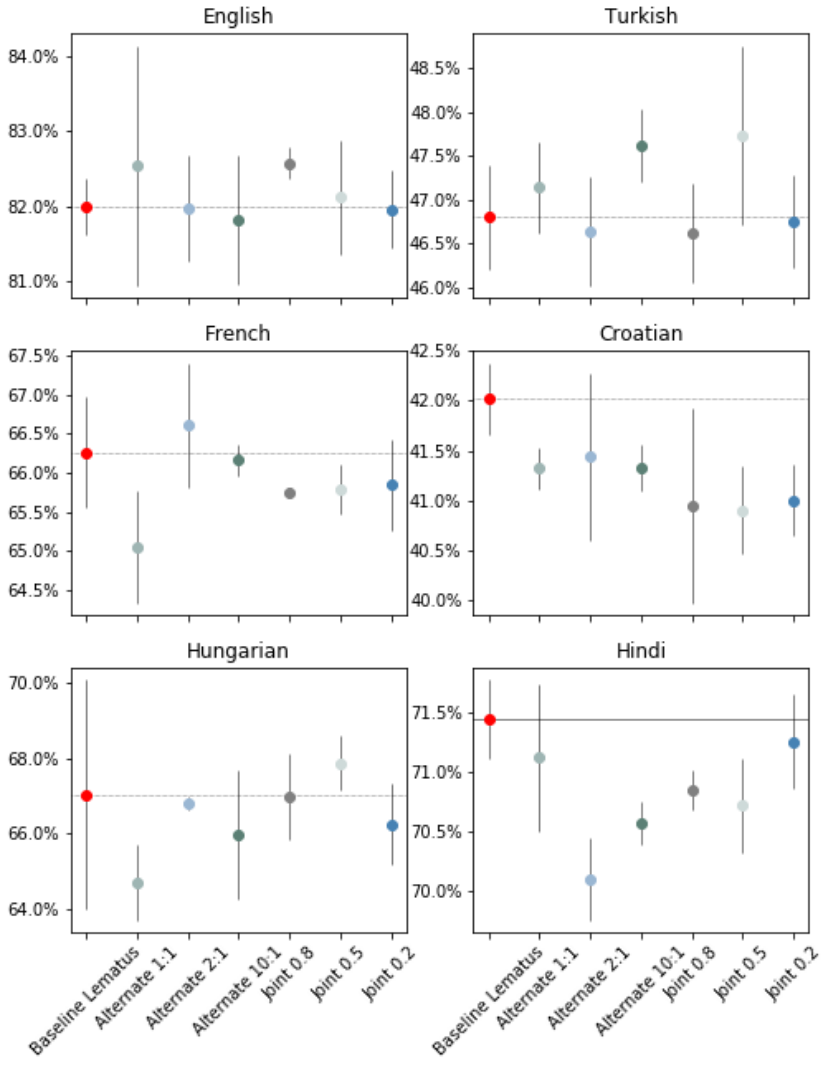


Figure 4.2: Copying Behaviour: Overall % of (input wordform = predicted lemma) with 10,000 training examples for the auxiliary task of auto-encoding. See Figure 4.1 for more details about the plot layout

As discussed previously, Caruna (1993) indicated that MTL may influence models to choose outputs and/or representations which benefited both tasks. In this case does this translate to more input wordforms being copied (with the output predicted lemma being equal to the input wordform)?

The results reported in Figure 4.2 clearly show that the auxiliary task of auto-encoding does not systematically increase the number of input wordforms directly copied as the output lemma. For both Croatian and Hindi there is a clear decrease in copying behaviour. This is also true for unseen tokens (see Appendix B) which in general have a much lower rate of copying than seen tokens across all neural models implemented. For example all Turkish models of lemmatization copy around 66% of seen tokens but only 25% of unseen tokens.

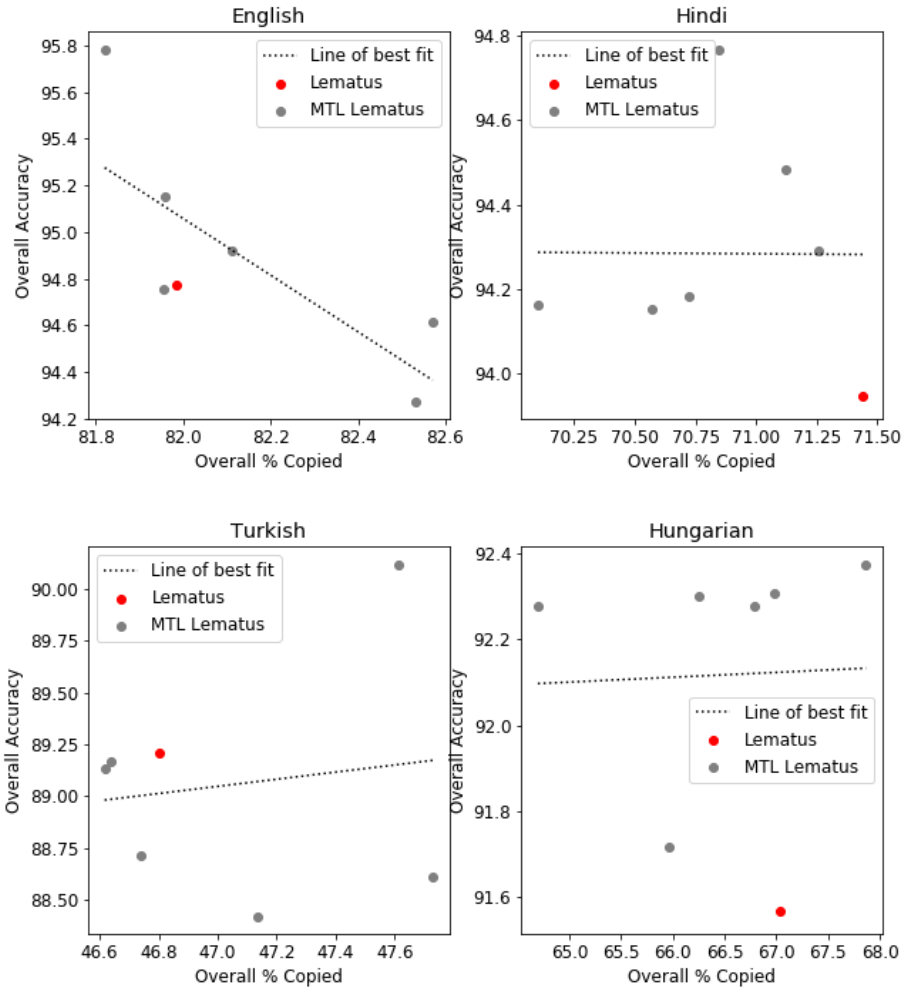


Figure 4.3: Overall Validation accuracies vs % Inputs Words equal to Output Lemma for 10,000 training examples

Instead, as demonstrated for overall tokens in Figure 4.3, there is evidence that the hypothesis that increasing overall string copying in this way might increase accuracy was incorrect for many languages.

4.3 Accuracy in the Low Resource Setting

The low resource setting uses the first 1000 training examples and the first 800 validation examples. Similarly to the medium resource setting, the conclusions varied by language.

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	87.04%	77.25%	61.58%	74.37%	67.42%	64.79%
Best MTL Model	88.79%	77.88%	66.29%	74.04%	73.12%	75.25%
Improvement	+1.75%	+0.63%	+4.71%	-0.33%	+5.7%	+10.46%

Table 4.5: (Auxiliary Task: Auto-Encoding) Average Overall Validation Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	98.54%	97.45%	96.30%	99.40%	92.66%	90.83%
Best MTL Model	98.24%	97.69%	97.82%	99.40%	95.15%	95.49%
Improvement	-0.30%	+0.24%	+1.52%	-	+2.49%	+4.66%

Table 4.6: Auxiliary Task: Auto-Encoding) Average Seen Validation Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	73.33%	55.15%	40.08%	60.90%	57.42%	33.61%
Best MTL Model	77.81%	57.59%	46.76%	60.64%	64.63%	51.01%
Improvement	+4.48%	+2.44%	+6.68%	-0.26%	+7.21%	+17.40%

Table 4.7: Auxiliary Task: Auto-Encoding) Average Unseen Validation Accuracy figures

As shown in Table 4.5 and Figure 4.4, the overall accuracies of Croatian, Turkish and Hindi improved by considerable margins of between 4.7% and 10.5%, with most of this improvement caused by better performance for unseen tokens. The results were

less impressive for English and French, although again the majority of the improvement was in the unseen case for these languages. Hungarian is made worse by the addition of the auxiliary task of auto-encoding in the low resource setting, although this decrease was small. These conclusions were again similar when the training runs were ensembled (Appendix C). No ambiguous figures were included in the low resource setting as the numbers were too low to provide a reliable estimate.

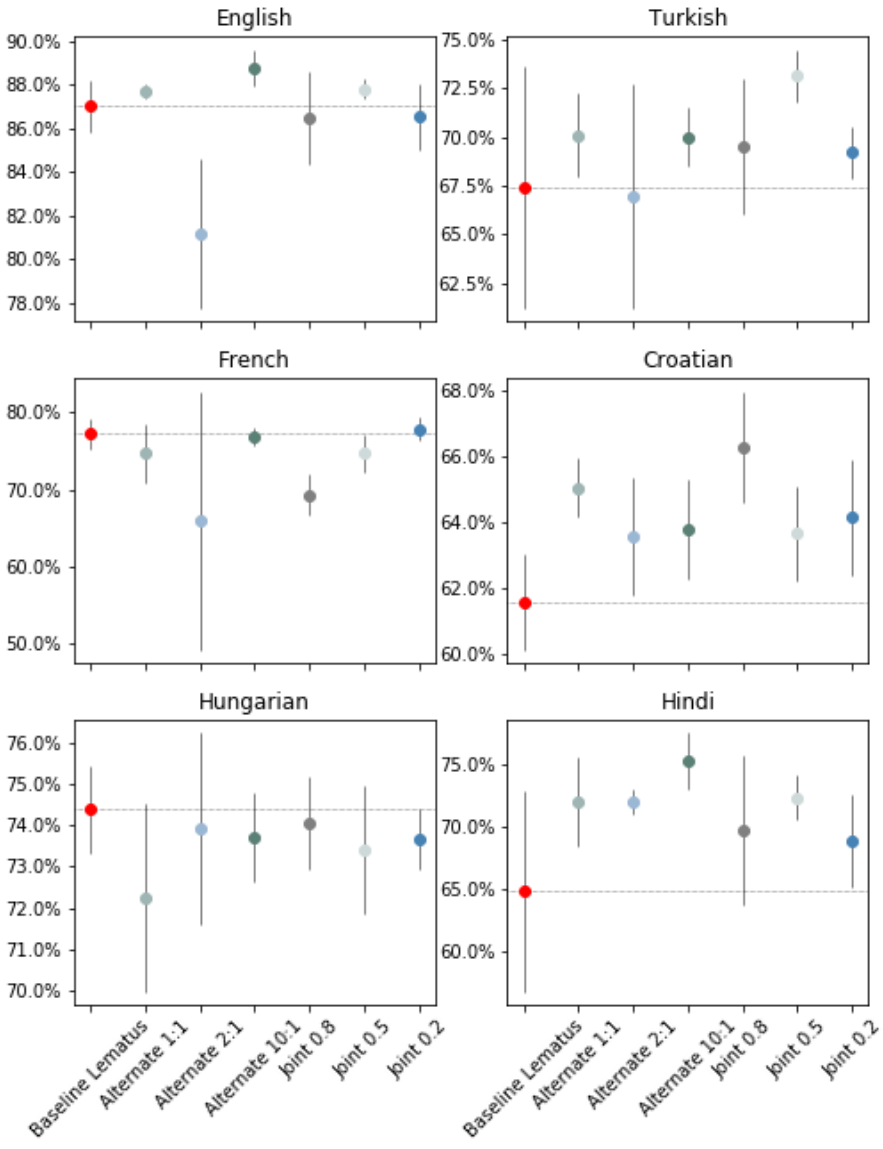


Figure 4.4: Overall Validation accuracies for 1000 training examples with the auxiliary task of auto-encoding. See Figure 4.1 for further details about the plot layout

The improvements obtained in the low resource setting are noticeably larger than that of the medium resource setting. For example, the best MTL model for Hindi in the low resource setting reduced the errors made by the baseline model by 29.7% in comparison to a 13.5% reduction in errors in the medium resource setting. However the standard deviations of the average accuracies for both Turkish and Hindi are quite large, as are those of many other models in this setting, implying that the models are quite unstable. This instability somewhat weakens these results as it could reasonably be the case that many of the changes are due to random differences in performance as opposed to systematic improvement due to some underlying trend.

4.4 Copying in the Low Resource Setting

Except for Hindi, there is no clear evidence of increased copying behaviour in the low resource setting, as demonstrated in Figure 4.5.

In contrast to the medium resource setting, there is a strong relationship between increased copying and improved accuracy scores for some of the languages. As shown in Figure 4.6 for English, French and Hindi more copying implies better performance. However, this is not true to the same extent for Hungarian, Turkish and Croatian and it is unclear which is the causal effect. Is the level of copying increased therefore improving accuracy or does improved accuracy for some other reason necessarily lead to increased copying behaviour in this case?

There was no noticeable effect on the types of errors discussed in Section 3.4 for French or English in the low resource setting. However 87% of the errors corrected by the best Hindi multi-task model involved errors made when the baseline model did not copy the input wordform as the lemma but should have and this was corrected by the MTL model. The same trend was not evident for the other languages most improved by the addition of the auxiliary task of auto-encoding, Turkish and Croatian.

