On Context Restriction for Low Resource Neural Morphological Analysis

Aibek Makazhanov

Master of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2022

Abstract

Contextual morphological analysis is the task of finding the most probable lemma and morpho-syntactic description (i.e. part of speech and grammatical markers, such as case, tense, etc.) for a given word in a given context. Historically, approaches to the task relied on using few words of local context due to both model design (e.g. HMM) and data sparseness concerns. With the advent of deep learning, resorting to local context ceased to be a necessity, and modern approaches exclusively use global (sentential) context. In this thesis we investigate whether context restriction can still be useful for neural morphological analysis.

For our first set of experiments we adapt a character-level encoder-decoder model that was previously used for related tasks of lemmatization and morphological generation. We start by showing that using just one word of surrounding context not only yields better results, but is also more efficient than using global context. Then, on a data set of more than hundred corpora, we show that relying on larger context windows is preferable only when training data is sufficiently large, while using a single word of context is better both in low resource scenarios and on average. We also discuss competitive performance of our model at SIGMORPHON-2019 shared task on contextual morphological analysis, where it was bested only by the systems that used pre-trained contextualized and/or regular word embeddings. Finally, we show that augmenting our model with contextualized word embeddings does not increase its performance.

Inspired by the success of our character-level model, in our second set of experiments we try context restriction with a popular off-the-shelf word-level neural morphological analyzer. Here too, we show, that when training data is scarce, limiting context to a few words does improve performance, especially for agglutinative and fusional languages. However, with enough data using global context is still better. To investigate what restricted models miss from global context, in a follow-up experiment we show, that context restriction hinders the ability of the model to correctly analyze words, whose dependency heads are left beyond the context window. Finally, we find, that to improve performance on small data sets, one does not even have to train in a context-restricted manner. It is enough to limit context at inference time to achieve comparable performance.

Acknowledgements

I am sincerely grateful to my supervisor, Sharon Goldwater, for her patience, support, guidance, and advice. This could be much worse, if not for her.

I am thankful to my co-supervisor, Adam Lopez, for helpful advice, encouragement, and old paper recommendations.

Thank you to my first and second year review committee, Alex Lascarides and Frank Keller for useful feedback and understanding.

Thank you to Bonnie Webber and Mark Steedman for their hospitality.

Thank you to fellow students: Sameer, Clara, Yevgen, Arthur, Ramon, Kate, Toms, Elizabeth, Lucia, Nick, Lena, and others.

Thank you, Bibs, Popsie, mom, and dad.

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.

(*Aibek Makazhanov*)

Table of Contents

Chapter 1

Introduction

Natural language is ambiguous. This is the first rule that NLP students are encouraged to live with. Luckily, the students soon learn that words keep company, which they can be known by. Thus, the importance of local context is hinted at early on. In practice, it is manifested by the necessity of limiting context, be it choosing the order of an N-gram language model or a word2vec window size. In statistical era, this strategy was motivated by both, design of certain models (e.g. HMM) and the fact that linguistic data are sparse; that is, using larger context windows causes a model to end up with more parameters or higher feature variance.

With the rise of neural NLP, especially ubiquitous usage of various types of recurrent neural networks, RNN (Elman, 1990), the practice of context restriction has fallen from grace. The reason being, of course, that it defeats the whole purpose of RNNs, which were designed to handle long sequences. Yet, recent findings suggest that language models based on long short-term memory networks, LSTM (Hochreiter and Schmidhuber, 1997), when limited to 9-13 previous tokens, achieve about the same perplexity as in the unrestricted mode (Chelba et al., 2017) and sharply distinguish between nearest 50 tokens and more distant history (Khandelwal et al., 2018). Thus, for LSTMs the utility of context might ultimately be limited to a certain number of nearest tokens. If so, could that be true of other NLP tasks, not just language modeling? Also, the smallest corpus used in both of the aforementioned studies contains about million tokens of running English text (Mikolov et al., 2010). Both works report consistent findings on larger corpora, but do not experiment with smaller data. Could it be that, due to data sparseness, the amount of "useful" local context is even smaller for smaller corpora? This is important, because linguistically annotated data is expensive to obtain and for the vast majority of languages having million tokens of even POS-tagged data

is unheard of. To answer the questions raised above, in this thesis we experiment with context restriction for neural morphological analysis.

Why set morphological processing in context, if morphology itself is concerned with the structure of individual words and not phrases or sentences? Indeed, if shared tasks reflect a trend in community effort, judging by the recent competitions, one must conclude that computational morphology is geared towards addressing context-insensitive tasks, such as paradigm[1] segmentation (Wiemerslage et al., 2021), paradigm completion (Kann et al., 2020), and morphological inflection (Cotterell et al., 2018b, 2017). Historically too, some of the earliest works on morphological processing focused on developing finite state transducers (Koskenniemi, 1983; Oflazer, 1994) to morphologically analyze individual words. Spell-checking became one of the first practical applications of such transducers, especially for languages with extensive morphological systems, such as Basque (Aduriz et al., 1997) and Turkish (Oflazer and Guzey, 1994), among others. Further research into processing such morphologically complex languages has shown (Hajic and Hladka, 1998; Hakkani-Tür et al., 2002) that approaching even very basic tasks, e.g. POS-tagging, made little sense without extracting grammatical information (e.g. number, case, etc.) morphologically encoded in word forms. This meant developing context-sensitive morphological analyzers, since word forms may stem from different lemmata (dictionary form) and have different grammatical properties, depending on context. Moreover, there has been numerous evidence for the utility of the morphological information in tasks, such as parsing (Eryiğit and Oflazer, 2006; Tsarfaty et al., 2010), NER (Tür et al., 2003; Straková et al., 2014), and SMT (Goldwater and McClosky, 2005; Koehn and Hoang, 2007). More recent work on language modeling (Vania and Lopez, 2017) and dependency parsing (Vania et al., 2018) has also shown that *ground truth* morphological analyses are more effective as neural representations than various sub-word units. However, due to limitations of automatic analyzers, *predicted* analyses fail to deliver comparable gains and, in case of language modeling, are even inferior to char-level representations. Thus, there is an incentive in developing highly accurate contextual morphological analyzers.

Given a sentence, the task of contextual morphological analysis is to assign each word most probable analysis, which consists of a lemma, part-of-speech tag, and a set of grammemes (values of grammatical categories, such as accusative case, future tense, etc.). In terms of use of context, existing statistical approaches operate in a restricted fashion, relying on surface-level features and/or already predicted lemmata/tags. Com-

[1] A totality of inflected forms of a lexeme, e.g. {run, ran, running} is a paradigm of RUN.

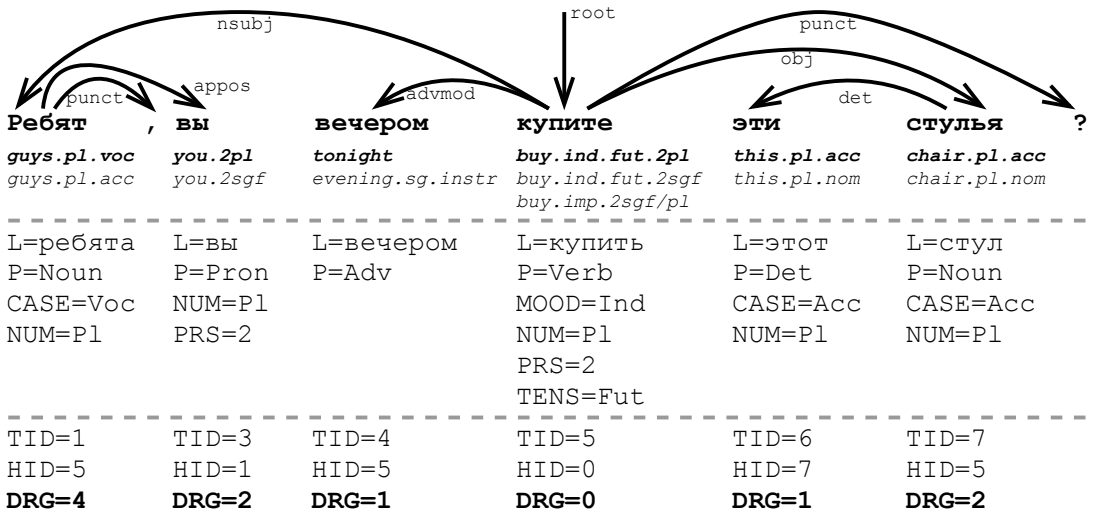


Figure 1.1: UD-style dependency tree for a Russian sentence. Each token (except punctuation) is annotated for the following: English gloss for the current (top) and alternative analyses; lemma (L=), POS tag (P=), present grammatical markers, token id (TID), head id (HID), and dependency range, i.e. $DRG = |TID - HID|$ (the root token always has zero range).

mon implementations include HMM (Hakkani-Tür et al., 2002; Makhambetov et al., 2015), log-linear (Chrupała et al., 2008; Straka et al., 2016), and CRF (Müller et al., 2015) models. Conversely, apart from an early work by Yildiz et al. (2016), who use a feed-forward network with a context window of two words, all of the existing neural approaches employ some variation of RNN and utilize global context.

It can be said that contextual morpho-analysis deals with two levels of structure: morphological (word level) and syntactic (sentence/context level). There is a somewhat odd relationship between the two. While context helps to resolve word level ambiguity, increase in the latter (i.e. richness of morphology) allows flexibility in word order, making context patterns sparser and leading to long range dependency issues. An example[2] from Russian (cf. Figure 1.1) helps to illustrate this point. Every word in the given sentence has at least one alternative analysis, showcasing various types of ambiguity. The determiner and the nouns are ambiguous due to case syncretism. The number and register (formal/informal) of the pronoun depends on the number of the noun it refers to,[3] i.e. anaphoric agreement. The adverb can also be a singular

[2]Although the example is constructed, most of the analyses occur at least once in one or more Russian UD treebanks. Also, in UD the first word is always lemmatized as a plural noun, such as *pliers*, *scissors*, etc.; hence glossed as *guys*.pl.voc/acc.

[3]For instance, “dear friend”, instead of “guys”, would have licensed a singular formal “you”.

noun in instrumental case; an example of lemma ambiguity. The verb is ambiguous for mood and person-number. The latter must agree with the subject, while the former can be resolved through sentence-final punctuation: 2nd person imperatives typically do not issue questions, though can appear in interrogative sentences.[4] To disambiguate the verb, we need the first and the last tokens, which in practice means using context window of at least four words. However, the verb can be easily placed at the end of the sentence, increasing dependency range for the subject to six words and urging us to consider global context for generality. On the other hand, from our example, we can see that most of the tokens have dependency range of less than three. In fact, about 83% of tokens across four Russian treebanks in our data set have dependency range of at most three.[5] Given that large context windows capture increasingly sparser patterns, especially when training data is small, and most of the words need just three words of context to disambiguate, the practice of context restriction, largely abandoned by neural approaches to morphological analysis, may be reconsidered.

In this thesis we experiment with context restriction for neural morphological analysis. We consider both character- and word-level context, using sliding windows that capture one to three words of surrounding context. In case of character-level context, we find that relying on local rather than global context is generally preferable. Even more so when training data is scarce, where we find that using the least amount of context yields best results. When working with word-level context, we find that context restriction leads to better performance only in low resource scenarios and only for morphologically complex languages.

1.1 Thesis Outline

Chapter 2 provides relevant background, starting with morphology and morphological typology. We continue by surveying approaches to relevant morphological processing tasks. We close the chapter with an overview of SIGMORPHON 2019 shared task on contextual morphological analysis (McCarthy et al., 2019), which had direct influence on our work.