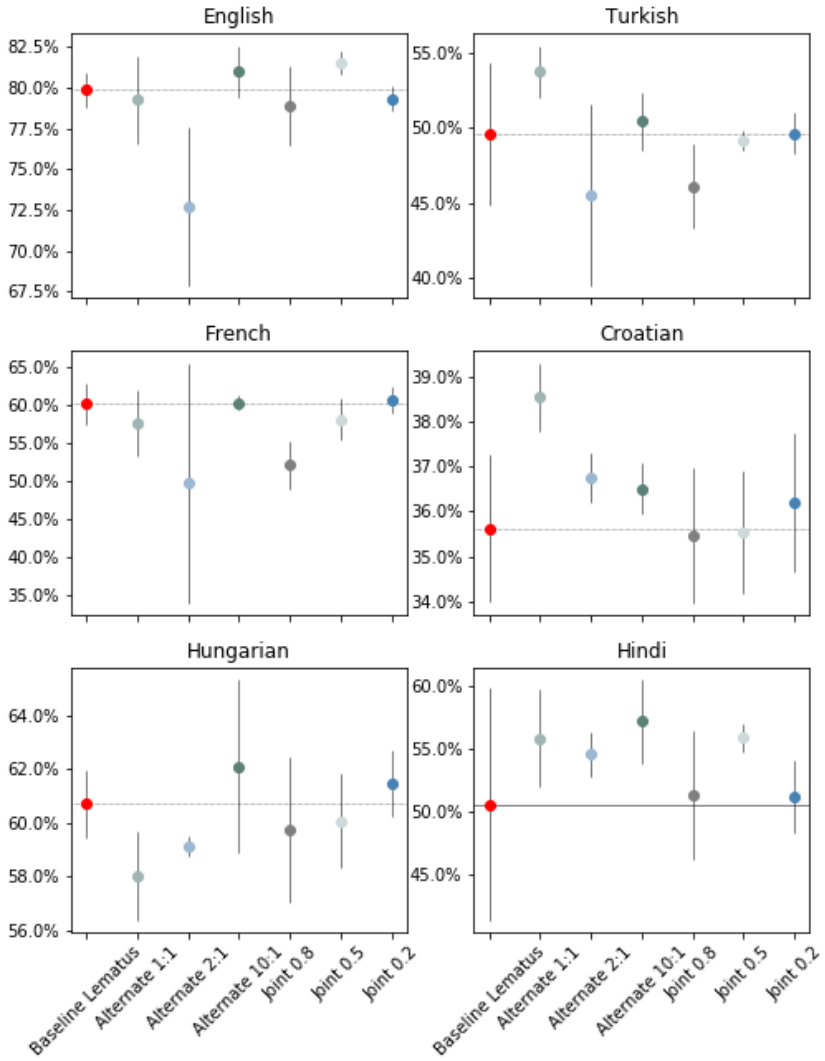


Figure 4.5: Overall % of input wordform = predicted lemma for models trained using 1000 training examples with the auxiliary task of auto-encoding. See Figure 4.1 for further details about this plot layout

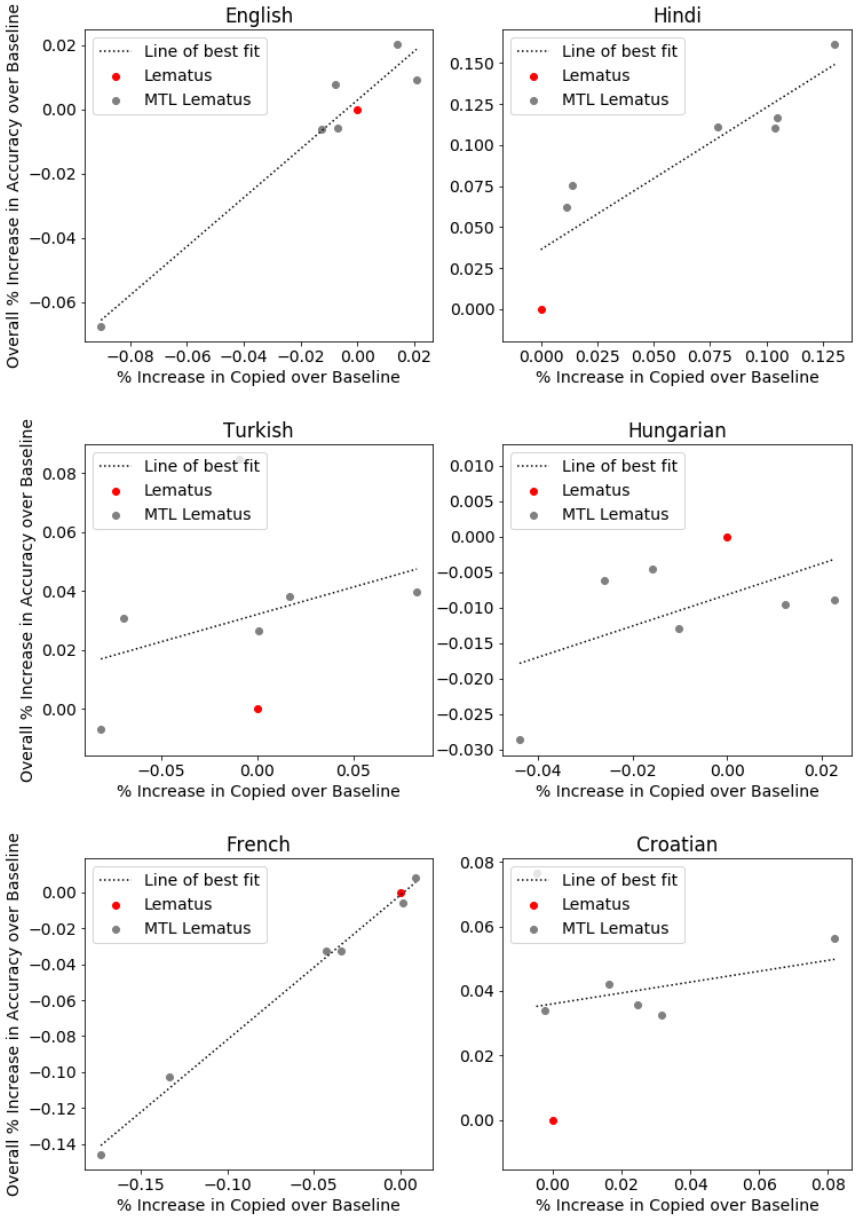


Figure 4.6: Overall % improvement in validation accuracies of the MTL models over the baseline Lematus model versus the % difference in copying with the auxiliary task of auto-encoding

4.5 Research Question Review and Discussion

The following questions were posed at the beginning of this Chapter:

1. *Does MTL with the auxiliary task of auto-encoding improve the performance of*

Lematus?

Whether or not the extension of Lematus to a MTL framework with the auxiliary task of auto-encoding improves performance appears to depend on both the language and size of the training data.

2. *Does MTL with the auxiliary task of auto-encoding bias the model towards copying the input, therefore benefiting languages exhibiting a higher level of lemmas being equal to the inflected wordform more?*

   There is little convincing evidence that the auxiliary task of auto-encoding biases the model towards copying. Specifically, the increases do not appear to be related to the proportion of lemmas equal to their inflected wordform in the training data for the language.

3. *Are these conclusions different in a low-resource setting?*

   These conclusions are also true to an extent in the low resource setting. Although some of the improvements in accuracy in this case are more substantial (up to 10%), there are also significantly higher differences in performance between different training runs for the same model. As could be expected, the low resource models are less stable and therefore the results are somewhat less convincing. For some languages in the low resource setting there is a clear relationship between this improvement in accuracy and increased levels of copying, in direct contrast to the lack of relationship or negative relationship in the medium resource setting. The increased levels of copying were far from unanimous and could either be due to the aforementioned variation in accuracy between training runs in the low resource setting causing a misleading trend, or could also potentially be caused by the different resource settings simply being entirely different problems with respect to model training, with correspondingly different behaviours when the auxiliary task of auto-encoding is added.

What is clear is that regardless of training set size, adding the auxiliary task of auto-encoding does not reliably increase the level of copying, but despite this in many cases it does still increase accuracy. Some further analysis of these results which is not directly related to the research questions is included in Appendix D.

4.6 Test Set Results

To date, conclusions about the auxiliary task of auto-encoding have been made using a validation set which was also used to decide when to stop training each model. The final step is to test these conclusions on a held-out test set, which has been ignored until this point in order to avoid inadvertently tuning the models to this set. The results on this test set are the therefore the best estimate of a each model's ability to generalize to new data. It is important to note in this case accuracy is calculated at the token level. Although every input to the model is different due to the context characters which surround it, many of the actual wordforms to be lemmatized are the same word repeated (such as *'the'* in English). This potentially explains why there is less of a drop between validation and test set accuracy than is often found in other machine learning results.

4.6.1 Medium Resource

In the medium resource setting the conclusions from the validation and test set results were very similar with improvements which range from very small for French of around 0.15% to a more convincing 1.24% and 1.32% for English and Hungarian respectively.

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	94.30%	92.02%	83.65%	88.91%	89.86%	93.95%
Best MTL Model	95.54%	92.17%	84.36%	90.23%	90.94%	94.60%
Improvement	+1.24%	+0.15%	+0.71%	+1.32%	+1.08%	+0.66%

Table 4.8: (Auxiliary Task: Auto-Encoding) Average Overall Test Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	96.70%	96.58%	94.72%	97.51%	96.75%	96.92%
Best MTL Model	98.38%	96.64%	95.31%	98.26%	97.54%	97.70%
Improvement	+1.68%	+0.06%	+0.59%	+0.75%	+0.80%	+0.78%

Table 4.9: (Auxiliary Task: Auto-Encoding) Average Seen Test Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	87.94%	79.19%	68.69%	78.90%	80.59%	86.77%
Best MTL Model	88.52%	80.18%	69.77%	81.60%	82.06%	87.14%
Improvement	+0.58%	+0.99%	+1.08%	+2.70%	+1.47%	+0.37%

Table 4.10: (Auxiliary Task: Auto-Encoding) Average Unseen Test Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	72.22%	95.73%	87.36%	97.95%	75.00%	86.66%
Best MTL Model	80.37%	96.09%	88.65%	98.87%	71.13%	89.38%
Improvement	+8.15%	+0.35%	+1.29%	+0.91%	-3.87%	+2.72%

Table 4.11: (Auxiliary Task: Auto-Encoding) Average Ambiguous Test Accuracy figures

Similarly to the validation set results, French, Croatian, Hungarian and Turkish improved more for unseen than seen tokens and there were some larger improvements in ambiguous tokens, for example 8.15% for English. There was also no noticeable change in copying behaviour for the test set, with the results similar to that of the validation set.

4.6.2 Low Resource

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	85.42%	78.71%	61.67%	72.42%	66.54%	69.29%
Best MTL Model	86.96%	78.92%	66.12%	72.25%	72.50%	78.75%
Improvement	+1.54%	+0.21%	+4.46%	-0.17%	+5.96%	+9.46%

Table 4.12: (Auxiliary Task: Auto-Encoding) Average Overall Test Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	97.49%	95.29%	94.83%	97.61%	91.67%	88.65%
Best MTL Model	97.95%	94.93%	96.09%	98.24%	95.69%	93.22%
Improvement	+0.46%	-0.36%	+1.26%	+0.63%	+4.02%	+4.57%

Table 4.13: (Auxiliary Task: Auto-Encoding) Average Seen Test Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	70.73%	56.27%	42.81%	59.94%	56.28%	47.57%
Best MTL Model	74.15%	57.25%	49.80%	60.00%	63.03%	65.43%
Improvement	+3.42%	+0.98%	+6.99%	+0.06%	+6.75%	+17.86%

Table 4.14: (Auxiliary Task: Auto-Encoding) Average Unseen Test Accuracy figures

In the low resource setting the test set results are again similar to validation set results. Although all improvements except for Turkish are slightly lower, the same languages benefit the most (Hindi, Turkish and Croatian) and the same languages benefit very little or not at all (French and Hungarian).

The trend remains that the most improvement for those languages which did improve is for unseen tokens, although that gap is narrower for Turkish than it was with the validation set. The trends in copying behaviour for the test set in the low resource setting are also very similar to that of the validation set.

Therefore the test set results confirm the findings that for specific languages and training set sizes, the auxiliary task of auto-encoding improves overall accuracy but does not show convincing evidence of increasing the number of inputs copied as the output in general.

Chapter 5

Results Part 2: Part-of-Speech Tagging and Ambiguity

The three key questions to be answered by these results are:

1. Does MTL with the auxiliary task of part-of-speech tagging improve the performance of Lematus?
2. Does MTL with the auxiliary task of POS tagging improve performance for specific POS tags and/or ambiguous tokens therefore indicating increased knowledge of the POS tag of the input wordform by the MTL model in comparison to the baseline Lematus model?
3. Are these conclusions different in a low-resource setting?

5.1 Accuracy in the Medium Resource Setting

As in the previous chapter, the reported results are for exact match validation accuracy. Tables 5.1 to 5.4 demonstrate that in each case, the best MTL model outperformed the baseline Lematus model in terms of overall validation accuracy by margins of between 0.83% and 2.25% and by between 0.39% and 3.34% for unseen accuracy. The type of performance which improved most depended on the language, for example for English and Turkish there was a greater percentage increase in ambiguous accuracy, however this is less impressive when the amount of ambiguous tokens for these languages is considered. For French, Hungarian, Croatian and Turkish the largest percentage im-

provement was for unseen inputs, with Hungarian unseen accuracy improving by an impressive 3.34%.