In Chapter 3 we present an attention-based encoder-decoder that works directly with character-level context. We work with data, containing 107 Universal Depen-

[4] As in "*do* this, will you?". In Russian this type of mood ambiguity is resolved through stress, e.g. the verb from our example would be imperative rather than indicative had stress fallen on the second instead of the first syllable.

[5] For all 66 languages (107 treebanks) in our data set this ratio is 81.8%.

dency (Nivre et al., 2016) treebanks for 66 languages. We show that our model: (i) achieves both better performance and efficiency in context restricted mode; (ii) should be limited to as little context as possible in low resource settings; (iii) can compete with world-level context models that do not use pre-trained contextualized word embeddings; (iv) cannot be improved via incorporation of pre-trained contextualized word embeddings.

In Chapter 4 we experiment with word-level context restriction. To this end we train an off-the-shelf implementation of a stacked LSTM network. Here, we work with a small sub-set of the data we used in our experiments with character-level context. This allows us to better control for linguistic and domain variation, as we experiment with incremental increase in data size. We conclude that word-level context restriction only makes sense for agglutinative and fusional languages in low resource scenarios. We also investigate shortcomings of the restricted model and find that, when there is enough training data, it is less likely (compared to an unrestricted counterpart) to correctly analyze words whose dependency heads are left outside the context window. Finally, we find that performance gains from context restriction can be achieved at inference time, without having to train the model in the restricted mode.

Chapter 5 contains conclusions and discussion of future work.

Chapter 2

Background

In this chapter we summarize background relevant for the whole thesis. We start by laying out relevant key concepts of morphology as a field of linguistics and explain how languages relate to each other with respect to morphological properties. Following that we survey existing work on both context-insensitive and -sensitive morphological analysis and related tasks. The chapter closes with an overview of a shared task on contextual morphological analysis, which had direct influence on many experiments discussed in this thesis.

2.1 Morphology and Morphological Typology

Morphology is a subfield of linguistics that studies formation and structure of words. For our purposes words can be defined as meaningful units that are combined to produce phrases and sentences. Words themselves consist of morphemes, the smallest linguistic units that bear meaning. Consider words like +*read*, +*read*+*s*, +*read*+*ing*, +*read*+*er*, and +*re*+*read*, whose every morpheme is preceded by a plus sign. All of them have a common morpheme +*read*, that bears some core meaning. It is called a free morpheme, because it does not need any other morphemes to convey meaning. In this case, the free morpheme has *lexical* meaning and corresponds to the **lexeme** *to read*, which has its own dictionary entry and can be further changed as *reads*, and *reading*. Conversely, morphemes like +*s* and +*er* are called grammatical or bound morphemes, because their meaning is manifested only upon combination with free morphemes and has no use otherwise. Such attachable morphemes are also called **affixes** and the process of attachment is called **affixation**. Depending on the place of attachment, an affix is called: a **prefix** (frontal attachment, e.g. +*re*+*read*); a **suffix**

(trailing attachment, e.g. +*read*+*s*); an **infix** (insertion, e.g. +*ma*+*wa*+*ni*, Lakhota for "he walks" (Albright, 2000)); a **circumfix** (simultaneous frontal and trailing attachment, e.g. +ge+geb+en, German for "given" (Haspelmath, 2002)).

Bound morphemes either add to or change the core meaning of free morphemes and are classified as **inflectional** or **derivational** accordingly. The morphemes +*s* and +*ing* in +*read*+*s* and +*read*+*ing* are inflectional[1] as they do not change the core meaning of +*read*, but rather mark person and tense to clarify who and when is doing the act of reading. On the other hand, the morphemes +*er* and +*re* in +*read*+*er* and +*re*+*read* are derivational, as they change the core meaning of +*read*. Derivation also happens via compounding (e.g. +*goose*+*berry*) or blending (e.g. [+*smoke*] + [+*fog*] = +*smog*) of free morphemes, sometimes with necessary addition of bound morphemes, e.g. +*best*+*sell*+*er*. Regardless of how a derived form came to be, it constitutes a new lexeme[2] that may be further inflected just like a lexeme or lexemes it was derived from. Indeed, +read+er has a plural form (+*read*+*er*+*s*) and +*re*+*read* must agree with a third person singular subject in present indefinite tense (+*re*+*read*+*s*). Thus, a single lexeme may have one or more **word forms** or **surface forms**.

A totality of surface forms of a given lexeme is called a **paradigm**. In our example, *read*, *reads*, and *reading* exhaust all possible surface forms of the verb *to read* and constitute its paradigm.[3] The canonical form of a paradigm is called a **lemma**; in this case it is *read*. Each surface form corresponds to one or more **lexical forms** or **morphological analyses**, which consist of a lemma, a part of speech, and a set of **grammemes**, i.e. values of grammatical categories, such as singular or plural grammatical number. For example, depending on context *read* can agree with a first/second person singular/plural subject in present indefinite and all person/number subjects in past indefinite tense, i.e. be analyzed as *read*.v.1sg.pres, *read*.v.1sg.past, etc. Moreover, *read* can be a member of non-verbal paradigms, i.e. be a noun (as in "a long *read*.n.sg") or an adjective (as in "she is well *read*.adj"). The task of contextual morphological analysis, therefore, is to find the most probable sequence of lexical forms given a sequence

[1] Here +*ing* is also a derrivational morpheme that produces a noun (not merely a gerund), e.g. "The two short *readings* are for tomorrow".

[2] Not in all languages e.g. in Kazakh a new form may be derived from an already inflected one: *taularymdaghylargha*, here +*tau* (mountain.sg.nom) is first inflected for number and possession (+*tau*+*lar*+*ym* - my mountains), then relativized (+*tau*+*lar*+*ym*+*daghy* - that, who is in my mountains), and then further inflected (+*tau*+*lar*+*ym*+*daghy*+*lar*+*gha* - to those, who are in my mountains). Such patterns of derivation after inflection make it difficult to associate new lexemes with relativization and some other derivational constructs. This happens in other Turkic languages too (Tyers et al., 2017).

[3] Paradigms may include not only inflected, but also suppletive or irregular forms, e.g. the paradigm of *to be* is completely irregular: *am, are, be, been, is, was, were*.

(typically, a sentence) of surface forms. Naturally, certain languages pose more of a challenge just by virtue of having larger paradigms. For instance, while most English verbs have four or five surface forms in a paradigm, Turkish verbs have paradigms of 120 forms on average (Cotterell et al., 2018a). On the other hand, in Turkish a single morpheme is typically mapped to a single grammeme (*ev-ler-in* - house-pl-gen), while in Slovak a morpheme can correspond to multiple grammemes (*dom-ov* - house-masc.pl.gen). Thus, it is useful to understand how languages relate to and differ from each other in terms of intricacies of their morphology.

The sub-field of linguistic typology is concerned with studying the range of variation across languages (Bender, 2013). **Morphological typology**, in particular, studies variation in morphological traits. Languages can be classified by the average number of morphemes per word, by the strictness of morpheme boundaries, and by the manner of combination of morphemes (Fromkin et al., 2017; Bender, 2013).

Analytic and synthetic languages

On the spectrum from "few" to "many" morphemes per word, languages are classified as analytic or synthetic respectively. On the extreme ends of the spectrum are isolating and polysynthetic languages. Isolating languages, such as Vietnamese and Yoruba, have no inflectional morphology and consequently have monomorphemic words. Conversely, polysynthetic languages, such as Cherokee and Menominee, are often described as having sentence-words, e.g. the Menominee word *paehtawaewesew* means "*He is heard by higher powers*" (Fromkin et al., 2017). For other languages, however, there are no clear quantitative criteria for classification. It is generally understood that analytic languages have more rigid word order and rely more on functional and auxiliary words, while synthetic languages employ developed declension, conjugation, and agreement mechanisms instead. Synthetic languages are further classified with respect to the ease of morpheme boundary detection.

Agglutinative and fusional languages

In agglutinative languages, inflection happens via attachment of morphemes (from either or both left and right) to the lemma, usually in strict relative order and with easily recognizable boundaries, e.g. case marker always follows plurality marker and there is no merger between them. In fusional languages, morphemes are often merged or fused together to the point that markers of grammatical categories cannot be singled out. Table 2.1 illustrates the distinction on the example of Kazakh (agglutinative) and Russian (fusional) noun declension.[5] As it can be seen, declension of singular forms

[5]Both languages are written in Cyrillic script, which, unfortunately, we could not use due to technical

	Kazakh (agglutinative)		**Russian (fusional)**	
Case	**Singular**	**Plural**	**Singular**	**Plural**
Nominal[4]	tau	tau+lar	gor+a	gor+y
Genitive	tau+dyng	tau+lar+dyng	gor+y	gor
Accusative	tau+dy	tau+lar+dy	gor+u	gor+y
Dative	tau+gha	tau+lar+gha	gor+e	gor+am
Locative	tau+da	tau+lar+da	-	-
Ablative	tau+dan	tau+lar+dan	-	-
Prepositional	-	-	gor+e	gor+ah
Instrumental	tau+men	tau+lar+men	gor+oi	gor+ami

Table 2.1: Declension of the noun "mountain" in Kazakh vs Russian.

is straightforward in both languages: one morpheme - one case marker. The difference becomes evident in the declension of plural forms. In Kazakh, the plural form morpheme *+lar*[6] attaches first followed by case endings with clearly distinguishable morpheme boundaries. In Russian, however, number and case endings blend together into a single morpheme.

Nonconcatenative morphology

In some languages words undergo morphological change without affixation, involving processes collectively referred to as non-concatenative morphology (Haspelmath, 2002). Ablaut and umlaut are prominent examples of non-concatenation that involve sound alternations (Fromkin et al., 2017), e.g. *see - saw*, *mouse - mice*, etc. Another type of non-concatenation is reduplication, where words are completely or partially repeated, e.g. Kazakh *kyzyl* (red) becomes *kyp-kyzyl* (flaming red). Finally, **templatic languages** undergo transfixation, where root consonants fill slots in various templates to produce word forms (cf. Table 2.2).

difficulties and had to resort to English transliteration.

[6]Just like certain cases in English, e.g. *fox+es* not *fox+s*, Kazakh (and many other Turkic languages) has its own morphophonological rules, mostly concerning vowel harmony and consonant agreement. Thus, *+lar* actually has five more variations or **allomorphs**: *+ler*, *+dar*, *+der*, *+tar*, *+ter*. A particular allomorph is used depending on the vowel (back or front) and the final consonant (voiced, voiceless, or nasal) of the preceding syllable, e.g. a plural for "*bucket*" is *shelek+ter* (front vowel, voiceless final consonant). This is true for the majority of the inflectional morphemes.

[6]In both languages lemmata correspond to the singular nominative form of "*mountain*", that is, *tau* in Kazakh and *gora* in Russian. However, in Kazakh nominative case has a form of a **null morpheme** with no surface realization, while in Russian there are forms of nominative case endings that vary according to grammatical gender. Also, note that in Kazakh the lemma remains intact in all inflected forms, while in Russian it blends with inflectional morphemes and in many forms cannot be singled out by mere segmentation of a word form.

Template	POS	Word form	Gloss
CaCaC	verb	katav	wrote
hiCCiC	verb	hixtiv	dictated
miCCaC	noun	mixtav	a letter
CCaC	noun	ktav	writing, alphabet

Table 2.2: Transfixation of the Hebrew root *ktb* (Arad, 2005; Bender, 2013). Here, C denotes root consonant.

As it can be seen various languages employ elaborate morphological systems making morphological analysis a challenging task. Before we survey language-specific and -agnostic approaches to some of the morphological processing tasks, let us make an important note on derivational morphology. As we have mentioned before, in some languages, a derived form is usually considered to be a new lexeme with its own lemma and part of speech. In this case, a morphological analyser is expected to work accordingly by producing the lemma and any grammemes present in the surface form of the new, derived lexeme. To our knowledge, most of the trainable systems do (certainly, the ones we adapt/use), as long as training data is annotated accordingly. However, depending on annotation conventions, which are, in turn, influenced by the tradition of morphological processing, in some languages productive derivational morphemes maybe analyzed as "detached" from the rest of the surface form and assigned its own lemma and inflections. A known case in our data set is Turkish, where some derivational morphemes are treated as separate tokens. For example, *masadakilerin* (table.sg.loc.rel.pl.gen - *of those at the table*) is analyzed as two tokens: *masada* (table.sg.loc) and *kilerin* (_.rel.pl.gen), with the latter having a lemma of *ki* (Sulubacak et al., 2016). The models we experiment with in the following chapters (as well as the baselines) are able to work with these cases, provided that the input is tokenized accordingly; which it is.

2.2 Morphological Processing

2.2.1 Finite State Morphology

Early work on morphological analysis and inflection (an opposite operation to analysis) centered around design and usage of finite state automata and transducers (FST).

Formally, FST is a six-tuple $(\Sigma_1, \Sigma_2, Q, i, F, E)$, where: Σ_1, Σ_2 are finite input and output alphabets, respectively; Q is a finite set of states; $i \in Q$ is the initial state; $F \subset Q$ is the set of final states; E is the set of state transitions, where each transition is a quadruple $(q \in Q, \alpha \in \Sigma_1, \beta \in \Sigma_2, p \in Q)$ (Roche and Schabes, 1997). An FST accepts an input sequence, if all the elements of the sequence are in the input alphabet and yield a valid path through state transitions. Since each valid transition between symbols of the input alphabet, corresponds to a transition between symbols of the output alphabet, an accepted input sequence corresponds to a unique[7] output sequence.

Application of FSTs to morphology began when it was shown that phonological rewrite rules applied in succession could be compiled into an FST (Johnson, 1972; Kaplan and Kay, 1981). The problem was that those rules mapped phonological representations into surface forms with a deterministic outcome, but not vice versa. Applied to morphology that meant ability to unambiguously generate words from lexical forms, but not the other way around (Karttunen and Beesley, 2001). The solution was found in two-level morphology (Koskenniemi, 1983), where rewrite rules were formalized as constraints and could be applied in parallel, not in succession. This allowed to unambiguously produce all possible lexical forms for a given surface form, by compiling FSTs from language-specific lexicons, and phonological and morpho-syntactic rules. Soon after original work on Finnish, two-level morphology descriptions started appearing for other languages, among others, for Arabic (Beesley, 1991), Basque (Agirre et al., 1992), Turkish (Oflazer, 1994). This, in turn, lead to development of language-specific (Sproat, 1991; Oflazer and Guzey, 1994) FSTs with hard-coded rules and, later, generic libraries (Beesley and Karttunen, 2003; Linden et al., 2010) capable of compiling FSTs for any language, given a lexicon and a two-level description.

Earliest practical applications of morphological FSTs were spell-checkers, where words were considered misspelled if they could not be analyzed by a transducer (Oflazer and Guzey, 1994; Aduriz et al., 1997). More recent applications include rule-based machine translation systems (Forcada et al., 2011) and contextual morphological analyzers (Hakkani-Tür et al., 2002; Makhambetov et al., 2015).

2.2.2 Morphological Disambiguation

Morphological disambiguation is the task of choosing most probable morphological analysis or its part (usually morphological tag) for each word in a sentence given

[7] Non-determenistic transducers may produce several outputs (Roche and Schabes, 1997).

supplied candidate set. Candidate analyses are usually obtained from morphological FSTs or look up tables. The benefit of tackling the task is that it solves both lemmatization and morphological tagging problems (provided that disambiguated candidates constitute complete analyses). Common approaches to the task include modeling either joint or conditional distribution over morphological tags and surface forms, using HMM (Hakkani-Tür et al., 2002; Makhambetov et al., 2015; Assylbekov et al., 2016) or log-linear models (Hajic, 2000; Sak et al., 2007; Habash and Rambow, 2005) respectively. The main challenge is data sparseness caused by large tag sets involved, as the task itself commonly arises for morphologically complex languages.

Hakkani-Tür et al. (2002) use 2$^{\text{nd}}$ order HMM to model a joint distribution of surface and lexical forms. They use Viterbi algorithm to decode the lattice of candidate analyses, by maximizing probability $P(L|S) = P(S|L)P(L)$,[8] where S and L denote surface and corresponding lexical forms in a given sentence. The authors make two key assumptions: (i) a lexical form "generates" exactly one surface form, thus term $P(S|L)$ can be dropped; (ii) the last inflectional group of the current tag depends on some inflectional group of preceding tags. Here, an inflectional group is a part of a tag separated by a derivational boundary. Thus, effectively, the authors maximize a marginal distribution of morpho-tag trigrams.

Hajic (2000) perform morphologcial tagging of five European languages using look-up tables to generate candidates tags. They define a positional tag set, where each grammatical category has a fixed position, e.g. case has position 3, and for each position a separate maximum entropy classifier is trained, that uses various features from surrounding context of up-to four words.

There also were neural approaches to the task, which disambiguated FST analyses by embedding both lemmata and tags into fixed sized representation and using feed forward network (Yildiz et al., 2016) or LSTMs Toleu et al. (2017) to predict the most probable candidates. Excluding Toleu et al. (2017), all of the aforementioned approaches operate within context-restricted paradigm.

2.2.3 Lemmatization

Early work on token-level or contextual lemmatization was based either on morphological disambiguation (Oflazer and Kuruoz, 1994; Ezeiza et al., 1998), or rule-based approaches (Erjavec and Dzeroski, 2004). One of the first works that framed contextual

[8]The term $P(S)$ is dropped from the denominator, since there is always only one surface form for a given set of candidate lexical forms.

lemmatization as a classification task was done by Chrupała (2006). The author automatically labeled lemmata with edit scripts that encoded edit operations (delete, insert, replace) between a surface form and a lemma. They then train a log-linear model with a simple set of surface-level features, including trailing characters of a preceding word. This method of using edit scripts, had since been used in many approaches to lemmatization and analysis, including modern neural approaches (Chakrabarty et al., 2017; Straka et al., 2019; Kondratyuk, 2019). Chakrabarty et al. (2017) and Straka et al. (2019) both use bidirectional RNNs to first encode each word as a sequence of characters and then pass the output to word-level contextual LSTM that performs predictions. Kondratyuk (2019) also uses character-level embeddings, but they use a transformer network to generate predictions.

All of the aforementioned neural approaches utilize global word-level context. Lematus (Bergmanis and Goldwater, 2018) is the only neural approach to lemmatization that does not use global context. They develop an attention based encoder-decoder that uses fixed amount of local character-level context to directly generate lemmata character-by-character. Varying the amount of context the authors show that using no context at all hurts performance, especially on ambiguous words. Malaviya et al. (2019); Kanerva et al. (2021) also do not use surface-level context with a similar sequence-to-sequence model. Instead a sentence is initially tagged by a morphological tagger, and then encoder-decoder model is trained to produce lemmata. Not surprisingly, being implemented as LSTM networks, these models utilize context, they just do not use exclusively local context as Lematus does.

2.2.4 Tagging

Just as with lemmatization, most of the early work on morphological tagging was in the framework of morphological disambiguation. One notable exception is the work by Mueller et al. (2013), where the authors develop a conditional random field-based approach, using context-level features extracted within two words of the current word. The tag set is decomposed into part of speech and grammeme tags.

2.2.5 Analysis

Most of the statistical approaches to morphological analysis employ morphological disambiguation. Notable exceptions include (Chrupała et al., 2008; Straka et al., 2016; Müller et al., 2015). Chrupała et al. (2008) extends their lemmatization model to jointly

predict tags in a log-linear fashion. Straka et al. (2016); Müller et al. (2015) both use edit scripts to classify lemmata and use log-linear and CRF models to predict tags respectively.

Neural approaches to contextual morphological analysis are reviewed in the next section.

2.3 Overview of the SIGMORPHON-2019 Shared Task

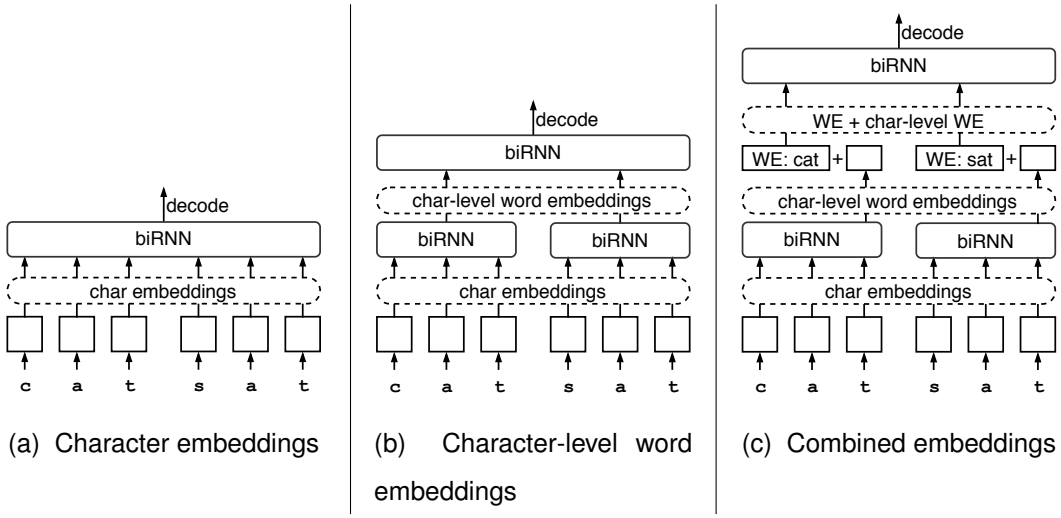
In 2019, ACL special interest group on morphology, phonetics, and phonology held a shared task on contextual morphological analysis (McCarthy et al., 2019). The data set consisted of 107 Universal Dependencies treebanks (Nivre et al., 2016) spanning 66 languages. As much of the work reported in the next chapter was influenced by our participation in this shared task, we close the background section with its overview.

Eleven teams (including ours) have provided 15 submissions to the shared task, all of which were based on neural approaches.[9] There was considerable variation in certain design decisions, e.g. using various combinations of end-to-end and pre-trained (both regular and contextualized, e.g. BERT (Devlin et al., 2018) and ELMO (Peters et al., 2018)) word embeddings; predicting tags and lemmata as whole or compositional units; using pipeline or multi-task learning approaches. Thus, at the most coarse level all of the works could be divided into three groups with respect to input/output encoding/predicting approaches and learning strategies used.

Figure 2.1 shows approaches to input encoding, on the example of the phrase "cat sat". Here, the universal strategy (used by everyone except us) was to use character-level word embeddings in the manner of the word-level composition method proposed by (Ling et al., 2015), where every word is represented by a sequence of its characters, which is then encoded using a bidirectional RNN (LSTM or GRU). The final states of the left-to-right and the opposite passes of such per-word networks are concatenated and fed to another biRNN (cf. Figure 2.1b). Some of the works have used combined embeddings (cf. Figure 2.1c), augmenting character-level word embeddings with end-to-end and/or pre-trained regular or contextualized word embeddings. Our strategy (cf. Figure 2.1a) was to use a simple character-level encoder.

When it comes to generating output, the systems either used edit scripts (Chrupała

[9]We do not review approaches by four teams, three of which (Aiken et al., 2019; Cardenas et al., 2019; Chaudhary et al., 2019) have focused either on tagging or lemmatization (not both), while another (by NLPCUBE), although mentioned in the results table, is not described in the any of the SIGMORPHON-2019 workshop papers.