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	94.77%	93.34%	85.40%	91.57%	89.21%	93.95%
Best MTL Model	96.28%	94.65%	86.38%	93.82%	90.92%	94.78%
Improvement	+1.51%	+1.31%	+0.97%	+2.25%	+1.70%	+0.83%

Table 5.1: (Auxiliary Task: POS Tagging) Average Overall Validation Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	97.13%	97.67%	95.53%	97.62%	96.96%	96.78%
Best MTL Model	99.06%	98.52%	96.39%	99.04%	98.23%	97.88%
Improvement	+1.93%	+0.85%	+0.86%	+1.42%	+1.28%	+1.11%

Table 5.2: (Auxiliary Task: POS Tagging) Average Seen Validation Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	88.31%	82.39%	69.72%	83.67%	78.74%	87.67%
Best MTL Model	88.70%	84.88%	70.86%	87.01%	81.09%	88.65%
Improvement	+0.39%	+2.48%	+1.15%	+3.34%	+2.35%	+0.98%

Table 5.3: (Auxiliary Task: POS Tagging) Average Unseen Validation Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	63.27%	97.59%	90.94%	99.30%	75.76%	86.94%
Best MTL Model	89.80%	97.91%	91.59%	99.49%	76.86%	90.25%
Improvement	+26.53%	+0.31%	+0.65%	+0.19%	+1.10%	+3.32%

Table 5.4: (Auxiliary Task: POS Tagging) Average Ambiguous Validation Accuracy figures

Similarly to the previous chapter, these promising results are somewhat undermined by the fact that a degree of hyperparameter tuning (with respect to the proportion of auxiliary to main task training) was afforded to the MTL models for each language

which was not also available to the baseline. Therefore, Figure 5.2 instead shows the performance for all MTL models for each language in comparison to the baseline.

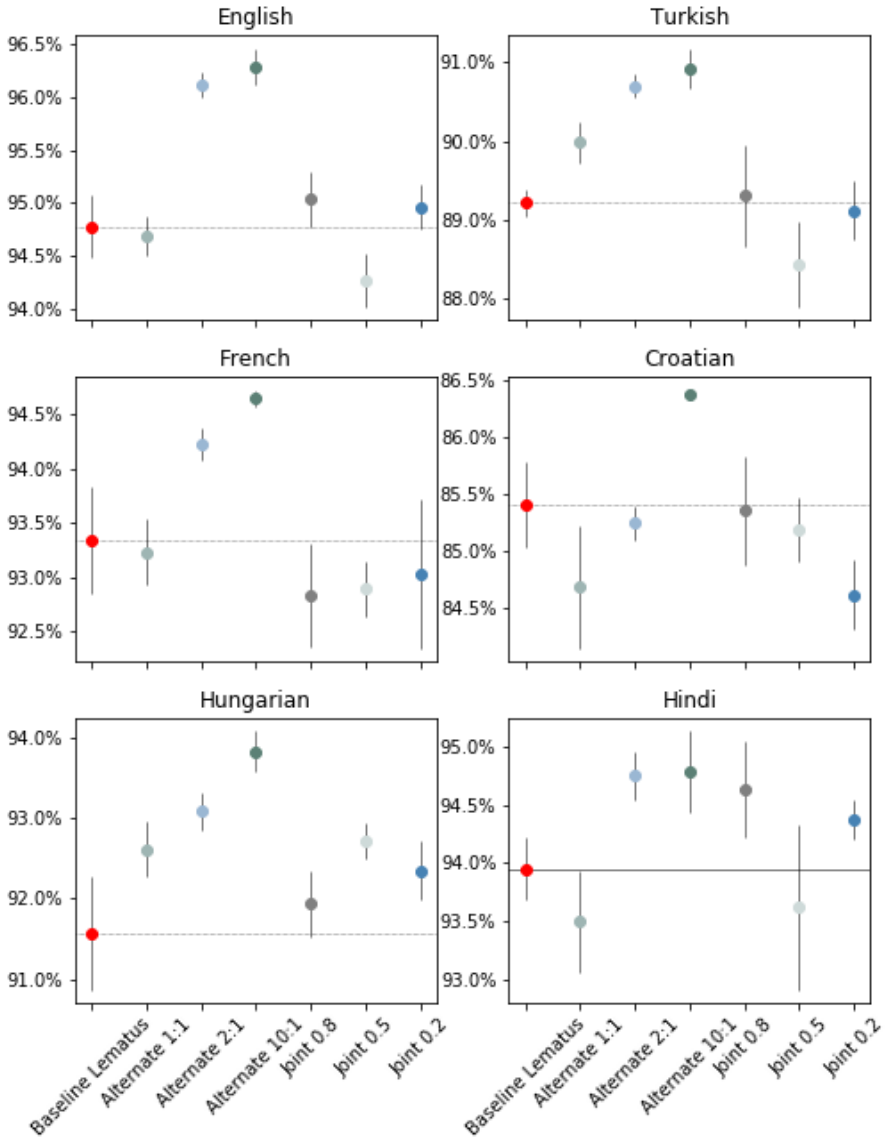


Figure 5.1: Overall validation accuracies for 10000 training examples with the auxiliary task of POS tagging

Interestingly, in this case there is a clear 'best MTL model' with respect to overall accuracy. The MTL model which used alternating training with a 10:1 ratio of training the main task to the auxiliary task was the best performing model for every language.

This result agrees with the findings of Luong et al. (2015) that MTL works best with a small ratio of auxiliary task training, as discussed in Section 2.3. Each joint model appears to have similar performance to the 1:1 alternating training model, which was always the worst performing of the alternating models.

5.2 Ambiguity in the Medium Resource Setting

As discussed previously in Section 3.4, the baseline Lematus model appears to struggle with whether or not *'Mohammed'* should be deinflected to *'Mohamm'*. Between 3 training runs of the Lematus baseline, two models output *'Mohamm'* and the third output *'Mohammed'* with probabilities of between 69% and 86% of either ending the word or continuing from *'Mohamm'* to *'Mohammed'*, as shown in Section 3.4. Does this behaviour change with the addition of the auxiliary task of part-of-speech tagging? Intuitively being aware in some way that *'Mohammed'* was a noun should indicate that it is very unlikely that *'ed'* should be deleted.

Indeed, when the auxiliary task of POS tagging is added, every training run of the Alternating 10:1 MTL model outputs *'Mohammed'*. More concretely, the probability of outputting *'e'* after *'Mohamm'* is now always over 96%, with the next most probable output being to end the word with a probability of only around 3% in each case.

Output	Run 1
<**w**>	100%
M	99.90%
o	99.99%
h	100%
a	100%
m	100%
m	99.99%
e	96.80%
d	100%
<**w**>	100%

Output	Run 2
<**w**>	100%
M	99.87%
o	100%
h	100%
a	100%
m	100%
m	100%
e	96.83%
d	99.99%
<**w**>	100%

Output	Run 3
<**w**>	100%
M	99.74%
o	100%
h	100%
a	100%
m	100%
m	100%
e	96.62%
d	99.97%
<**w**>	100%

Table 5.5: Probability of outputting the given output symbol for each training run of the MTL Lematus model with Alternating 10:1 training and the auxiliary task of POS tagging

If this type of improvement is also reflected across other languages for similar reasons then potentially accuracies for inputs with specific POS tags, such as nouns, will improve for the MTL models in comparison to the baseline model. Observing the breakdown of performance improvement by part-of-speech tag in Figure 5.2 neither proves nor disproves this hypothesized reason for improved performance. For some languages such as Turkish some of the largest improvements are for proper nouns and nouns. However, there is also significant improvements in many languages for both verbs and auxiliary verbs.

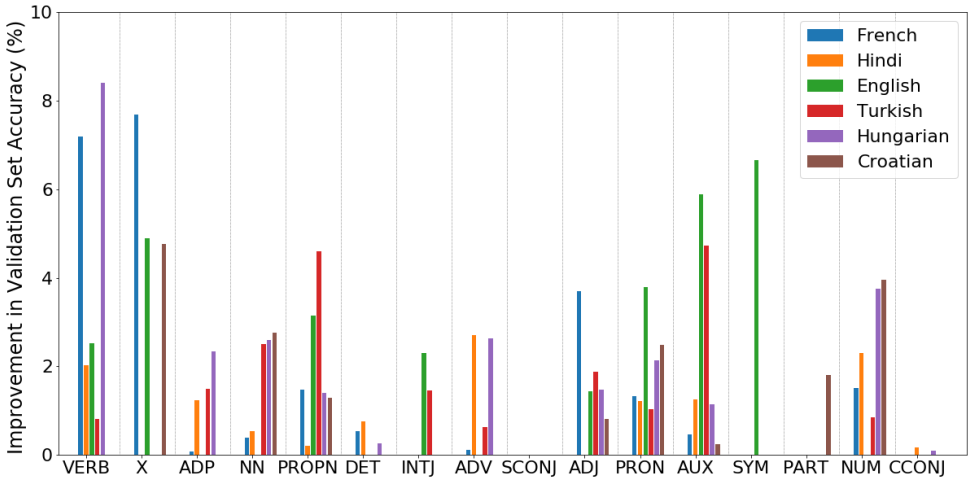


Figure 5.2: (Auxiliary Task: POS Tagging) Overall validation accuracies for 10000 training examples by tag

Another type of error identified in Section 3.4 was that of verbs being incorrectly deinflected, for example *'couvrent'* becoming *'couvre'* instead of *'couvrir'* and*'suivirent'* becoming *'suivirer'* instead of *'suivre'* . The multi-task learning model seems to still make errors, although often an entirely different error, in these cases:

'couvrent' → *'couvenir'*
'fallait' → *'faller'*
'suivirent' → *'suivir'*

In contrast, the second type of error which occurs when the Lematus baseline doesn't attempt to de-inflect verbs, for example mapping *'donne'* to *'donne'* instead of *'donner'*, does seem to be improved by the extension of the model to MTL with the auxiliary task of part-of-speech tagging trained using alternating 10:1 training (Section

3.5.1):

'donne' → *'donner'*
'serrent' → *'serrer'*
'adopta' → *'adopter'*

An estimated 27% of errors for the baseline model for French were the second type of error (when a verb was not de-inflected at all). Of these 45% were corrected by the MTL model. In contrast, around 14% of the errors were of the first type (when a verb was de-inflected to an incorrect de-inflection which would not be helped by knowledge of the part-of-speech tag) and of these only 4% were corrected by the MTL model. For English, an estimated 24% of errors for the baseline model were the second type of error and 38% of these were corrected by the MTL model in comparison to 6% errors of the first type of which only around 5% where corrected by the MTL model. These improvements suggest that some form of knowledge of when to deinflect an input word or not is being provided by the auxiliary task of part-of-speech tagging, potentially in the form of increased knowledge of the tag for each input wordform.

5.3 Accuracy in the Low Resource Setting

In the low resource setting there was an improvement in overall accuracy for every language except English. In each case this improvement is mostly associated with an improvement in accuracy for unseen tokens. Similarly to the low resource results for the auxiliary task of auto-encoding, many of the gains in accuracy in the low resource setting were larger than those of the medium resource setting. Although in the medium resource setting POS tagging unanimously improved overall validation accuracy more than auto-encoding, the same is not true in the low resource setting. The performance of the model for Hindi and Turkish in the low resource setting also improved more with the auxiliary task of auto-encoding than POS tagging.

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	87.04%	77.25%	61.58%	74.37%	67.42%	64.79%
Best MTL Model	86.88%	79.50%	66.92%	75.75%	71.67%	72.46%
Improvement	-0.17%	+2.25%	+5.33%	+1.38%	+4.25%	+7.67%

Table 5.6: (Auxiliary Task: POS Tagging) Average Overall Validation Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	98.54%	97.45%	96.30%	99.40%	92.66%	90.83%
Best MTL Model	98.54%	97.93%	97.82%	99.29%	95.74%	94.50%
Improvement	0.00%	+0.48%	+1.53%	+-0.12%	+3.08%	+3.67%

Table 5.7: (Auxiliary Task: POS Tagging) Average Seen Validation Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	73.33%	55.15%	40.08%	60.90%	57.42%	33.61%
Best MTL Model	72.97%	59.95%	47.91%	63.33%	63.35%	46.06%
Improvement	-0.37%	+4.80%	+7.83%	+2.44%	+5.93%	+12.45%
% Unseen Tokens	45.6%	47.8%	61.8%	65.0%	71.6%	45.5%

Table 5.8: (Auxiliary Task: POS Tagging) Average Unseen Validation Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	100.00%	96.94%	66.67%	0.00%	0.00%	51.39%
Best MTL Model	100.00%	96.60%	100.00%	0.00%	0.00%	63.89%
Improvement	+0.00%	-0.34%	+33.33%	+0.00%	+0.00%	+12.50%
Total Amb. Tokens	6	98	1	0	0	48

Table 5.9: (Auxiliary Task: POS Tagging) Average Ambiguous Validation Accuracy figures

In this setting, there appears to be little relationship between the percentage of unseen tokens and the level of improvement. Similarly there is also not a clear relationship between the percentage of unseen tokens present in the validation set and the overall accuracy of the model despite the percentage of unseen tokens being a measure of the morphological productivity (and therefore morphological complexity) of a language.

Baseline 2, which either outputs the most frequent lemma for previously seen inputs or just copies the input wordform for unseen inputs, outperforms both the Lematus Baseline and the best MTL model in the low resource setting for English, French and Hindi.

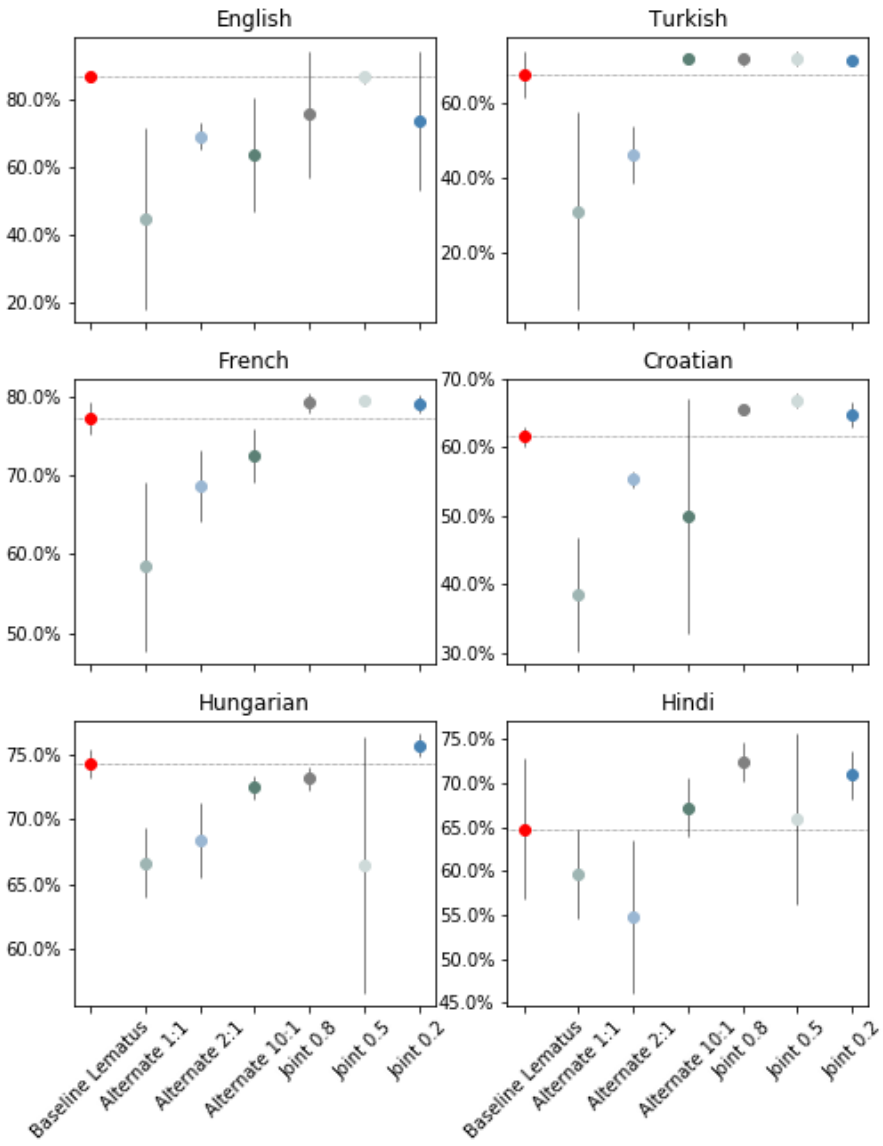


Figure 5.3: Overall validation accuracies for 10000 training examples with the auxiliary task of POS tagging

Additionally, as demonstrated by Figure 5.3, the alternating models with a large proportion of batches used to train the auxiliary task (for example 50% of batches for the 1:1 Alternating model) perform very poorly. Clearly in the low resource setting the wrong auxiliary task and wrong MTL model can cause a much larger dip in performance. For example, the largest reduction in overall accuracy in the low resource

setting for auto-encoding was around 8%. Here, the 1:1 Alternating model at times lowers accuracy by almost 20%. Clearly the dataset size effects the most suitable auxiliary task, as in no case did a MTL model with the auxiliary task of auto-encoding achieve such poor performance.

Interestingly, the same language are most improved by the addition of auto-encoding and part-of-speech tagging in the low resource setting (Hindi, Turkish and Croatian).

As with all other results to date, these conclusions are also evident (with a lower level of variation) when the training runs for each model are ensembled instead of averaged (Appendix E).

5.4 Ambiguity in the Low Resource Setting

It is challenging to observe trends by part-of-speech tag in the low resource setting due to the low numbers of all types of tags, however it is apparent that the trends which were observed in the medium resource setting and discussed in Section 5.2 are less common in the low resource setting. Instead, there were some entirely different types of common errors, as discussed in Section 3.4. Firstly the baseline model makes many mistakes with consonants:

'film' → *'fily'*
'zone' → *'jone'*
'cot' → *'coyt'*

It also frequently predicts 'être' which is one of the most common in French for seemingly random input words:

'etape' → *'être'*
'kerr' → *'être'*
'zones' → *'être'*

There are also errors of a similar kind to the medium resource setting, when no attempt is made to de-inflect a verb or plural noun, despite similar verbs being correctly lemmatized:

'défoncé' → *'défoncé'* (instead of *défoncer*)
'rèvéle' → *'rèvéle'* (instead of *rèvéler*)
'*photos*' → '*photos*' (instead of *photo*)

None of these errors are obviously corrected by the MTL models. Potentially the gains

in this setting for some languages could be due to the large variances in performance between training runs or for another reason which was not obvious to us.

5.5 Research Question Review

1. *Does MTL with the auxiliary task of part-of-speech tagging improve the performance of Lematus?*

   Yes, MTL with the auxiliary task of POS tagging does appear to improve performance over the baseline model of Lematus, particularly with respect to unseen tokens in the medium resource setting.

2. *Does MTL with the auxiliary task of POS tagging improve performance for specific POS tags and/or ambiguous tokens?*

   Yes, in the medium resource setting certain types of error related to knowledge of the POS tag are clearly improved by the addition of the auxiliary task of POS tagging.

3. *Are these conclusions different in a low-resource setting?*

   For all languages except English, POS tagging does also appear to provide performance in the low resource setting. However, the reasons for this improvement are less clear.

   What is clear is that the two resource settings exhibit different issues in the behaviour of the baseline model and also different differences between the baseline model and MTL models. The different resource settings are apparently different problems which therefore behave differently when extended to multi-task learning.

5.6 Test Set Results

As in the previous chapter, conclusions about the auxiliary task of POS tagging until this point have been made on a validation set which was also used to decide when to stop training each model. The final step is to test these conclusions on a held-out test set, which has been ignored until this point in order to avoid inadvertently tuning the

models to this set. They offer the best estimate of how well a model's performance will generalize to new data which has not been used to tune the model.

5.6.1 Medium Resource

In the medium resource setting, the test set results yield similar conclusions to the validation set results previously discussed. There are performance gains with respect to overall accuracy of between 0.79% and 2.52% and gains of between 0.71% and 3.74% for unseen accuracy, confirming the usefulness of the auxiliary task of POS tagging in the medium resource setting for this task.

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	94.30%	92.02%	83.65%	88.91%	89.86%	93.95%
Best MTL Model	96.18%	93.51%	84.85%	91.43%	91.48%	94.73%
Improvement	+1.88%	+1.48%	+1.20%	+2.52%	+1.62%	+0.79%

Table 5.10: (Auxiliary Task: POS Tagging) Average Overall Test Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	96.70%	96.58%	94.72%	97.51%	96.75%	96.92%
Best MTL Model	99.02%	97.49%	95.71%	98.99%	98.03%	97.68%
Improvement	+2.32%	+0.91%	+0.99%	+1.47%	+1.29%	+0.77%

Table 5.11: (Auxiliary Task: POS Tagging) Average Seen Test Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	87.94%	79.19%	68.69%	78.90%	80.59%	86.77%
Best MTL Model	88.66%	82.29%	70.19%	82.64%	83.00%	87.61%
Improvement	+0.71%	+3.10%	+1.50%	+3.74%	+2.40%	+0.84%

Table 5.12: (Auxiliary Task: POS Tagging) Average Unseen Test Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	72.22%	95.73%	87.36%	97.95%	75.00%	86.66%
Best MTL Model	89.63%	96.09%	87.44%	99.17%	76.79%	89.42%
Improvement	+17.41%	+0.35%	+0.08%	+1.22%	+1.79%	+2.76%

Table 5.13: (Auxiliary Task: POS Tagging) Average Ambiguous Test Accuracy figures

5.6.2 Low Resource

Again, in the low resource setting the test set results confirm those of the validation set, with overall and unseen validation accuracy improving for all languages except English. It seems that POS tagging is therefore also a useful auxiliary task in the low resource setting in terms of improving overall accuracy.

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	85.42%	78.71%	61.67%	72.42%	66.54%	69.29%
Best MTL Model	85.38%	79.92%	65.17%	72.87%	71.17%	76.42%
Improvement	-0.04%	+1.21%	+3.50%	+0.46%	+4.62%	+7.12%

Table 5.14: (Auxiliary Task: POS Tagging) Average Overall Test Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	97.49%	95.29%	94.83%	97.61%	91.67%	88.65%
Best MTL Model	97.95%	95.29%	95.63%	98.24%	96.26%	91.73%
Improvement	+0.46%	+0.00%	+0.80%	+0.63%	+4.60%	+3.07%

Table 5.15: (Auxiliary Task: POS Tagging) Average Seen Test Accuracy figures

Model	English	French	Croatian	Hungarian	Turkish	Hindi
Lematus Baseline	70.73%	56.27%	42.81%	59.94%	56.28%	47.57%
Best MTL Model	70.08%	59.31%	48.89%	60.93%	61.74%	60.83%
Improvement	-0.65%	+3.04%	+6.08%	+1.00%	+5.46%	+13.26%

Table 5.16: (Auxiliary Task: POS Tagging) Average Unseen Test Accuracy figures

Chapter 6

Results Part 3: Auto-Encoding, Tagging and their Differences

The final research question posed was as follows:

> *In cases where both auxiliary tasks separately improved performance, does the use of both auxiliary tasks further improve performance?*

There was no clear link between the difference in performance caused for the two separate auxiliary tasks for a given language and resource settings. For some languages, both auxiliary tasks improved accuracy. For others, such as English in the low-resource setting, one auxiliary task improved performance but the other did not. The purpose of this question was therefore to investigate if for languages and resource setting in which both auxiliary tasks increased accuracy, the improvements offered to the baseline by the different tasks could be leveraged together to increase accuracy more than either auxiliary task could alone. If the improvement instead plateaued, this would potentially suggest that some of the benefits offered by the two tasks were the same.

In the medium resource setting Hindi, Hungarian and English were improved by both auxiliary tasks in terms of overall accuracies. In the low resource setting Hindi, Turkish and Croatian were improved by both tasks specifically when joint training was used. The lemmatization model for these languages was therefore trained with both auxiliary tasks and the relevant training type(s), with one encoder as before but 3 decoders corresponding to the main task, auto-encoding and part-of-speech tagging respectively, as briefly outlined in Figure 6.1.

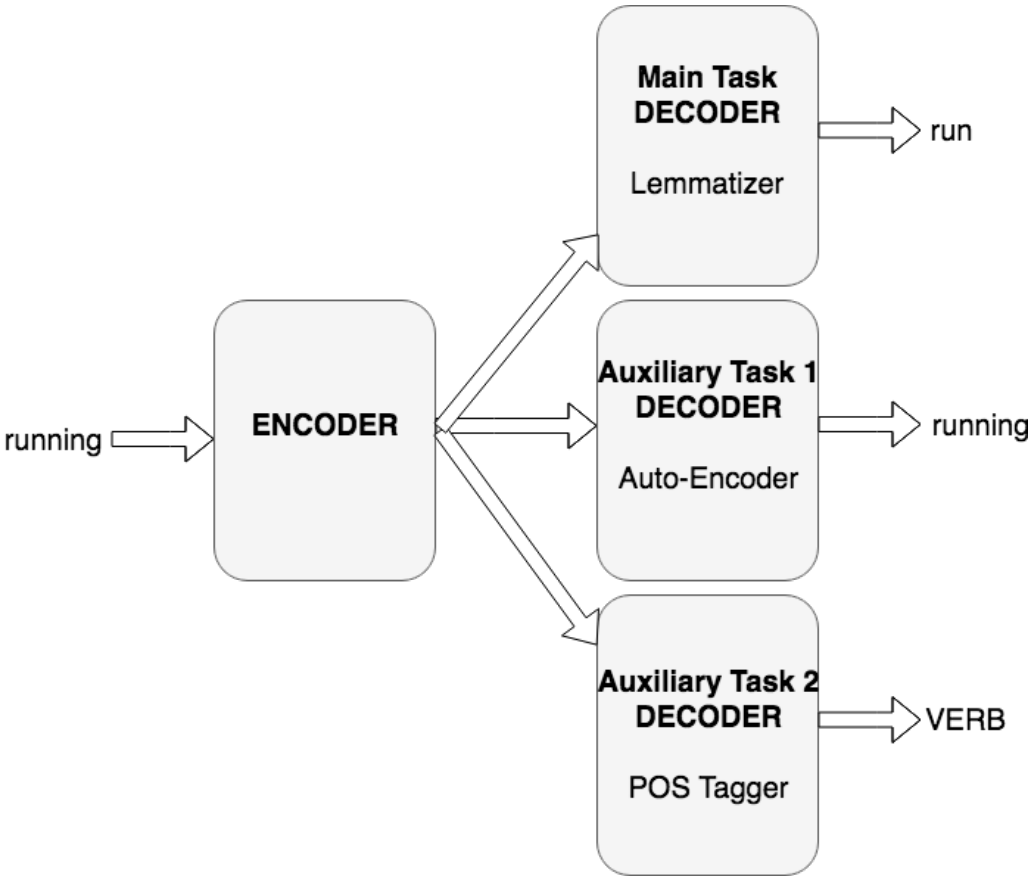


Figure 6.1: Multi-Task Lematus Architecture with both auxiliary tasks

6.1 Accuracy in the Medium Resource Setting

In the medium resource setting, the results using both tasks were either very similar to the best results obtained from single auxiliary tasks for each language or with small increases in accuracy. For example, the best result until this point for Hungarian increased the baseline accuracy by 2.25% using the auxiliary task of part-of-speech tagging. Using both tasks increased baseline accuracy by 2.56%. Similarly English improved by 1.78% in terms of overall accuracy using both auxiliary tasks, in comparison to 1.5% with just part-of-speech tagging. Hindi had very similar performance in both cases, with 0.8% improvement using both auxiliary tasks versus 0.83% with just part-of-speech tagging.

Model	English	Hindi	Hungarian
Lematus Baseline	94.77%	93.95%	91.57%
Best MTL Model	96.55%	94.75%	94.12%
Improvement	+1.78%	+0.80%	+2.56%

Table 6.1: (Auxiliary task: Auto-Encoding + POS Tagging) Average Overall Validation Accuracy figures

Model	English	Hindi	Hungarian
Lematus Baseline	97.13%	96.78%	97.62%
Best MTL Model	99.18%	97.54%	99.02%
Improvement	+2.05%	+0.76%	+1.41%

Table 6.2: (Auxiliary task: Auto-Encoding + POS Tagging) Average Seen Validation Accuracy figures

Model	English	Hindi	Hungarian
Lematus Baseline	88.31%	87.67%	83.67%
Best MTL Model	89.34%	88.83%	87.73%
Improvement	+1.03%	+1.15%	+4.06%

Table 6.3: (Auxiliary task: Auto-Encoding + POS Tagging) Average Unseen Validation Accuracy figures

Model	English	Hindi	Hungarian
Lematus Baseline	63.27%	86.94%	99.30%
Best MTL Model	86.39%	89.58%	99.49%
Improvement	+23.13%	+2.65%	+0.19%

Table 6.4: (Auxiliary task: Auto-Encoding + POS Tagging) Average Ambiguous Validation Accuracy figures

6.2 Accuracy in the Low Resource Setting

In the low resource setting both Hindi and Croatian improve slightly more in comparison to single task accuracy. In contrast, the improvement for Turkish decreases significantly from 5.7% for the auxiliary task of auto-encoding alone or 4.25% for POS tagging alone to 2.25% when both are used.