(a) Character embeddings (b) Character-level word embeddings (c) Combined embeddings

Figure 2.1: Input encoding strategies, employed by the shared task participants.

et al., 2008) to generate lemmata or decoders that output lemmata character by character. Similarly tags were either predicted as whole units or generated by sequence decoders one grammeme at a time. Our model was unique in this respect, as it generated both lemmata and tags via a single decoder one character/grammeme at a time. As for learning methods, most of the systems employed either joint or pipeline-based learning strategies. The former involved having separate decoders for lemmata and tags, while the latter - conditioning either a lemmatizer or a tagger on the output of the other. Again, our model was unique in this respect, as both lemmatization and tagging were performed as one task.

Table 2.3 summarizes the key implementation decisions employed by the shared task systems, listing the latter by decreasing performance order. Our model is listed at the bottom just to illustrate contrast with existing approaches, its performance in the shared task will be discussed in the next chapter. It must be noted once again that *all* of the systems utilize global surface-level context. Also, the top three systems use the same representation and learning strategy, with the only difference being that Üstün et al. (2019) do not use ELMO (Peters et al., 2018) rather than BERT (Devlin et al., 2018) embeddings and generate output one character/grammeme at a time (wc2c), as opposed to generating an edit script for the lemma and predicting the tag as a whole. Interestingly, another system (Shadikhodjaev and Lee, 2019) that employs the same representation as the top two, but uses pipeline approach and no contextual embeddings performs rather poorly. Thus, best results are achieved by the systems that augment contextualized embeddings with char-level WE (cf. Figure 2.1c), train

System	Input enc.	Out Lem.	Out Tag	Learning
Straka et al. (2019)*	CWE+	Script	Tag	Joint
Kondratyuk (2019)*	CWE+	Script	Tag	Joint
Üstün et al. (2019)*	CWE+	Char	Gram.	Joint
Shadikhodjaev and Lee (2019)	CWE+	Char	Tag	Pipe
Oh et al. (2019)	CWE	Char	Gram.	Pipe
Yildiz and Tantuğ (2019)	CWE	Char	Gram.	Joint
Baseline (Malaviya et al., 2019)	CWE	Char	Tag	Pipe
Ours	CE	Char	Gram.	Single

Table 2.3: Input encoding, output granularity, and learning strategies employed by shared task systems. Here, CWE - character-level word embeddings (cf. Figure 3.4a); CWE+ - character-level word embeddings augmented with contextualized embeddings (cf. Figure 2.1b); CWE+ - character-level embeddings (cf. Figure 2.1a); * denotes usage of contextualized embeddings embeddings. For multiple submissions from a single team, the properties and results of the best performing system are reported.

separate decoders, and predict lemmata and tags on word level.

Chapter 3

Character-level Context Restriction

As was shown in Chapter 2, all of the systems submitted to the largest to date shared task on contextual morphological analysis utilize word-level character embeddings, i.e. encode words as character sequences. The incentive is to learn common patterns in word structure. However, when it comes to context, characters are deemed of no use, as words (regardless of representation) are taken as basic units of context. In this chapter we evaluate alternative strategy and present a neural morphological analyzer that works directly with character-level context. We show that our model: (i) achieves both better performance and efficiency in context restricted mode; (ii) should be limited to as little context as possible in low resource settings; (iii) can compete with world-level context models that do not use pre-trained contextualized word embeddings; (iv) cannot be improved via incorporation of pre-trained contextualized word embeddings.

3.1 Model

Our model is inspired by Lematus (Bergmanis and Goldwater, 2018), character-level attention-based sequence-to-sequence contextual lemmatizer. At the same time, our model extends morphological encoder-decoder (MED) developed by Kann and Schütze (2016) for context-insensitive re-inflection. In turn, both of these works modify the encoder-decoder with attention proposed by Bahdanau et al. (2015) for neural machine translation. Here, given an input sequence of vectors $\mathbf{x} = \{x_1, ..., x_N\}$, an RNN encoder computes hidden states $\{h_1, ..., h_N\}$, using a non-linear function f, as:

$$h_t = f(x_t, h_{t-1}) \tag{3.1}$$

Following that, conditional probability over output sequence $\mathbf{y} = \{y_1, ..., y_M\}$ is estimated as:

$$y_t = p(y_t|y_1, ..., y_{t-1}, \mathbf{x}) = g(y_{t-1}, s_t, c_t) \tag{3.2}$$

where, g is a non-linear function, $s_t = f(y_{t-1}, s_{t-1}, c_t)$ is a hidden state at time t of the decoder RNN, and c_t is a context vector computed for each step, as a weighted sum of encoder hidden states:

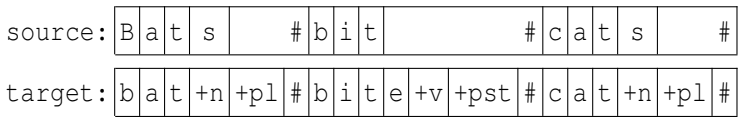
$$c_t = \sum_{i=1}^{N} \frac{e^{q_{ti}}}{\sum_{j=1}^{N} e^{q_{tj}}} \tag{3.3}$$

where, $q_{ti} = a(s_{t-1}, h_i)$ is the attention model, implemented as a feed forward network and trained jointly with both the encoder and decoder.

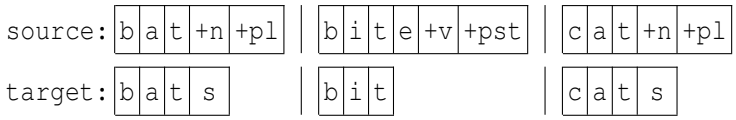
Our model is also a character-level encoder-decoder with attention. It is implemented in OpenNMT-py (Klein et al., 2017), a Python-based open source toolkit designed for working with sequence-to-sequence models. OpenNMT-py allows to assemble models from various pre-implemented components and to fine tune a rich set of parameters. In our model we use a stacked bidirectional LSTM for the encoder, an LSTM (Schuster and Paliwal, 1997) for the decoder, and global attention (Luong et al., 2015). Despite the difference in the choice of encoder/decoder architectures, conceptually it is the same basic attention model, as the one used in Lematus and MED. The only thing required to extend those approaches to the task at hand is to change what goes into source and target sequences that the models operate with. Thus, our is named after both, Lematus and MED, and hereinafter is referred to as LemMED.

LemMED is trained in a machine translation fashion, but instead of an aligned pair of sentences in two different languages, a training example contains the same sentence at the surface (words) and lexical (morpho-analyses) level (cf. Figure 3.1a). If we now consider each pair of surface and lexical forms in isolation and flip the source and target sides, as shown on Figure 3.1b, we get a representation used by MED for the inflection sub-task of the SIGMORPHON-2016 shared task (Cotterell et al., 2016). Thus, in the full sequence mode, LemMED extends MED simply by accounting for context and changing the direction of processing from inflection to analysis.

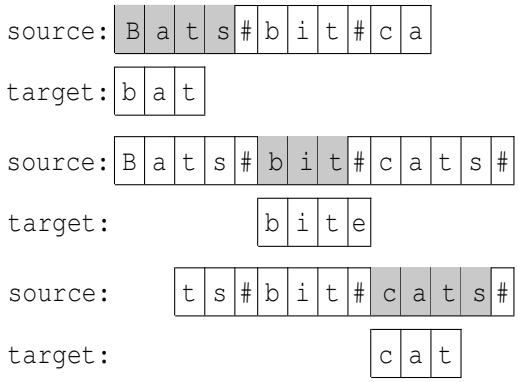
Similarly to MED, Lematus treats each token-lemma pair separately, but not in isolation from the rest of the sentence. Local context is retained on the source side in a form of surrounding characters captured by a fixed-sized context window (cf. Figure 3.1c). The context window is slid over a given sequence of surface forms and corresponding lemmata one form-lemma pair at a time. This results in a representation that has a pair of short source and target sequences per token, rather than one pair of



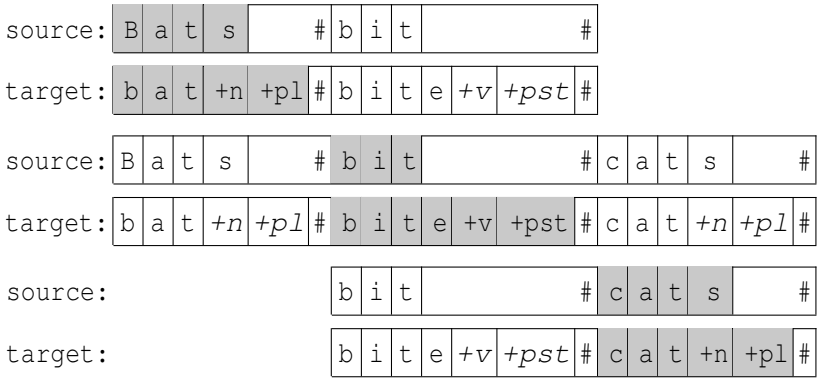
(a) Contextual analysis by LemMED in the full sequence mode.



(b) Non-contextual inflection by MED. Vertical lines signify isolated processing of each source-target pair.



(c) Contextual lemmatization by Lematus with a seven-character context window. The focal tokens are highlighted.



(d) Contextual analysis by LemMED with a single-word context window. For each source-target pair the highlighted parts denote the focal token and the corresponding lexical form.

Figure 3.1: An example sentence and the corresponding sequence of analyses/lemmata as represented by LemMED, MED, and Lematus for their respective tasks. The "source" and "target" labels convey the meaning generally attached to these terms in the sequence-to-sequence modeling framework and denote corresponding sequences. In every such sequence each individual cell represents a single processing unit: either a character or a grammeme. Hash sign (#) denotes a word boundary

sequences per whole sentence. We refer to this method of slicing the input into short snippets as "the context window mode". As can be seen from Figure 3.1d, in this mode our model extends Lematus by introducing three modifications. First, for LemMED the size of the context window is measured in words, not in characters like it is done for Lematus. Second, as opposed to Lematus, whose output (target sequence) corresponds to lemmata, in LemMED outputs complete lexical forms. Third, unlike Lematus that has one target lemma per snippet, our model has as many target-side forms as there are corresponding source-side forms. Let us elaborate on the latter distinction.

By default, in the context window mode there is an equal number of surface and lexical forms on the source and target sides respectively. This means that the model learns to predict target-side context (cf. Figure 3.1d, unhighlighted) along with the focal lexical form (highlighted). Yet, at test time only the latter is considered for the final prediction and evaluation, while context is ignored. Thus, including target-side context seems useless, as most of what is predicted gets thrown away, suggesting that the model optimizes for potentially redundant data. On the other hand, this way we are able to implicitly incorporate lexical context, without making any changes to the architecture of the model. Moreover, precisely because it is not evaluated, we can use virtually anything for target-side context, e.g. only lemmata or only tags or neither. At any rate, particular usage of target-side context is more of a design choice, and various possibilities are compared in subsection 3.3.3. Here it must be noted that such flexibility is not available in the full sequence mode, as input tokens are considered all at once and lack their own target context.

Finally, let us point out another important aspect of modeling target-side context. Suppose that all three target sequences depicted on Figure 3.1d are the outputs generated by LemMED at test time. Notice that each input token is predicted across multiple snippets. Specifically, predictions for "*Bats*" and "*cats*" appear in the snippets #1, 2 and #2, 3 respectively, while predictions for "*bit*" appear in all of the snippets. In general, given a context window of size W, sentence-initial and sentence-final tokens will be predicted $2W$ times, while predictions for the rest of the tokens will be made $2W + 1$ times. This allows for a majority voting scheme to be implemented, where for each input token the most frequent output is considered for the final prediction and evaluation. Hence, if modeled, not only target side context may provide contextual clues during training, it can, after all, be used directly at test time.