Model	Hindi	Turkish	Croatian
Lematus Baseline	64.79%	67.42%	61.58%
Best MTL Model	76.17%	69.67%	67.42%
Improvement	+11.38%	+2.25%	+5.83%

Table 6.5: (Auxiliary task: Auto-Encoding + POS Tagging) Average Overall Validation Accuracy figures

Model	Hindi	Turkish	Croatian
Lematus Baseline	90.83%	92.66%	96.30%
Best MTL Model	94.65%	92.22%	97.17%
Improvement	+3.82%	-0.44%	+0.87%

Table 6.6: (Auxiliary task: Auto-Encoding + POS Tagging) Average Seen Validation Accuracy figures

Model	Hindi	Turkish	Croatian
Lematus Baseline	33.61%	57.42%	40.08%
Best MTL Model	54.49%	61.31%	48.99%
Improvement	+20.88%	+3.90%	+8.91%

Table 6.7: (Auxiliary task: Auto-Encoding + POS Tagging) Average Unseen Validation Accuracy figures

For each language the improvement in accuracy for unseen tokens was significantly greater than that of seen tokens, particularly for Hindi and Croatian. These results were similar for the test sets, which are included in Appendix F.

6.3 Research Question Review

In cases where both auxiliary tasks separately improved performance, does the use of both auxiliary tasks further improve performance?

For some languages there was further improvement above that of a single auxiliary task, but not unanimously. When there is further improvement it is often by a relatively small margin. This potentially suggests that some of the same errors were corrected by the addition of each auxiliary task.

Chapter 7

Conclusion

In conclusion, improvements can be made to the performance of a sequence-to-sequence neural model of lemmatization by extending the model to a multi-task learning setting with an appropriate auxiliary task. However, which auxiliary task this is and why it improves the model appears to depend on both the language being lemmatized and the size of the dataset. It remains unclear why auto-encoding improves the performance of the model for languages such as Croatian and Turkish, although this improvement was nevertheless confirmed by the test set results. In contrast, in the medium resource setting there is evidence that information about the part-of-speech tag is reducing the prevalence of errors related to the part-of-speech tag of the input word, such as the model incorrectly attempting to de-inflect nouns or not attempting to de-inflect verbs.

In relation to Caruna (1993)'s previously discussed proposed effect of MTL that *'MTL uses the information contained in the training signal of related tasks to bias the learner to hypotheses that benefit multiple tasks'*, MTL with this particular *one-to-many* model architecture of a single encoder and multiple decoders does not appear to bias the actual output of the main task towards an output that benefits both tasks by increasing the number of outputs which are identical to the input wordform in the case of auto-encoding. Instead, there is evidence that the auxiliary task of part-of-speech tagging is providing a more indirect effect by providing information about the part-of-speech tag in some way. In relation to previous findings that a lower proportion of auxiliary task to main task training is preferable, in the medium resource setting for the auxiliary task of part-of-speech tagging this was confirmed. However the same was not true for the auxiliary task of auto-encoding or for either task in the low resource setting.

In general, the effects of multi-task learning for this model of lemmatization clearly depend on both the language and dataset size. When there are improvements offered by MTL for a given language for both the medium and low resource settings, they are of a larger magnitude for the low resource models. However, it is not always true that the same languages are improved or not improved in the medium and low resource settings for a given auxiliary task. For example POS tagging improves the performance of English in only the medium resource setting and similarly Hungarian is only improved by the auxiliary task of auto-encoding in the medium resource setting. This suggests that for each language the two resource settings are different problems with correspondingly different solutions.

Ultimately, each language and resource setting pair was improved by at least one of the two auxiliary tasks, sometimes by considerable margins. The largest improvements were almost unanimously for unseen tokens although overall unseen and ambiguous tokens still under-perform in comparison to seen tokens. An key strength of using MTL in the way outlined in this work in order to improve performance is that at test time all that is needed is the input words in context. While part-of-speech tags are needed to train the model, they are not input when the model is in use meaning that the models needs only running text as input.

Although this work answered some questions about the conjectured behaviour of MTL models in this setting, many remain. In particular as the hypothesized behaviour was not observed for the auxiliary task of auto-encoding and in the low-resource setting for the auxiliary task of POS tagging, the question remains as to where these improvements originated.

7.1 Research Question Review

The following questions were posed at the beginning of this report:

1. *Does the auxiliary task of auto-encoding bias the model towards copying behavior, therefore benefiting languages with a high proportion of lemmas which are identical to their corresponding inflected wordforms?*

   No, the auxiliary task of auto-encoding does not appear to bias the model towards copying behaviour, however despite this it does improve performance for English, Hungarian, Turkish and Hindi.

2. *Does the auxiliary task of POS tagging appear to provide information relevant to the main task of lemmatization, for example in helping to disambiguate ambiguous words?*

   Yes, the auxiliary task of POS tagging does appear to provide relevant information for the main task of lemmatization as demonstrated in the error analysis of the MTL models in the medium resource setting.

3. *Does the MTL framework benefit lower resource models more than higher resource models?*

   When a given auxiliary task improves performance in both resource settings for a given language, the margin of improvement is higher in the lower resource setting. However the reasons for this improvement are also less clear and there is more variation in performance in the low resource setting. It is also not a given that an auxiliary task which improves performance for a language in the medium resource setting will also improve it in the low resource setting and vice versa.

4. *In cases where both auxiliary tasks separately improved performance, does the use of both auxiliary tasks further improve performance thus potentially implying that they are providing different improvements?*

   Yes, for some languages and resource settings when the individual tasks both improved performance, there is further gains made by using both auxiliary tasks. However, for other languages the use of both auxiliary tasks decreases accuracy.

7.2 Limitations

There were several limitations of the work presented here:

- This work was based on one specific sequence-to-sequence neural model, that of Lematus (Bergmanis and Goldwater, 2018). The effects of MTL as reported here may differ for other sequence-to-sequence models applied to lemmatization, for example models with fewer layers or without an attention mechanism.

- More training runs for each model (for example 5 or 10 instead of 3) would have been preferable had time allowed.

- Training models for more languages would have been preferable had time allowed. Although an attempt was made to select languages with varied properties, this could clearly have been more comprehensively done with 30 languages than 6. This may have revealed more relationships between dataset properties and performance.
- Examining the effects for the full available data set size for each language would provide a clearer picture of the effects of MTL when the full amount of training data is available for a given language
- Our error analysis of the output data for Hindi, Turkish, Hungarian and Croatian would have benefited from input by a native speaker of those languages. Although basic patterns such as incorrect consonants or cutting off the end of words were easily identifiable, other trends may have been immediately obvious to speakers of languages which were not obvious to us.

7.3 Future Directions

Apart from the obvious extensions addressing the limitations listed above, there the following are promising further directions for this work:

- Investigating other training schedules related to alternating training. For example in the medium resource decreasing the proportion of batches used to train the auxiliary task as the main tasks nears convergence may speed up this convergence.
- Investigating pre-training the low-resource models to auto-encode inputs, motivated by the fact that many low-resource models appear to often output the wrong consonant, for example outputting *lone* as the lemma of *zone*.
- Further investigating other proposed effects of MTL in this setting, for example that of regularization, with appropriate experiments.

Appendices

Appendix A

Baseline Results Continued

Baseline 1: Copy the Input Word as the Output Lemma

Language	Training Size	Overall Acc	Seen Acc	Unseen Acc	Ambiguous Acc.
English	**10k**	82.39%	83.95%	78.10%	50%
	1k	83.75%	86.67%	80.27%	100% (from 6)
Turkish	**10k**	46.28%	62.66%	24.31%	35.83%
	1k	49.13%	76.44%	38.43%	0%(from 0)
French	**10k**	65.13%	65.88%	63.23%	42.16%
	1k	67.25%	66.51%	68.06%	62.24%(from 98)
Hungarian	**10k**	66.08%	83.82%	42.98%	97.72%
	1k	61.88%	92.75%	45.61%	0%(from 0)
Croatian	**10k**	41.99%	50.05%	29.50%	16.75%
	1k	39.00%	52.29%	30.77%	0%(from 1)
Hindi	**10k**	71.64%	66.09%	79.50%	47.67%
	1k	65.88%	65.60%	66.21%	52.08%(from 48)

Baseline 2: If input word seen during training, output the most frequent lemma for that input word as output. If unseen during training, copy the input word as the output lemma

Language	Training Size	Overall Acc	Seen Acc	Unseen Acc	Ambiguous Acc.
English	**10k**	93.25%	98.76%	78.10%	59.18%
	1k	90.00%	98.16%	80.27%	0%(from 6)
Turkish	**10k**	66.78%	98.45%	24.31%	67.50%
	1k	55.50%	99.11%	38.43%	0%(from 0)
French	**10k**	88.58%	98.62%	63.23%	97.49%
	1k	84.25%	99.04%	68.07%	95.90%
Hungarian	**10k**	74.79%	99.23%	42.98%	99.51%
	1k	64.38%	100%	45.61%	0%(from 0)
Croatian	**10k**	70.58%	97.10%	29.50%	91.02%
	1k	56.88%	99.02%	30.77%	100%(from 1)
Hindi	**10k**	90.22%	95.05%	79.50%	73.31%
	1k	81.25%	93.81%	66.21%	52.08%(from 48)

Baseline 3: Lematus (Bergmanis and Goldwater, 2018)

Language	Training Size	Overall Acc	Seen Acc	Unseen Acc	Ambiguous Acc.
English	**10k**	94.77%	97.70%	88.26%	97.70%
	1k	87.04%	98.52%	73.13%	98.52%
Turkish	**10k**	89.21%	97.53%	78.71%	97.53%
	1k	67.42%	92.66%	57.32%	92.66%
French	**10k**	93.34%	97.70%	82.36%	97.70%
	1k	77.25%	97.60%	55.00%	97.60%
Hungarian	**10k**	91.57%	96.99%	83.65%	96.99%
	1k	74.38%	99.40%	60.78%	99.40%
Croatian	**10k**	85.40%	95.96%	69.69%	95.96%
	1k	61.58%	96.39%	40.00%	96.39%
Hindi	**10k**	93.95%	98.81%	87.64%	98.81%
	1k	64.79%	95.70%	33.52%	95.70%

Lematus Copying Behaviour

Note the drop in copying for many languages between seen and unseen tokens.

Language	Train Size	Overall Copied	Seen Copied	Unseen Copied	Amb. Copied
English	**10k**	81.98%	85.13%	73.31%	81.29%
	1k	79.87%	87.05%	71.32%	100%(from 6)
Turkish	**10k**	46.80%	63.06%	24.80%	32.23%
	1k	49.58%	80.47%	37.35%	0%(from 0)
French	**10k**	66.25%	66.76%	64.99%	42.77%
	1k	60.17%	67.38%	52.27%	65.31%
Hungarian	**10k**	67.04%	83.62%	38.24%	98.13%
	1k	60.71%	92.86%	43.40%	0%(from 0)
Croatian	**10k**	42.02%	50.76%	28.77%	17.72%
	1k	35.63%	52.07%	25.44%	33%(from 1)
Hindi	**10k**	71.44%	68.12%	78.79%	46.50%
	1k	50.54%	67.66%	30.04%	72.92%

Appendix B

Further Auto-Encoding Results

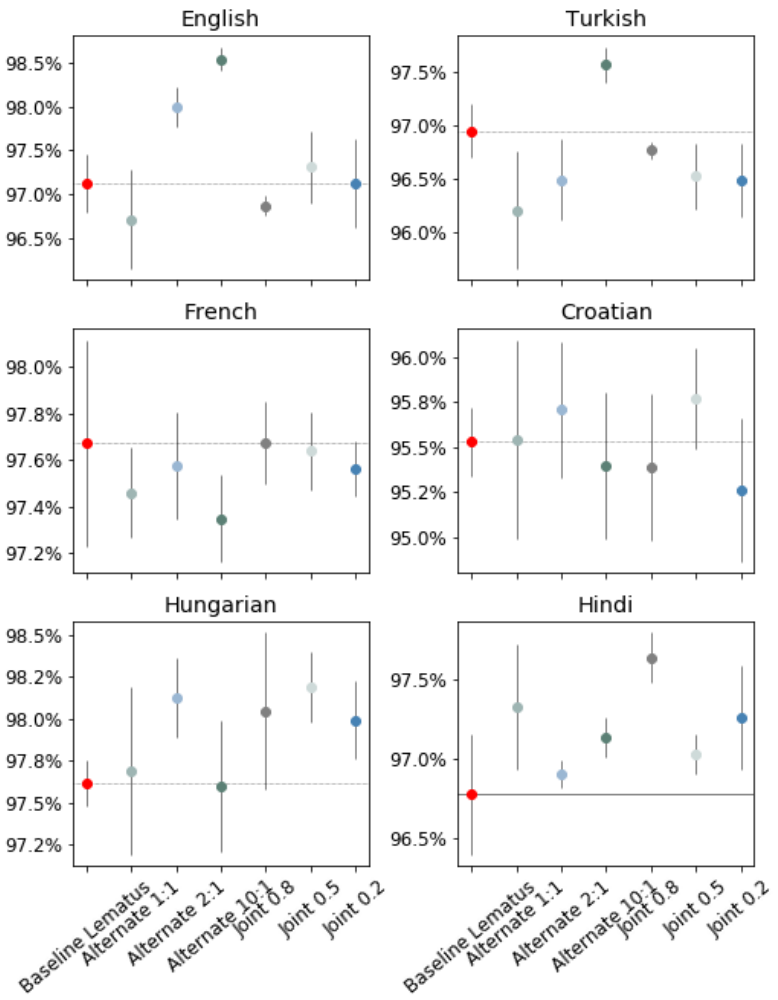


Figure B.1: Seen Validation accuracies for 10,000 training examples with the auxiliary task of auto-encoding, obtained by averaging three training runs. Plot details discussed Figure 4.1