Metric	Splits		
	train	dev	test
sent-s, thou.	6.1±9.2	0.8±1.2	0.8±1.2
tokens, thou.	108.1±165.8	13.5±20.6	13.5±20.5
OOV rate	–	0.17±0.11	0.17±0.11
gramm-form	2.64±0.91	2.65±0.91	2.64±0.90

Table 3.1: Data set statistics: values correspond to means and standard deviations of respective metrics computed across all of the treebanks; here "gramm-form" is an average number of grammemes per lexical form.

3.2 Experimental Setup

3.2.1 Data Set

We work with the SIGMORPHON-2019 shared task data (McCarthy et al., 2019), sub-task #2 (hereinafter Shared task). The original data set consists of 107 UD treebanks (Nivre et al., 2016) converted to the Universal Morphology scheme (Kirov et al., 2018). We had to exclude two of the treebanks (Estonian-EDT and Portuguese-GSD), as one of our baselines (Lemming) kept crashing when trained on them. Thus, our data set contains 105 treebanks that cover 65 different languages.

Table 3.1 summarizes basic stats of the data set averaged over all treebanks and broken down into splits. Notice how standard deviation values of sentence and token counts are more than 1.5 times larger than the respective means. This suggest great variation in treebank sizes and should be accounted for during evaluation. Another thing to consider is a relatively high OOV[1] rate: with 1/6 of test data unseen, overall performance strongly depends on success in handling OOV. For this reason, whenever possible, in addition to overall performance we also report OOV results. Lastly, as a rough morphological complexity estimate, we gather statistics on the average number of grammemes per lexical form. The metric is used as a more convenient substitute to the degree of synthesis, defined as an average morpheme per word ratio (Greenberg, 1960), which we have no reliable way of estimating, since the data is not annotated for morpheme boundaries.[2] For our data set the average grammeme-form ratio of 2.6 is

[1]We apply the concept of OOV not to words, but to lexical forms. Thus, if we have observed *cut* only as a verb during training and it appears as a noun at test time, it is counted as unseen, even though it is the same *cut* on the surface.

[2]For instance, *workers* consists of three morphemes (*work-er-s*), but is annotated as bearing two

Parameter	Value	Comment
encoder type	brnn	biLSTM
decoder type	rnn	LSTM
# layers	2	both enc./dec.
# hidden units	500	both enc./dec.
dropout probability	0.3	both enc./dec.
attention type	global	Luong et al. (2015)
embeddings size	700	both source/target
beam size, at inference	5	default value

Table 3.2: LemMED: values of the key parameters.

about the same across all of the splits and is relatively high, attesting to the difficulty of the task. For reference: the average grammeme-form ratio computed over five English treebanks is 1.9 ± 0.05.

3.2.2 Parameter Settings and Baselines

LemMED is implemented in OpenNMT-py (Klein et al., 2017), v.0.5.0.[3] Table 3.2 lists the the key parameter-value pairs, which we use in all of the experiments, unless stated otherwise. For optimization we use SGD with an initial learning rate of 1.0, which is halved every 10k training steps starting from the step #25k. OpenNMT-py v.0.5.0 does not implement a stopping criterion and training continues for a fixed number of steps, which we set to 50k. Every 1000 steps the current version of the model is saved. As a result we end up with 50 models, of which the best one is selected based on the dev set performance. This model is then used for evaluation.

We compare LemMED to three baselines: (i) SIGMORPHON (Malaviya et al., 2019), Shared task baseline, a pipeline of an LSTM tagger and seq2seq lemmatizer; (ii) UDPipe (Straka et al., 2016), a statistical disambiguation model; (iii) Lemming (Müller et al., 2015), a statistical joint learning model. For SIGMORPHON we rely on results reported in Shared task review paper (McCarthy et al., 2019). For the other two baselines we use off-the-shelf implementations: UDPipe v.1.2.0 and Lemmingatize[4], a Python wrapper over the latest version of Lemming. We use both systems with default

grammemes and no morpheme boundaries, i.e. [*worker*; n, pl].

[3] https://github.com/OpenNMT/OpenNMT-py/commit/b085d57

[4] https://tinyurl.com/y2l7e6at

settings, changing only those options that affect general task-related behaviour, e.g. instructing Lemming to use morpho-tags, which it ignores by default. Both systems are trained for 50 iterations. Similar to LemMED, UDPipe selects the best version of the model based on the dev set performance. To our knowledge, Lemming does not employ any model selection strategy and uses the parameters updated and learnt during the final iteration of training.

Lastly, we would like to comment on the issues with the data format. As already mentioned (cf. subsection 3.2.1), the data was obtained by converting UD treebanks to the UniMorph scheme. As a result, POS- and morpho-tags that were stored in separate CoNLL-U fields before conversion, end up being mixed in one field following no particular order. This may potentially cause trouble for LemMED, therefore, for our model we pre-process the data by ordering grammemes alphabetically. The baselines are also affected by formatting. While UDPipe requires POS-tags and grammemes to be clearly separated, Lemming benefits from such an arrangement, albeit does not obligate it. Nevertheless, for both baselines we convert the data back to the original format, using the UM-UD compatibility[5] mapping in reverse.

3.2.3 Evaluation Metrics

Arguably, the metric that matters the most for the task at hand is the analysis accuracy, i.e. the percentage of tokens for which both lemma and tag were correctly predicted. Nevertheless, this metric was not considered in the shared task and the script provided by the organizers evaluates lemmatization and tagging performance separately, though on two levels. On a coarse level simple accuracy is computed as percent correct. On a finer level lemmatization is assessed in terms of average lemmatization distance, i.e. Levenshtein distance between predicted and golden lemmata. For tagging a fine-grained metric is an average F1 score between predicted and golden tags.[6] We modify the script to include the calculation of analysis accuracy and consider five evaluation metrics altogether: (i) lemmatization accuracy; (ii) average lemmatization distance; (iii) tagging accuracy; (iv) average tagging F1 score; (v) analysis accuracy.

[5] `https://tinyurl.com/y4qft2cy`

[6] For example, if the predicted tag is $p = \{n\}$ and the golden tag is $g = \{n, pl\}$, then precision is $P = |p \cap g|/|p| = 1$, recall is $R = |p \cap g|/|g| = 0.5$, and F1 score is $F1 = 2PR/(P+R) = 0.67$.

Model	Hardware	Quantity
LemMED	GPU, GeForce GTX, 1080 Ti	1
UDPipe	CPU, AMD Operton 6348	1
Lemming	CPU, AMD Operton 6348	48

Table 3.3: Hardware used to train the models.

3.3 Results and Discussion

In this section we discuss the results of three experiments designed to analyze the role of context in sequence-to-sequence morphological analysis. First, in subsection 3.3.1 we try to answer whether it is better to analyze a sentence as one long sequence or break it down into smaller sub-sequences, and if the latter is preferable, what optimal length such sub-sequences should be. Then, in subsection 3.3.2 we investigate if size or morphological complexity of the data affect the choice of optimal context size. Having decided on the size of context, in subsection 3.3.3 we investigate what type of context should be modeled within a target sequence, e.g. lemmata, morpho-tags, or a combination of both. Finally, in subsection 3.3.4 we evaluate the best version of LemMED on the test set.

3.3.1 Full Sequence vs Context Window Mode

In this experiment we compare two modes of operation of our model: full sequence (cf. Figure 3.1a) and context window (cf. Figure 3.1d). Recall that using a model in either mode simply means training and evaluating the model on either complete sentences or overlapping snippets of a given fixed size. This means that we can use the baselines, namely Lemming and UDPipe,[7] in the context window mode as well. We do not expect these models to be affected by context restriction, as both of them utilize local features, which is, in effect, a context window approach implemented internally. For this experiment we chose 10 treebanks to produce a typologically-balanced sample of languages: three analytic (Afrikaans, Chinese, English); two nonconcatenative (Arabic, Hebrew); two fusional (Bulgarian, Russian); and three agglutinative (Hungarian, Finnish, Turkish). In addition to performance, we also measure efficiency in terms of wall time spent on training. Table 3.3 lists the hardware that we use to train the

[7]For SIGMORPHON baseline we report Shared task results, which are only available for the full sequence mode.

Method	LACC	LDIS	TACC	T-F1	ACC	TIME
Full sequence						
LemMED	93.1 (80.9)	0.25 (0.62)	89.3 (72.4)	93.0 (84.6)	87.9 (66.5)	1191.8±733.6
SIGMORPHON	96.7 (83.7)	0.08 (0.38)	74.3 (65.8)	89.6 (82.4)	73.0 (58.9)	-
UDPipe	93.1 (70.4)	0.14 (0.65)	90.6 (63.9)	94.6 (80.5)	87.4 (51.8)	37.1±18.4
Lemming	94.9 (79.9)	0.10 (0.43)	91.5 (70.8)	95.3 (85.5)	88.9 (61.9)	42.1±52.2
Context window						
LemMED-CW0	94.8 (84.0)	0.11 (0.35)	79.0 (62.6)	87.1 (79.5)	77.3 (58.3)	54.6±13.8
LemMED-CW1	**97.3** (**85.5**)	**0.06** (**0.31**)	92.1 (72.9)	95.3 (85.8)	91.2 (67.8)	122.9±27.4
LemMED-CW2	97.1 (85.2)	**0.06** (0.32)	**92.5** (**74.6**)	**95.5** (**86.4**)	**91.5** (**68.9**)	231.4±66.2
LemMED-CW3	96.7 (84.8)	0.08 (0.35)	92.0 (74.1)	95.2 (86.2)	91.0 (68.6)	281.1±101.6
UDPipe-CW0	89.0 (71.3)	0.26 (0.63)	70.3 (56.5)	81.5 (75.5)	67.5 (47.0)	**4.6±2.2**
UDPipe-CW1	93.2 (70.6)	0.13 (0.64)	90.2 (62.4)	94.5 (79.8)	86.9 (50.5)	52.9±27.0
UDPipe-CW2	93.2 (70.6)	0.13 (0.64)	90.5 (63.2)	94.6 (80.2)	87.3 (51.2)	102.6±54.4
UDPipe-CW3	93.2 (70.5)	0.13 (0.65)	90.6 (63.3)	94.6 (80.1)	87.4 (51.1)	143.1±74.0
Lemming-CW0	89.1 (77.2)	0.21 (0.49)	68.9 (58.4)	80.2 (77.6)	66.0 (51.0)	23.5±42.4
Lemming-CW1	94.8 (79.5)	0.10 (0.44)	91.0 (67.2)	95.2 (83.8)	88.4 (58.8)	54.1±55.1
Lemming-CW2	94.8 (79.1)	0.10 (0.45)	91.1 (66.8)	95.1 (83.6)	88.5 (58.4)	97.9±77.9
Lemming-CW3	94.8 (79.0)	0.10 (0.45)	91.0 (66.3)	95.1 (83.1)	88.4 (58.0)	109.5±98.6

Table 3.4: Performance on development sets of a sample of 10 treebanks in the full sequence and the context window modes. Here: CW*N* - context window of size *N*; LACC - lemmatization accuracy; LDIS - average lemmatization distance; TACC - tagging accuracy; T-F1 - average tagging F1 score; ACC - analysis accuracy; TIME - training time in minutes. Results for unseen data are given in parentheses. Best results for the full sequence mode are highlighted; best results overall are shown in bold.

models.