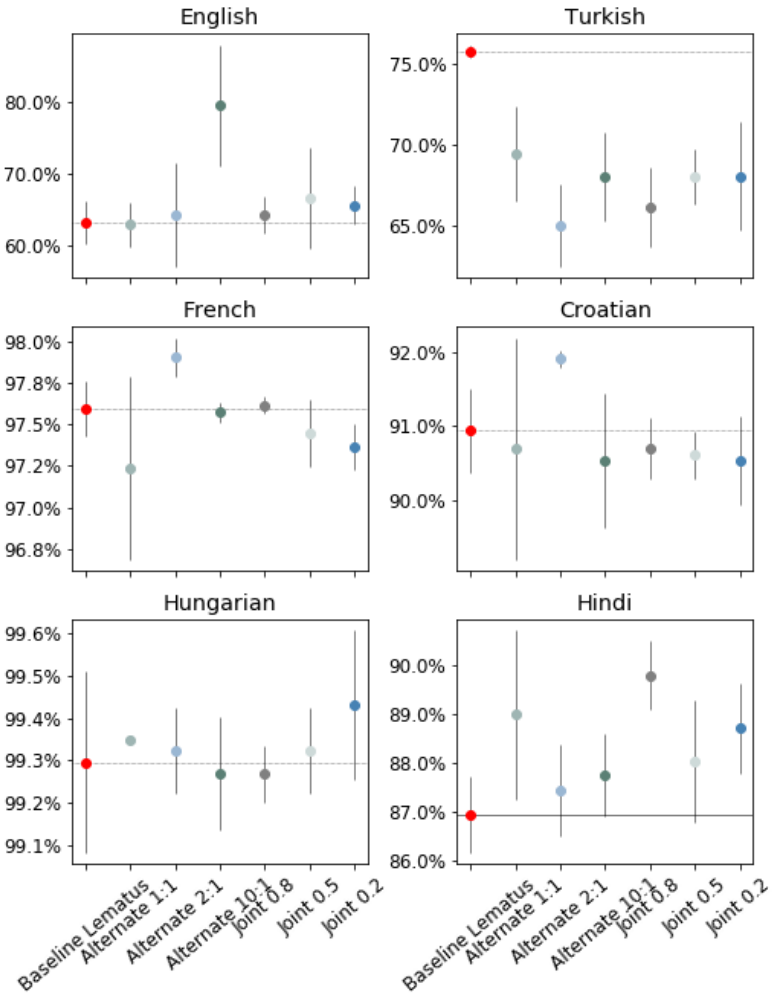


Figure B.2: Ambiguous Validation accuracies for 10,000 training examples with the auxiliary task of auto-encoding, obtained by averaging three training runs. Plot details discussed Figure 4.1

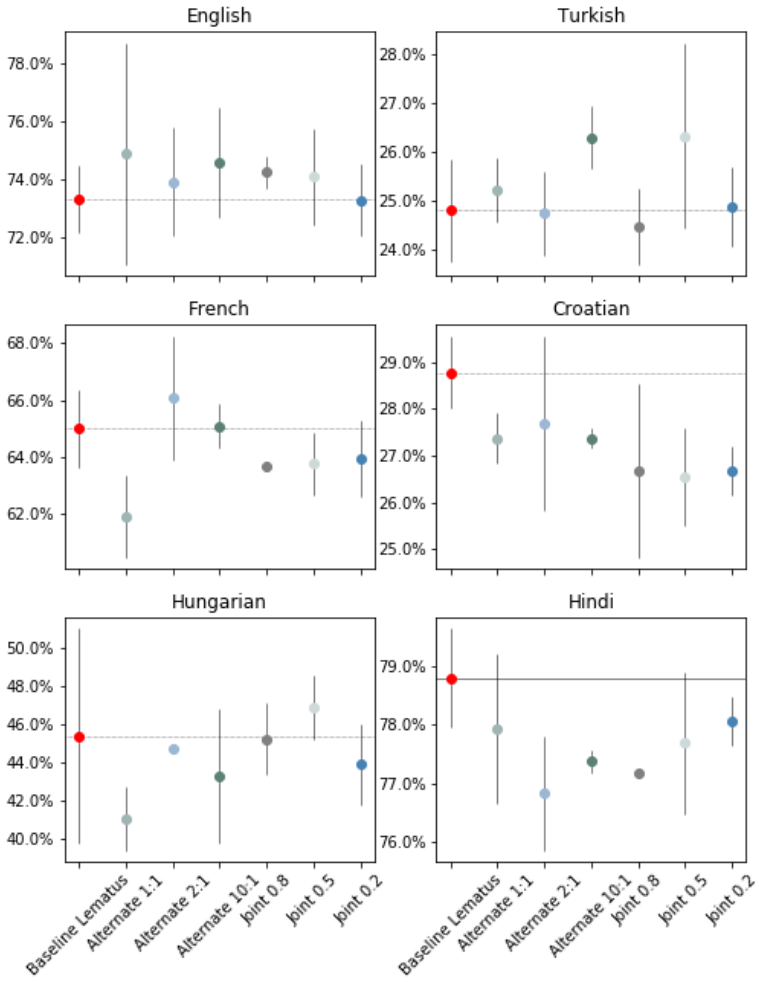


Figure B.3: Unseen copying behaviour for 10,000 training examples with the auxiliary task of auto-encoding, obtained by averaging three training runs. Plot details discussed Figure 4.1

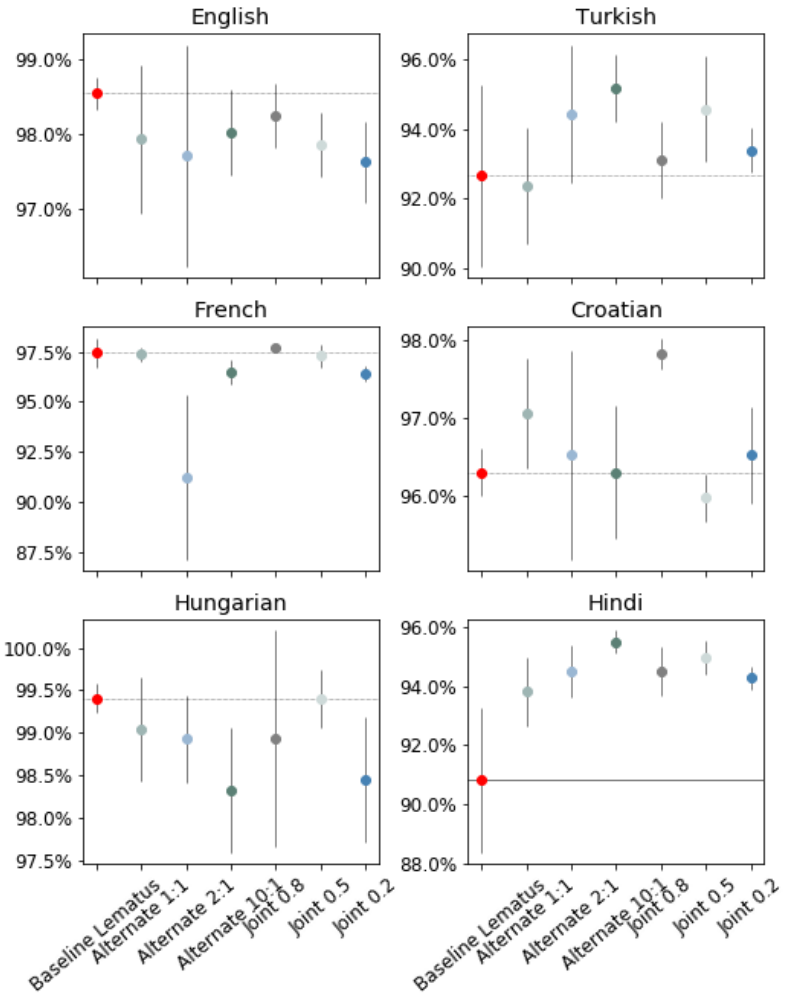


Figure B.4: Seen Validation accuracies for 1000 training examples with the auxiliary task of auto-encoding, obtained by averaging three training runs. Plot details discussed Figure 4.1

Appendix C

Example Ensembling Results: Auto-Encoding

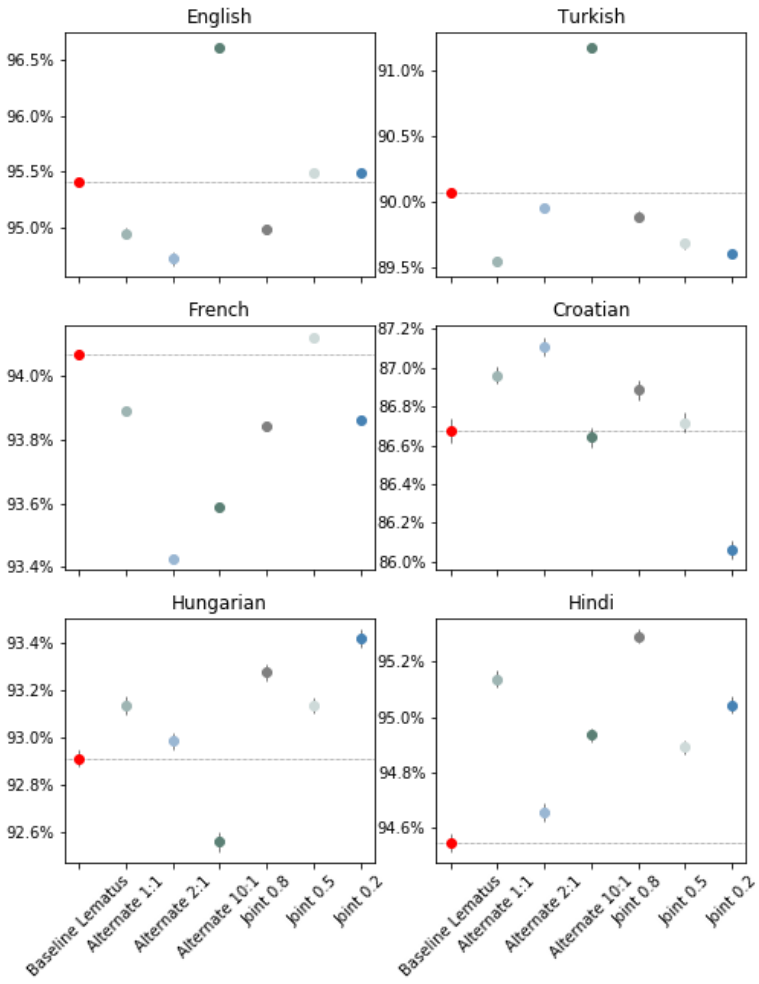


Figure C.1: Overall Validation accuracies for 10,000 training examples with the auxiliary task of auto-encoding, obtained by ensembling three training runs. Plot details discussed Figure 4.1

This appendix contains ensembling results, the plots for all types of accuracy were not included as they offered little extra insight. The conclusions are very similar to those of average accuracies, although as is often expected with ensembling, the accuracies are often slightly higher when ensembled in comparison to when they are averaged. It was interesting to observe how much disagreement there was between training runs for each model (e.g. the number of outputs for the ensemble for each setup which were randomly chosen from three different outputs). However, ultimately there was no clear trend in MTL either increasing or decreasing this level of disagreement.

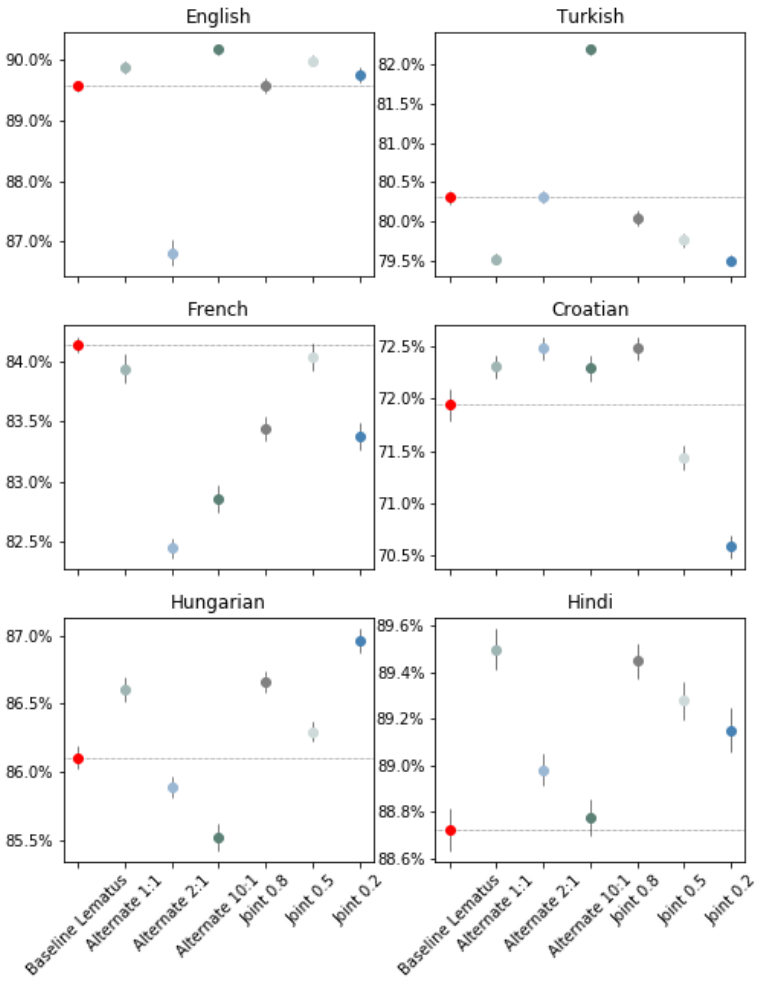


Figure C.2: Unseen Validation accuracies for 10,000 training examples with the auxiliary task of auto-encoding, obtained by ensembling three training runs. Plot details discussed Figure 4.1

Low Resource

It was also interesting to note what happened to those models which had high standard deviations for average accuracies. Often, for example in the case of the performance of the French Alternate 2:1 setup in the low resource setting, the ensemble result was not noticeably lower for these models than for those with less deviation in performance.