Table 3.4 contains two sets of results pertaining to the models' performance and efficiency in the two modes. As it can be seen, in the full sequence mode LemMED performs poorly achieving 93.1%, 89.3%, and 93.0% in lemmatization, and tagging accuracy and F1 score respectively. This corresponds to 3.6%, 2.2%, and 2.3% net decrease compared to the best performing baselines and ranks our model only third on the respective metrics. As for lemmatization distance, LemMED is ranked last, suggesting that, when it gets lemmata wrong, it is off by a greater number of edit operations, on average, than that of the baselines.

Despite poor performance on individual metrics, in terms of combined lemmatization and tagging (ACC), LemMED ranks second. Another strength of LemMED appears to be in handling unseen data, performance on which is given in parentheses for each metric. Here our model achieves the highest tagging and analysis accuracy of 72.4% and 66.5% with respective net improvement of 1.6% and 4.6% over the second

best model. On other metrics, with the exception of LDIS, LemMED is ranked second on unseen data. Nevertheless, these promising aspects of performance are tainted by terrible inefficiency of the full sequence mode, where, on average, our model takes almost 20 hours to train on a single treebank. As the time complexity of our model is polynomial with respect to the lengths of source and target sequences, the context window mode, where said sequences are much shorter, should speed things up.

For the context window mode we experiment with windows of size 0 (no context) to 3. The first thing to notice is the fact that using any amount of context is superior to using no context at all. This is observed across all models and metrics, except for LemMED, which does better in lemmatization accuracy and distance without context than with unlimited context (the full sequence mode). This is where the advantages end, as on the remaining metrics, LemMED-CW0 loses 6-10% to the full sequence version. Thus, we do not consider the no context scenario in the rest of the experiments. Also, as we expected, UDPipe and Lemming do not benefit from context restriction and perform on par or slightly worse than they do in the full sequence mode. Moreover, these models are more efficient in the full sequence mode; hence we use them as such in all the experiments that follow.

In contrast to the baselines, our model shows clear improvement across all of the metrics when restricted to context windows of size 1 to 3. The net increase in performance over the full sequence mode ranges from 2.2% in tagging F-score (for LemMED-CW3) to 0.19 points (76% relative) in lemmatization distance (for LemMED-CW1 and -CW2). Moreover, training time reduces as context gets shorter and for LemMED-CW1 reaches a reasonable average of 2 hours per treebank. Thus, limiting context to shorter sequences improves character-based sequence-to-sequence morphological analysis with respect to both performance and efficiency. In fact, LemMED-CW2 achieves the best overall results on most of the metrics, improving 2.6% in the analysis accuracy over the best performing baseline. On the other hand LemMED-CW1 is only slightly less accurate, while being twice as fast to train. Thus, before deciding on the optimal context size, we would like to compare all three models on a larger sample with more variation in size and linguistic properties of the data.

3.3.2 Size and Morphological Complexity of Data

In order to investigate the influence that size and morpho-complexity of the data have on performance, we evaluate our models and the baselines on the dev sets of all the

treebanks. Then, we take the training sets and build distributions over two statistics (averaged per treebank): number of tokens and number of grammemes per lexical form. The latter is used as a rough estimate of morphological complexity as explained in subsection 3.2.1. Lastly, we break each distribution into five bins and calculate average (per bin) performance for each model and baseline.

Figure 3.2 shows the resulting plots, where rows correspond to the data statistics and columns to the evaluation metrics. Unsurprisingly, increase in training set size leads to increase in performance across almost[8] all models and metrics. One would expect morpho-complexity to have the opposite effect, i.e. steady decline in performance with the growth of complexity. However, according to Figure 3.2 (middle row), performance of the models fluctuates on most of the metrics, typically rising on the first and the last intervals and falling in between. Apparently, if ranked by the training set size, 3 of the bottom 10 and 2 of the top 10 treebanks fall into the first and the last bins respectively, with the bulk of the rest falling into the middle bin. As we already know, all of the models perform at the opposite extremes on those two groups of treebanks. If, for the purpose of our analysis, we consider those treebanks to be outliers and exclude them, we get the expected steady decline in tagging and analysis accuracy for all of the models, except SIGMORPHON (cf. Figure 3.2, bottom row). As for lemmatization accuracy, with the outliers removed, the growth of morpho-complexity influences the statistical models to a greater degree and in a more expected fashion than it does the neural models. This may suggest greater reliance of the statistical models on the tagging component. LemMED, on the other hand, lacks a separate tagging component altogether, and, as we show in the next subsection, presence or absence of morpho-tags in the local context does not seem to influence lemmatization accuracy of our model.

When it comes to the baselines, none seem to cope with increasing morpho-complexity better than LemMED-CW1, performing worse than or on par with our model regardless of the evaluation metric. What matters, though, is the data size, as all of the baselines beat our model on the smallest treebanks in every evaluation metric (ignoring SIGMORPHON's poor tagging and analysis performance). Nevertheless, past the threshold of 6k training tokens our model delivers competitive performance in lemmatization and tagging accuracy, while clearly dominating the baselines in terms of analysis accuracy.

Lastly, we would like to comment on performance of LemMED relative to vary-

[8] Except for lemmatization accuracy of Lemming, which drops on the interval of [128k, 256k) training tokens.

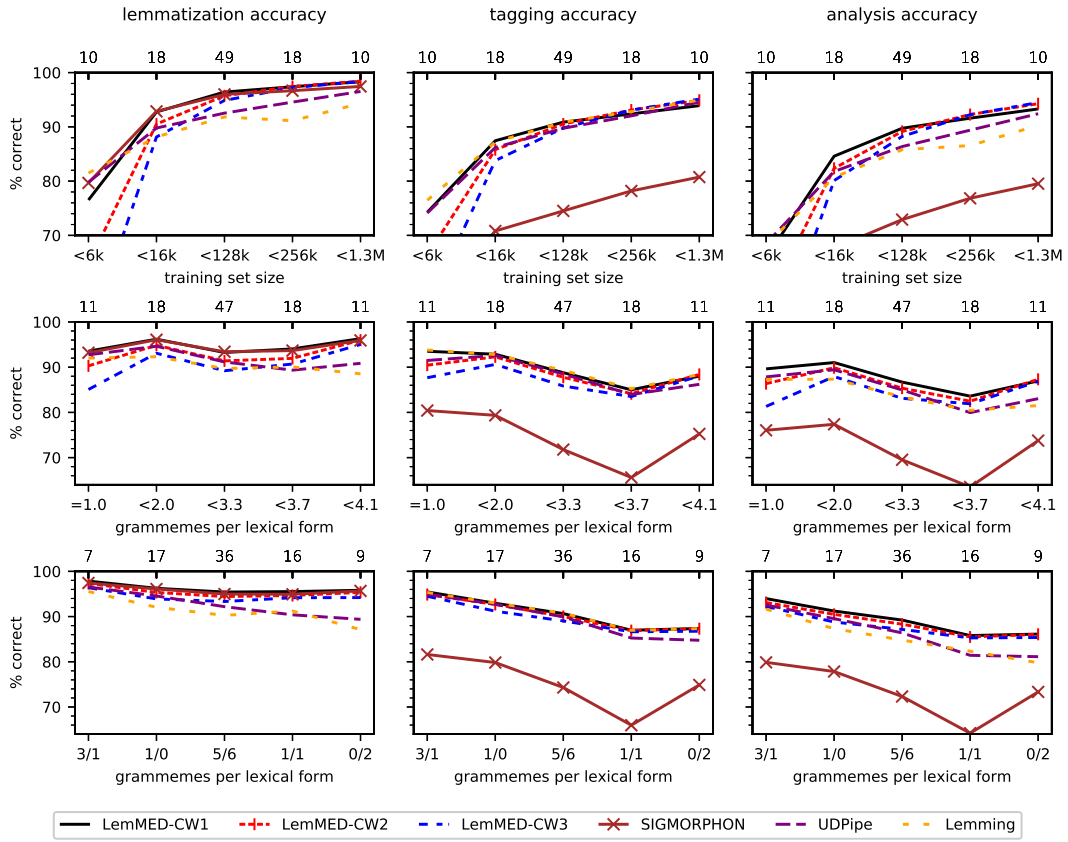


Figure 3.2: The effect that size and morpho-complexity of the data have on performance. The intervals are chosen in a way to achieve normal-like treebank per bin distributions. The number of treebanks in each bin is shown on the top horizontal axes. The bottom row contains the complexity vs performance plots with 10 of the smallest and the largest treebanks removed. Here the bottom horizontal axis shows the number of excluded treebanks in the smallest/largest format.

ing context size. Intuitively, models with larger context windows capture sparser patterns and therefore require more training data to achieve best performance. Indeed, LemMED-CW2 and -CW3 perform poorly on smaller treebanks, but quickly catch up with LemMED-CW1 as more data becomes available. When there are 256k+ training tokens, the models with larger context windows even gain about 2% in tagging and analysis accuracy over LemMED-CW1. Also, single-word windows may not provide enough context to handle morphologically complex data. This is evident from how fast the models that utilize more context close the performance gap with LemMED-CW1 (cf. Figure 3.2, middle row). Even when the outliers are removed, performance of those models drops slower than that of LemMED-CW1, suggesting better resilience

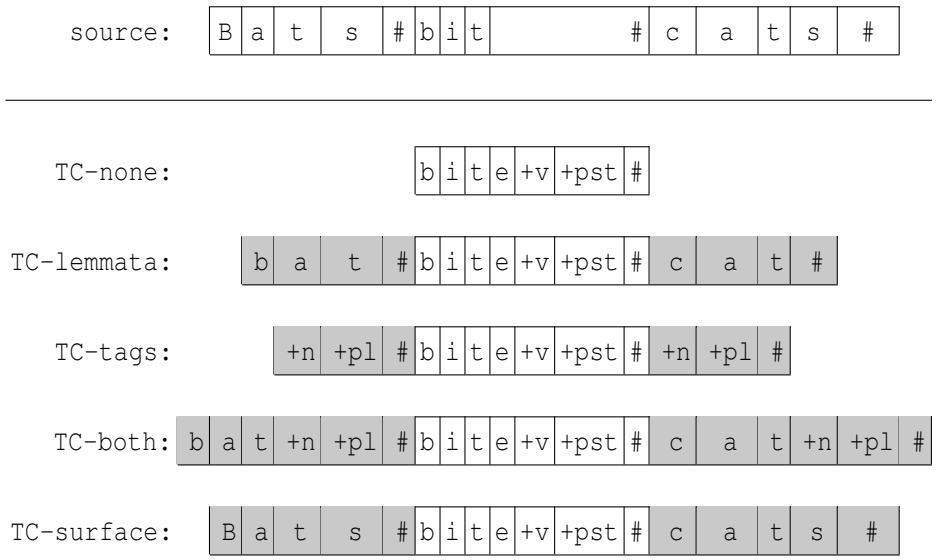


Figure 3.3: Possible target sequences for a given source sequence. Target-side context is highlighted if present.

to increase in morpho-complexity. Thus, in a data rich, morphologically complex scenario, where efficiency is not a concern, using context windows of two or three words may result in 1-2% performance gain over LemMED-CW1. However, in a scenario similar to ours, with many treebanks of varying size and morpho-complexity, using a single word of surrounding context is advisable for efficiency alone. At any rate, for the data set at hand, LemMED-CW1 is clearly the strongest of our models; therefore in the subsequent experiments we use LemMED only in the single-word context window mode.

3.3.3 Target-side Context Variations

So far, while training LemMED, we have been modeling target-side context by including both a lemma and a tag into each target side analysis. However, as was noted in Section 3.1, unless voting on majority prediction is implemented, target side context is ignored at test time; and therefore can be arbitrary modified. Figure 3.3 depicts some of the possible modifications. An obvious one is to do away with target-side context completely (TC-none), so that the model learns to predict only what matters and nothing else. On the other hand, target-side context may not be completely redundant, as it may provide contextual clues during training. Moreover, it might be the case that using only lemmata (TC-lemmata) or tags (TC-tags) tilts performance towards lemmatization or tagging respectively, while TC-both (i.e. current usage) maximizes analysis accuracy.