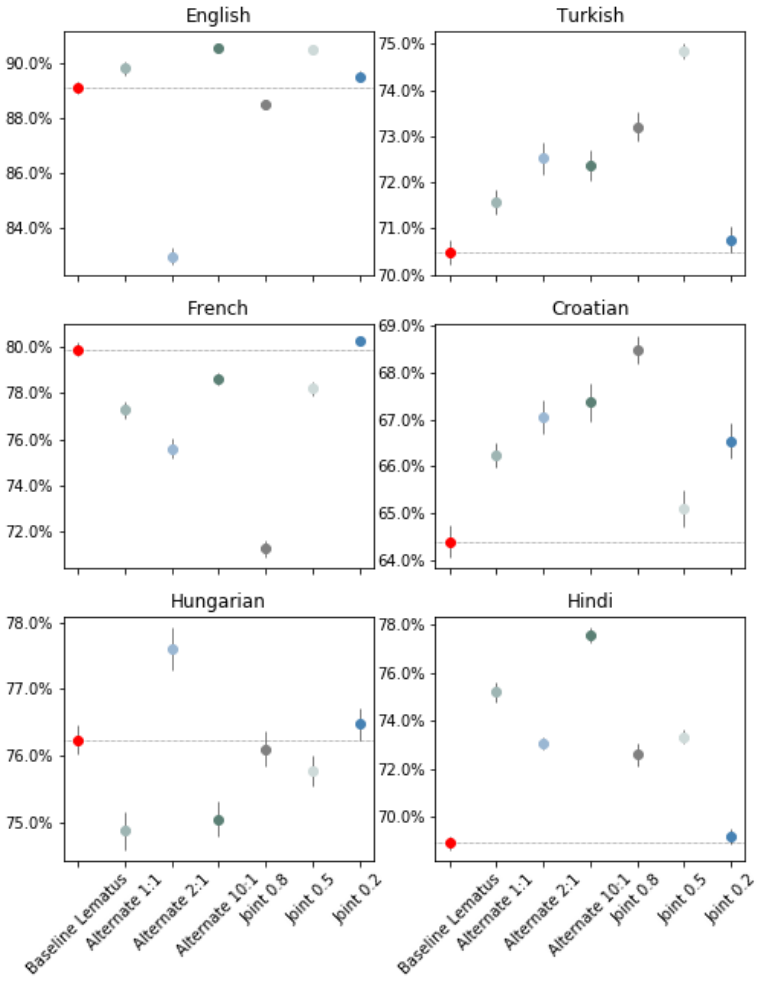


Figure C.3: Overall Validation accuracies for 1000 training examples with the auxiliary task of auto-encoding, obtained by ensembling three training runs. Plot details discussed Figure 4.1

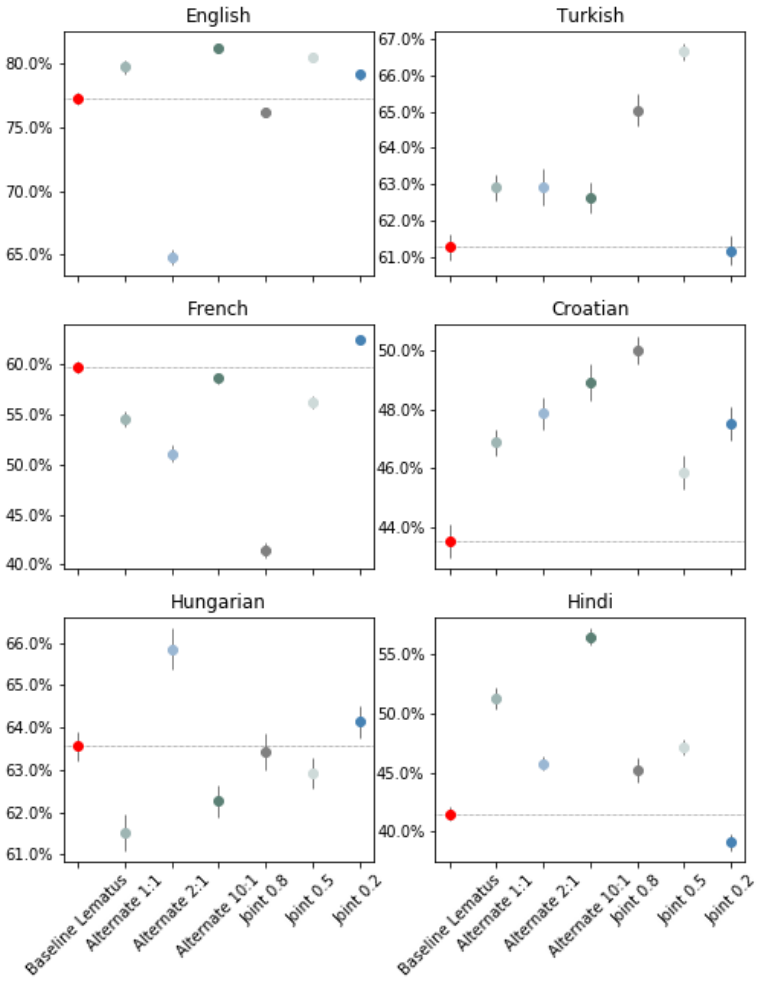
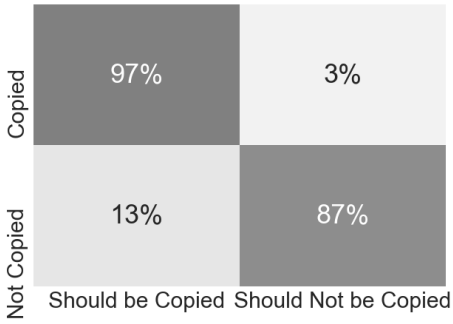


Figure C.4: Unseen Validation accuracies for 1000 training examples with the auxiliary task of auto-encoding, obtained by ensembling three training runs. Plot details discussed Figure 4.1

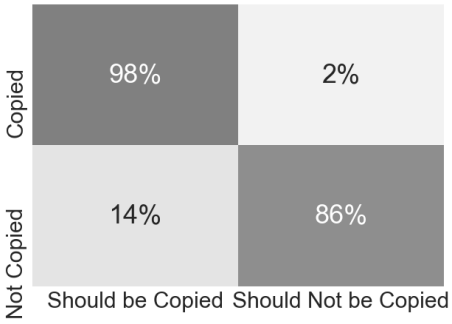
Appendix D

Further Analysis of Auto-Encoding Results

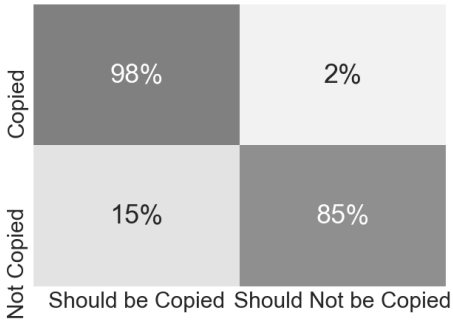
The results outlined in Chapter 4 lead to a very broad question. If the improvement observed for some languages when the auxiliary task of auto-encoding is added to the model is not caused by increasing the level of copying, particular in the medium resource setting, then what is it caused by? An obvious possible explanation is that although copying has not increased, perhaps the confusion matrices shown in Figures 3.13 and 3.15 have been improved. However, this is clearly not the case as demonstrated in Figures D.1 and D.2 where each model has either the same or a higher level of false negatives with respect to copying.



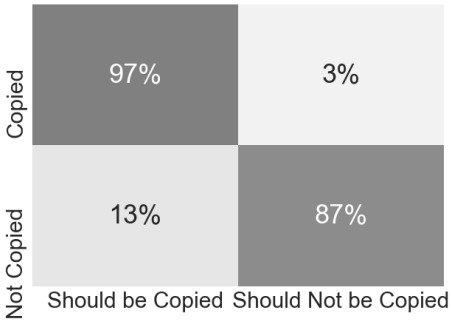
(a) Hindi Lematus Baseline



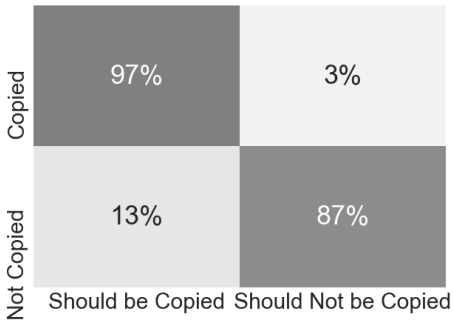
(b) Hindi Alternating 10:1



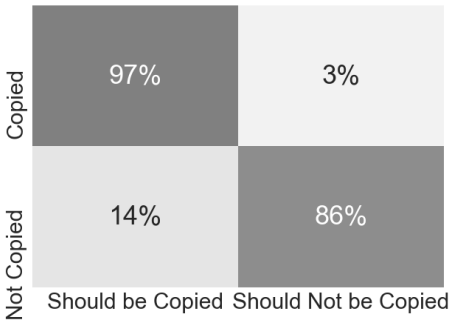
(c) Hindi Alternating 2:1



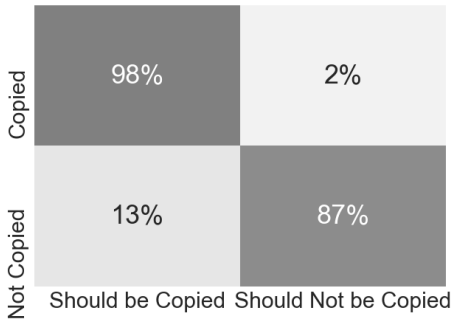
(d) Hindi Alternating 1:1



(e) Hindi Joint $\alpha = 0.2$

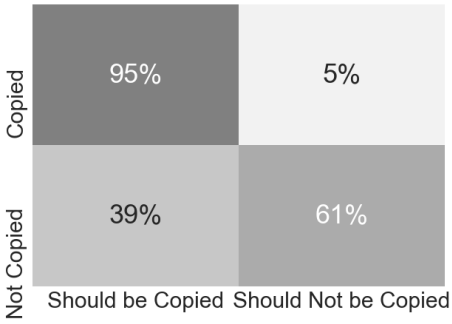


(f) Hindi Joint $\alpha = 0.5$

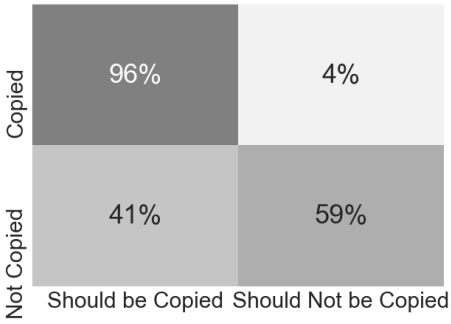


(g) Hindi Joint $\alpha = 0.8$

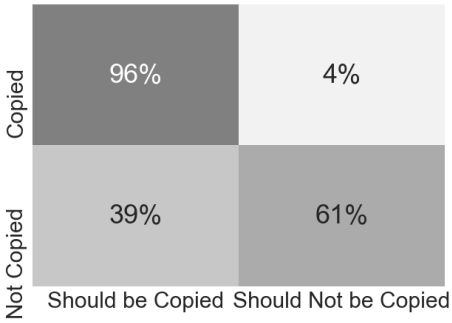
Figure D.1: Confusion Matrices for Copying Behavior in Lematus in the Medium Resource Setting



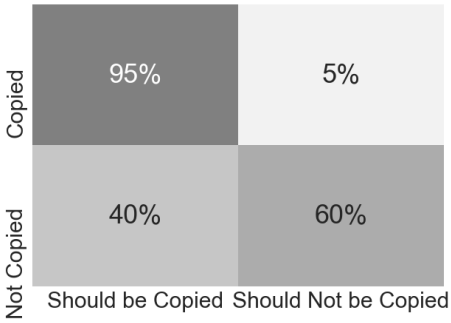
(a) Hindi Lematus Baseline



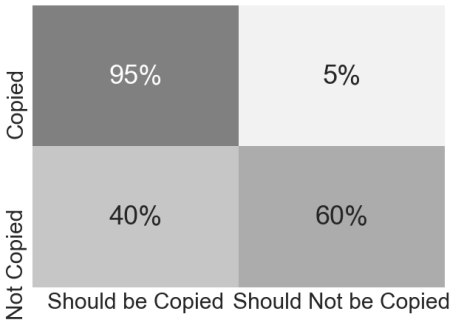
(b) Hindi Alternating 10:1



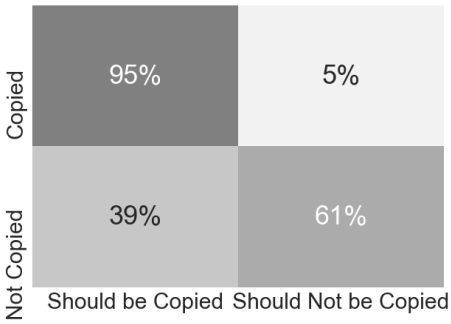
(c) Hindi Alternating 2:1



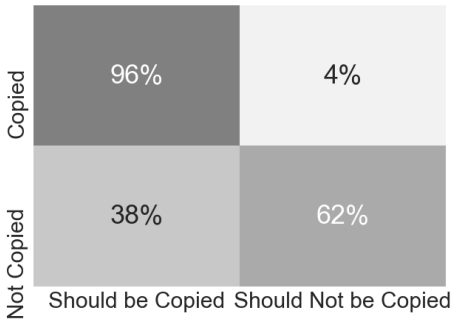
(d) Hindi Alternating 1:1



(e) Hindi Joint $\alpha = 0.2$



(f) Hindi Joint $\alpha = 0.5$



(g) Hindi Joint $\alpha = 0.8$

Figure D.2: Confusion Matrices for Unseen Copying Behavior in Lematus in the Medium Resource Setting