TC type	LACC	TACC	ACC
None	96.5 (84.8)	91.1 (72.9)	90.2 (67.6)
Lemmata	**97.3** (**85.8**)	91.0 (72.6)	90.2 (67.6)
Tags	96.6 (85.0)	**92.1** (**73.2**)	90.5 (67.0)
Both	**97.3** (85.5)	**92.1** (72.9)	**91.2** (**67.8**)
Surface	96.9 (85.1)	91.3 (72.9)	90.4 (67.5)

Table 3.5: Performance of LemMED-CW1 with respect to the types of target-side context. The results are averaged over the 10 treebank sample.

Lastly, for something that seems truly or, at least, more redundant than lexical context a blatant duplication of surface context can be used. This variation (TC-surface) results in identical surface forms appearing in both source- and target-side context.

To check our intuitions, we again use 10 arbitrarily-chosen typologically balanced treebanks that we used in subsection 3.3.1. We train LemMED-CW1 on those treebanks using all of the target-side context types depicted on Figure 3.3, except "target-both", for which the results are available from the previous experiments. The results reported for development sets are given in Table 3.5. As can be seen, using any type of target-side context is better than using none at all. Indeed, TC-none is beaten by every other variation, including TC-surface, in almost every metric. Also, using TC-lemmata and TC-tags does tilt performance towards lemmatization and tagging respectively, yet not enough to outperform TC-both on those metrics (though, there are gains in OOV accuracy in both cases). Overall, modeling complete target-side context, i.e. using both lemmata and tags, seems to be most beneficial, as TC-both beats or matches every other variation on every metric.

3.3.4 Test Set Evaluation and Shared Task Performance

For the final test set evaluation we use LemMED-CW1 with complete target-side context modeling. In addition, we implement voting on majority prediction and refer to this version of the model as LemMED-V. Bearing in mind sensitivity of our model to data size, we also consider a separate scenario that excludes 10 smallest treebanks, i.e. the treebanks with 6k or less training tokens. The results are shown in Table 3.6.

First, we notice that LemMED-V achieves 0.1-0.3% improvement over the basic model in general performance on all of the metrics, but loses 0.2% and 0.1% in OOV

Method	LACC	TACC	ACC
data, all			
LemMED-V	**94.3** (**78.4**)	89.0 (64.6)	**87.3** (57.3)
LemMED	94.1 (78.3)	88.8 (64.8)	87.0 (**57.4**)
UDPipe	91.5 (70.7)	88.1 (61.0)	84.6 (49.7)
Lemming	90.3 (71.8)	**89.2** (**65.9**)	83.6 (52.3)
SIGMORPH.	94.2 (77.2)	73.1 (62.0)	70.7 (52.0)
data, 6k+ training tokens			
LemMED-V	96.0 (82.4)	90.6 (67.7)	89.2 (61.5)
LemMED	95.9 (82.3)	90.4 (67.7)	89.0 (61.6)
UDPipe	92.7 (71.6)	89.6 (63.5)	86.3 (51.9)
Lemming	91.2 (72.8)	90.6 (68.1)	85.2 (54.3)
SIGMORPH.	95.5 (80.5)	74.9 (64.2)	73.0 (55.6)

Table 3.6: Final results on the test set. LemMED-V denotes the version with voting on majority prediction. Best results overall are shown in bold. Best results on the treebanks with 6k+ training tokens are highlighted.

performance in tagging and analysis accuracy respectively. Second, regardless of the presence or absence of the smallest treebanks, both versions of LemMED improve 2.4-2.9% in analysis accuracy over the next best model (UDPipe). Third, on the complete data set our model outperform SIGMORPHON in lemmatization only if used in the voting mode. However, when the smallest treebanks are removed LemMED-CW1 beats this baseline and, when it comes to OOV lemmatization accuracy, rather decisively. Fourth, Lemming proves to be a very strong tagging baseline, as only on the bigger treebanks can LemMED-V achieve a matching tagging accuracy, while still losing in tagging accuracy on OOV. Overall, the conclusions we drew from a number of dev set evaluations are valid here as well. While performing slightly better or slightly worse than the strongest baselines in lemmatization and tagging, LemMED is definitely better at predicting whole lexical forms, especially on unseen data.

Now we would like to report on our model's Shared task performance, where we have submitted the version of LemMED-CW1 used in subsection 3.3.4 (without voting on majority prediction). Table 3.7 compares the results of the competing systems ranked by the average value calculated over the four metrics used in the shared task. To save space, we consider only best performing versions among multiple submissions

#	System	LACC	LDIS	TACC	T-F1	AVG
1	Straka et al. (2019)*	95.78	0.10	93.19	95.92	93.76
2	Kondratyuk (2019)*	95.00	0.11	93.23	96.02	93.37
3	Üstün et al. (2019)*	93.91	0.14	90.53	94.54	91.30
4	LemMED	94.20	0.13	88.93	92.89	90.64
5	Shadikhodjaev and Lee (2019)	94.07	0.13	88.09	91.84	90.33
6	Yildiz and Tantuğ (2019)	94.46	0.11	86.67	90.54	90.22
7	Oh et al. (2019)	93.43	0.17	87.42	92.51	89.14
8	Baseline (Malaviya et al., 2019)	94.17	0.13	73.16	87.92	85.60

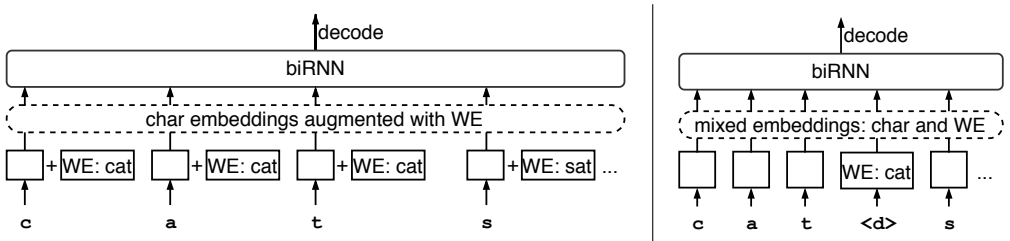
Table 3.7: Shared task performance. The systems are ranked by descending average over the four metrics. When computing the average LDIS is scaled as $100 * (1 - LDIS)$. Here: * denotes usage of pre-trained contextualized embeddings.

by the same team. We also exclude submissions that performed either lemmatization or tagging (not both) and those that were evaluated only on a subset of data. A version of the table with the results across all submissions is available in the overview paper by McCarthy et al. (2019).

Overall, LemMED is ranked fourth in terms of average performance, tagging accuracy and tagging F-score. It is tied for fourth lemmatization distance and is ranked fifth in lemmatization accuracy. However, if we ignored the systems that use contextualized embeddings, our model would rank first or second on all metrics. Thus, for a straightforward model that it is, LemMED achieves a competitive performance among systems that do not utilize external resources. Unfortunately, the organizers do not report analysis accuracy and results on unseen data, which would be interesting to compare, given that those are the two strongest aspects of LemMED's performance.

3.4 Incorporating Contextualized Embeddings

As was seen in the previous section, the shared task systems that utilize contextualized embeddings performed very well. This presumably allows capturing structure on both levels: morphological (via characters) and syntactic (via words). Importantly, word-level representations take precedence and final predictions are generated per word. Our model, on the other hand, employs "character-to-character" approach and, given a sequence of characters, generates another sequence, outputting at each step a lemma



(a) Character-only embeddings augmented with WE, BERT-Concat

(b) Mixed-level embeddings, BERT-Insert; <d> denotes a corresponding word-piece

Figure 3.4: Example embedding of the phrase "cat sat" with proposed representations.

character or a grammeme (part of a tag), presumably oblivious of word-level syntax.

As a possible compromise, we propose augmenting character embeddings with word embeddings by concatenating BERT embeddings Devlin et al. (2018) of a given word to each of its characters ("character+word-to-character" approach, **`cw2c`**, cf. Figure 3.4a). A drawback of this approach is that it seems highly redundant, as the same word-piece (sub-word units used in BERT) would have to be used multiple times for a given word. To minimize redundancy we concatenate only corresponding word and character embeddings, e.g if the word "table" is broken into word-pieces as "#tab' and "#le" by the BERT tokenizer, then each of the first three characters (the ones that form "tab") will be concatenated with the first word-piece and each of the last two with the second one. Another alternative representation is a plain mix of character and word embeddings (cf. Figure 3.4b). Here BERT word-pieces will be inserted right after character embeddings that make up a corresponding word. We refer to the first representation as BERT-Concat and to the second one as BERT-Insert.

For this experiment we modify the code of OpenNMT-py (Klein et al., 2017) to accommodate support for Huggingface implementation of BERT embeddings (Wolf et al., 2020). We retrain LemMED-CW1 using the same hyper parameters as before. As this experiment is more time consuming, we use the same sample of ten typologically balanced treebanks that we used to evaluate LemMED's performance in full sequence mode (cf. sub-section 3.3.1).

Results are given in Table 3.8. As it can be seen, both representations perform on par with each other, achieving only marginal improvement over the original model on certain treebanks and performing almost equally on average. When we looked at the training logs we found that the model converges slower than it does without BERT. We have tried to change hyper-parameters including using smaller character embeddings

Treebank	Accuracy, %		
	LemMED-CW1	BERT-Concat	BERT-Insert
Arabic-PADT	**89.4**	89.3	89.3
Hebrew-HTB	**94.6**	94.5	94.3
Finnish-FTB	**93.2**	93.0	92.8
Bulgarian-BTB	95.6	**95.7**	95.5
Chinese-GSD	89.7	90.4	**90.9**
Russian-GSD	**87.7**	**87.7**	87.2
English-LinES	93.8	93.7	**93.9**
Afrikaans-AfriBooms	96.5	**97.0**	96.6
Turkish-IMST	**90.7**	90.5	90.5
Hungarian-Szeged	**81.1**	80.4	79.9
Average	**91.2**	**91.2**	91.1

Table 3.8: Incorporation of BERT embeddings.

and more training steps, but it did not have significant effect on the results.

3.5 Summary

We presented LemMED, a character-level encode-decoder with attention applied to contextual morphological analysis. We showed how it extends similar attention-based models developed for lemmatization (Lematus) and non-contextual inflection (MED). Upon investigating the role of context, we have concluded that using small context windows to model short sequences is preferable to modeling whole sentences. It was found that a choice of specific window size may depend on characteristics of the data, such as size and morpho-complexity. In our case using a single-word context window yielded best average performance, while maintaining an acceptable efficiency of two hours of training per treebank. In addition to context size, we experimented with various types of target-side context and concluded that it should be modeled in full, i.e. including both lemmata and tags. We evaluated LemMED on a set of more than a hundred treebanks, comparing it to both popular baselines available off-the-shelf and state of the art systems submitted to the SIGMORPHON-2019 shared task on contextual analysis. In both cases our model delivered a competitive performance and when compared to the baselines showed clear dominance in terms of combined lemmatization and tagging accuracy and performance on unseen data. Finally we showed that

LemMED could not be improved by incorporation of pre-trained BERT embeddings.

Chapter 4

Word-level Context Restriction

In this chapter we experiment with word-level context restriction. To this end we train an off-the-shelf implementation of a stacked LSTM network. Here, we work with a small sub-set of the data we used in our experiments with character-level context. This allows us to better control for linguistic and domain variation, as we experiment with incremental increase in data size. We conclude that word-level context restriction only makes sense for agglutinative and fusional languages in low resource scenarios. We also investigate shortcomings of the restricted model and find that, when there is enough training data it is less likely (compared to an unrestricted counterpart) to correctly analyze words, whose dependency heads are left outside the context window. Finally, we find that performance gains from context restriction can be achieved at inference time, without having to train the model in the restricted mode.