Appendix E

Example Ensembling Results: Part of Speech Tagging

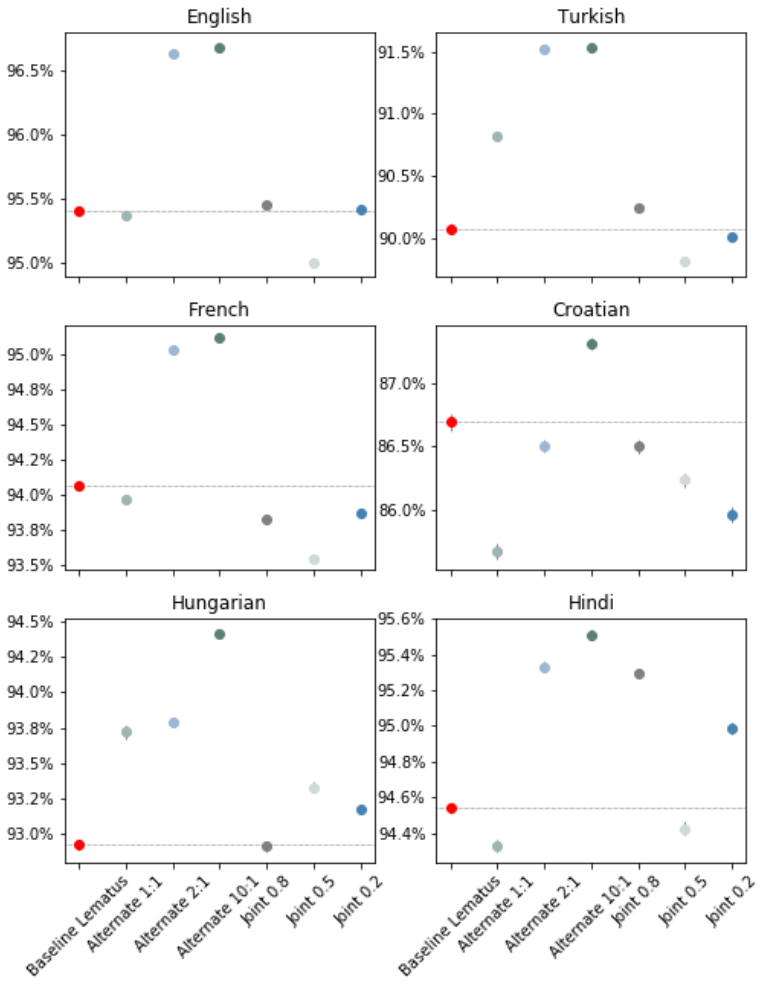


Figure E.1: Unseen Validation accuracies for 10,000 training examples with the auxiliary task of POS tagging, obtained by ensembling three training runs. Plot details discussed Figure 4.1

Again, some example results obtained by ensembling instead of averaging accuracies are included here, although they offered little addition insight.

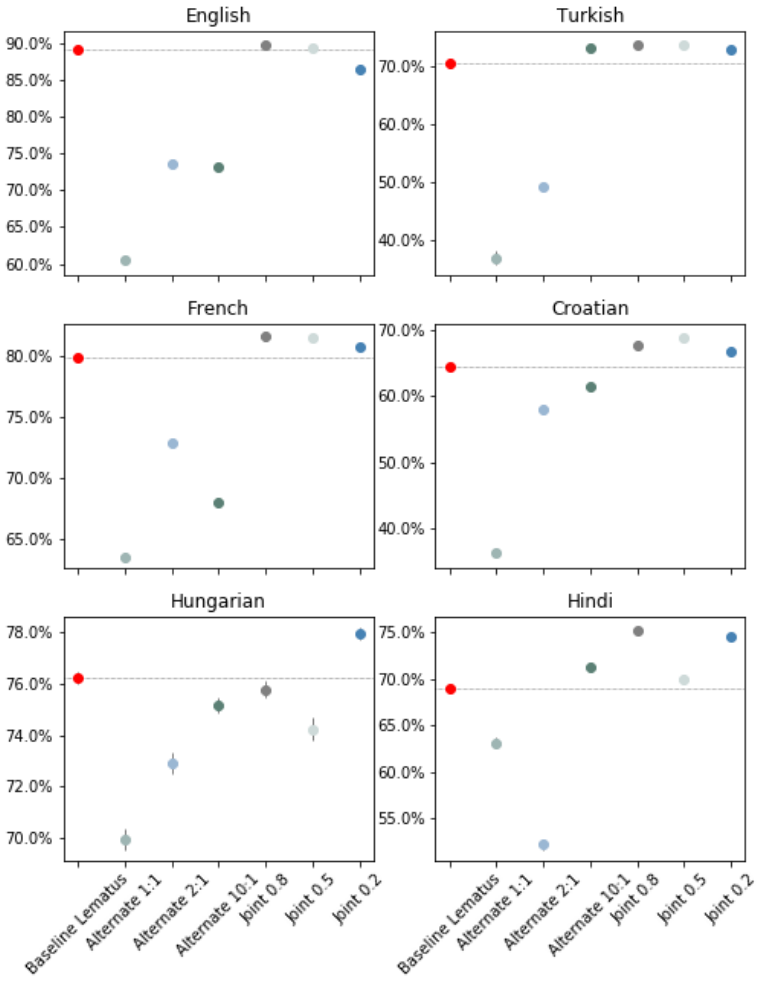


Figure E.2: Unseen Validation accuracies for 1000 training examples with the auxiliary task of POS tagging, obtained by ensembling three training runs. Plot details discussed Figure 4.1

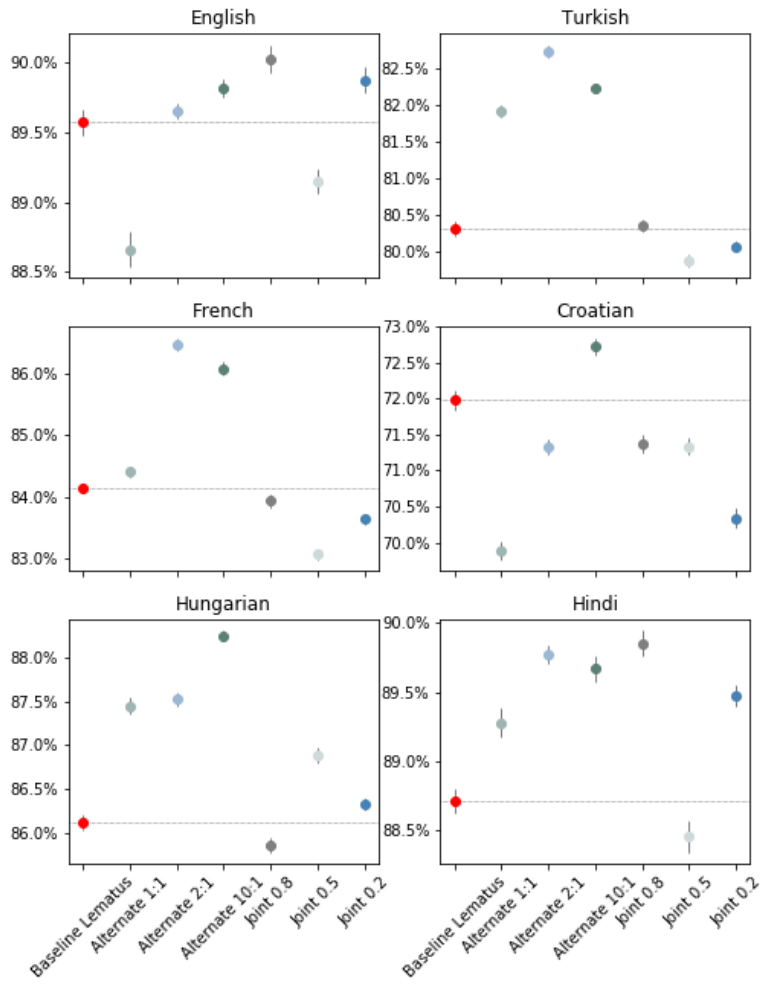


Figure E.3: Unseen Validation accuracies for 10,000 training examples with the auxiliary task of POS tagging, obtained by ensembling three training runs. Plot details discussed Figure 4.1

Appendix F

Test Set Results for Both Auxiliary Tasks

Medium Resource

The same patterns were shown in the test set results for multi-task learning using both tasks as were present for the validation set results. Both English and Hungarian slightly improved over the best test set results for an individual auxiliary task, while Hindi was slightly decreased.

Model	English	Hindi	Hungarian
Lematus Baseline	94.30%	93.95%	88.91%
Best MTL Model	96.46%	94.67%	91.77%
Improvement	+2.16%	+0.72%	+2.85%

Table F.1: (Auxiliary task: Auto-Encoding + POS Tagging) Average Overall Test Accuracy figures

Model	English	Hindi	Hungarian
Lematus Baseline	96.70%	96.92%	97.51%
Best MTL Model	99.04%	97.61%	98.91%
Improvement	+2.34%	+0.70%	+1.39%

Table F.2: (Auxiliary task: Auto-Encoding + POS Tagging) Average Seen Test Accuracy figures

Model	English	Hindi	Hungarian
Lematus Baseline	87.94%	86.77%	78.90%
Best MTL Model	89.63%	87.78%	83.46%
Improvement	+1.69%	+1.01%	+4.55%

Table F.3: (Auxiliary task: Auto-Encoding + POS Tagging) Average Unseen Test Accuracy figures

Model	English	Hindi	Hungarian
Lematus Baseline	72.22%	86.66%	97.95%
Best MTL Model	88.52%	90.55%	98.98%
Improvement	+16.30%	+3.89%	+1.02%

Table F.4: (Auxiliary task: Auto-Encoding + POS Tagging) Average Ambiguous Test Accuracy figures

Low Resource

Again, Hindi was slightly improved while Turkish was noticeably disimproved in comparison to test set results for a single auxiliary task. Croatian slightly decreases from 4.46% improvement for auto-encoding to 4.08% here.

Model	Hindi	Turkish	Croatian
Lematus Baseline	69.29%	66.54%	61.67%
Best MTL Model	79.25%	70.17%	65.75%
Improvement	+9.96%	+3.62%	+4.08%

Table F.5: (Auxiliary task: Auto-Encoding + POS Tagging) Average Overall Test Accuracy figures

Model	Hindi	Turkish	Croatian
Lematus Baseline	88.65%	91.67%	94.83%
Best MTL Model	92.28%	93.82%	94.37%
Improvement	+3.62%	+2.16%	+-0.46%

Table F.6: (Auxiliary task: Auto-Encoding + POS Tagging) Average Seen Test Accuracy figures

Model	Hindi	Turkish	Croatian
Lematus Baseline	47.57%	56.28%	42.81%
Best MTL Model	66.49%	61.21%	49.61%
Improvement	+18.92%	+4.93%	+6.80%

Table F.7: (Auxiliary task: Auto-Encoding + POS Tagging) Average Unseen Test Accuracy figures

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association. ISBN 978-1-931971-33-1. URL http://dl.acm.org/citation.cfm?id=3026877.3026899.

Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint*, 2016.

Héctor Martínez Alonso and Barbara Plank. When is multitask learning effective? semantic sequence prediction under varying data conditions. *arXiv preprint arXiv:1612.02251*, 2016.

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

Toms Bergmanis and Sharon Goldwater. Context sensitive neural lemmatization with lematus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1391–1400, 2018.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. Training data augmentation for low-resource morphological inflection. *Proceedings of the*

CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection, pages 31–39, 2017.

Jean Berko. The child's learning of english morphology. *Word*, 14(2-3):150–177, 1958.

Joachim Bingel and Anders Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303*, 2017.

R Caruna. Multitask learning: A knowledge-based source of inductive bias. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 41–48, 1993.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014a.

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

Grzegorz Chrupała. Simple data-driven context-sensitive lemmatization. *Procesamiento del lenguaje natural, n^o 37 (sept. 2006), pp. 121-127*, 2006.

Grzegorz Chrupała, Georgiana Dinu, and Josef Van Genabith. Learning morphology with morfette. 2008. URL `http://www.aclweb.org/anthology/L08-1176`.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1723–1732, 2015.

Wolfgang U Dressler and Ferenc Kiefer. Austro-hungarian morphopragmatics. *Contemporary morphology*, 49:69, 1990.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in

recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016.

Péter Halácsy and V Trón. Benefits of deep nlp-based lemmatization for information retrieval. In *CLEF (Working Notes)*, 2006.

Minwoo Jeong, Kristina Toutanova, Hisami Suzuki, and Chris Quirk. A discriminative lexicon model for complex morphology. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas (AMTA 2010)*, 2010.

Jakub Kanis and Lucie Skorkovská. Comparison of different lemmatization approaches through the means of information retrieval performance. In *International Conference on Text, Speech and Dialogue*, pages 93–100. Springer, 2010.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Zachary C Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*, 2018.

Qiuhua Liu, Xuejun Liao, and Lawrence Carin. Semi-supervised multitask learning. In *Advances in Neural Information Processing Systems*, pages 937–944, 2008.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, 2015.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çağrı Çöltekin, and Miriam Connor et al. Universal dependencies 2.1, 2017. URL `http://hdl.handle.net/11234/1-2515`. LINDAT/CLARIN

digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–909, 2016.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain, April 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/E17-3017`.

Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 231–235, 2016.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Zhizheng Wu, Cassia Valentini-Botinhao, Oliver Watts, and Simon King. Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4460–4464. IEEE, 2015.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark

detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014.