4.1 Model

UDPipe 2.0 (Straka et al., 2019) is a multi-task model that uses a multi-layer bidirectional LSTM network to jointly predict lemmata and morpho-tags. Input to the LSTM network consists of a combination of at least two types of embeddings: end-to-end word-level embeddings and character-level word embeddings, i.e. an output of a bidirectional character-level GRU applied to every word, as per Ling et al. (2015). In addition pre-trained regular word embeddings and BERT embeddings Devlin et al. (2018) can be used (cf. Figure 4.1). For each input word the system outputs an edit script that when applied to the word produces the lemma, as per Chrupała et al. (2008), and a morpho-tag. Morpho-tags are predicted as whole units, but grammatical categories and their values are used for regularization during training.

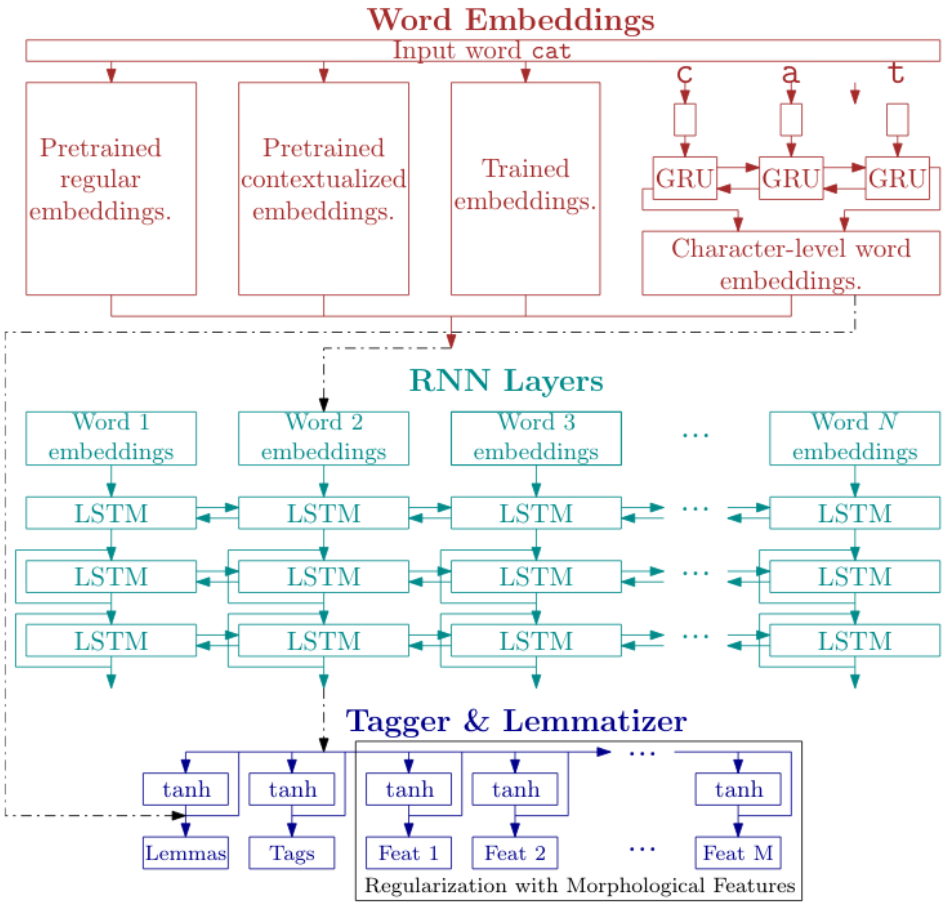
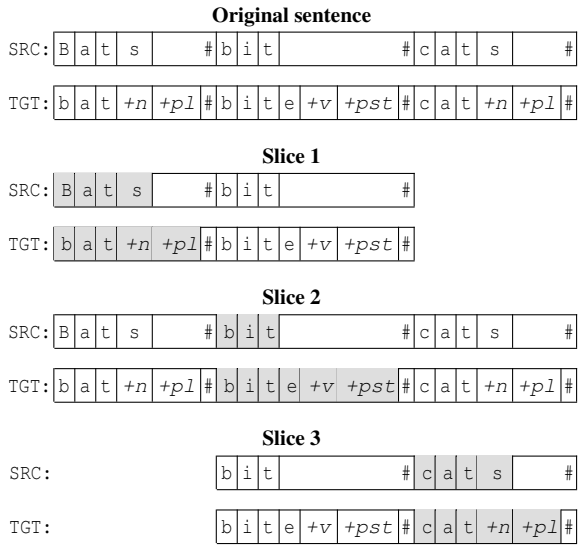


Figure 4.1: UDPipe 2.0 architecture (Straka et al., 2019)



(a) LemMED

ID	FORM	LEMMA	UPOS	FEATS	HEAD	DEP
			Original dependency tree			
1	Bats	bat	NOUN	N=Plur	2	nsubj
2	bit	bite	VERB	T=Past	0	root
3	cats	cat	NOUN	N=Plur	2	dobj
			Tree 1			
1	Bats	bat	NOUN	N=Plur	0	root
2	bit	bite	VERB	T=Past	1	dep
			Tree 2			
1	Bats	bat	NOUN	N=Plur	0	root
2	bit	bite	VERB	T=Past	1	dep
3	cats	cat	NOUN	N=Plur	1	dep
			Tree 3			
1	bit	bite	VERB	T=Past	0	root
2	cats	cat	NOUN	N=Plur	1	dep

(b) UDPipe

Figure 4.2: Example of context restriction implemented for LemMED and UDPipe. In both cases shaded areas denote window-center tokens. For LemMED: "`SRC:`", "`TGT:`" denote source and target sequences, respectively; leading "+" denotes a grammeme; "#" denotes a word boundary.

To better explain the context restriction for UDPipe 2.0 we use the example from Chapter 3, cf. Figure 4.2a. Being a seq2seq model, during training LemMED is given a source and a target sequence, where the former is a sentence split into characters and the latter is the corresponding sequence of morphological analyses also split into characters and grammemes. To limit the given input to one-word of context, we slide a window of three tokens (one central and one on each side) over each input token to produce three slices. We carry out this procedure on all data splits, i.e. training, dev, and test sets, and evaluate the predictions based only on the output given for window-center tokens, unless only one slice is produced for a sentence, in which case all output tokens are evaluated. This happens, when a window is larger than a sentence and all produced slices are identical to the input sentence.[1] Having understood context restriction for LemMED it is straightforward to adapt it for UDPipe, cf. Figure 4.2b. Here, the input is a dependency tree in CoNLL-U format. Again, a window of a desired context size is slid over each token, so that each tree across all data splits gets sliced up into several separate trees. Naturally, only window-center tokens are counted during evaluation, with the exception of single-slice-per-sentence cases described above.

[1]For instance, if we apply a window of five tokens to our example sentence.

Treebank	Typology	Size	LDIS	GPF	EC
Balanced					
Finnish-F	Agglutinative	127.5	1.7	3.4	42.7
Hungarian	Agglutinative	33.5	0.9	3.3	36.0
Turkish-I	Agglutinative	46.4	2.1	4.1	77.0
Afrikaans	Analytic	38.8	0.4	1.9	-
Chinese-G	Analytic	98.7	0.0	1.1	-
Bulgarian	Fusional	124.7	1.4	3.7	22.6
English-L	Fusional	66.4	0.6	2.1	5.1
Russian-G	Fusional	80.0	1.0	4.1	16.7
Arabic-PA	Templatic	225.5	2.8	3.3	33.9
Hebrew	Templatic	129.4	0.5	2.4	27.1
Smallest					
Komi Zyrian-I	Agglutinative	0.8	1.4	3.2	-
Komi Zyrian-L	Agglutinative	1.7	1.4	3.2	-
Marathi	Agglutinative	3.1	1.7	3.6	-
Tagalog-T	Agglutinative	0.2	0.8	2.3	-
Cantonese	Analytic	5.1	0.0	1.0	-
Chinese-C	Analytic	5.7	0.0	1.0	-
Yoruba	Analytic	2.2	0.1	1.5	-
Lithuanian-H	Fusional	4.3	1.2	3.9	22.6
Sanskrit-U	Fusional	1.4	1.8	3.5	36.3
Akkadian-P	Templatic	1.4	4.4	1.0	-

Table 4.1: Data sets statistics. Here: Size – training set size in thou. tokens; LDIS – average Levenstein distance between form and lemma; GPF – average number of grammemes per form; EC – average enumerative complexity, i.e. average paradigm size.

4.2 Experimental Setup

We work with Shared Task data, which is a collection of 107 Universal Dependencies (Nivre et al., 2016) treebanks converted into UniMorph (Kirov et al., 2018) schema. From there we chose two sets of treebanks: (i) typologically balanced set of 10 treebanks that cover agglutinative (Finnish, Hungarian, Turkish), analytic (Afrikaans, Chinese), fusional (Bulgarian, English, Russian), and templatic (Arabic, Hebrew) languages; (ii) 10 treebanks with the smallest training sets. To work with UDPipe we

convert both data sets from UniMorph back to UD format, using the compatibility mapping[2] between the formats in reverse. To experiment with dependency-range-based evaluation (cf. sub-section 4.3.2), we also restore dependency relations and head information, using original UD sentence IDs preserved in Shared Task data.

Table 4.1 lists basics statistics for both data sets with every other group of typologically[3] similar treebanks [4] highlighted for convenience. Here, LDIS and GPF scores provide an intuition on respective difficulty of lemmatization and tagging for the given treebank,[5] where respective lower bounds of 0 and 1 indicate that the tasks are reduced to copying (form into lemma) and POS-tagging respectively. As an approximate measure of morphological complexity, we also report enumerative complexity Ackerman and Malouf (2013); Cotterell et al. (2019), i.e. an average number of surface forms that can be produced for a given pair of lemma and part of speech of a language. The metric is calculated for languages present in UniMorph Kirov et al. (2018) data set, without distinguishing nominal and verbal paradigms.

For all experiments we use the following set up. UDPipe is used with the default parameters and without pre-trained embeddings, either regular or contextualized. The only parameter we change is the number of training epochs. UDPipe has an option to specify multiple learning rates and a number of epochs to train for with each learning rate. The defaults are 40 epochs with rate 0.01 and 20 epochs with rate 0.001. We keep the learning rates, but vary number of epochs per rate depending on context size. Since the model tends to converge faster, when trained on shorter sequences, we start from 20 and 10 epochs per each learning rate, when using context of 0 and 1 surrounding words. For each additional word of context we increase number of epochs per rate by 10 and 5 respectively. Since we do not train models that use more than 3 words of context, in the full context mode we end up training for 50 (20 + 30) and 25 (10 + 15) epochs per learning rate.

We use LemMED with the default parameters. We train the system for 50k training steps, halving the initial learning rate of 1.0 every 10k steps starting from the step #25k. The systems saves a checkpoint every 1000 steps, producing 50 saved models, which are compared on the dev set and the best one is used for evaluation.

[2] `https://github.com/unimorph/ud-compatibility`

[3] Information on language typology is taken from UniMorph repo (`https://unimorph.github.io/`), or general sources, if absent in the former.

[4] In case if there are multiple treebanks for a language in Shared Task data, a prefix of the sub-name is given to uniquely identify the treebank, e.g. Finnish-F denotes Finnish-FTB.

[5] To account for variety in treebank sizes, these scores were calculated on 1000 and 100 random dev set tokens (excluding punctuation) of balanced and smallest data sets respectively.

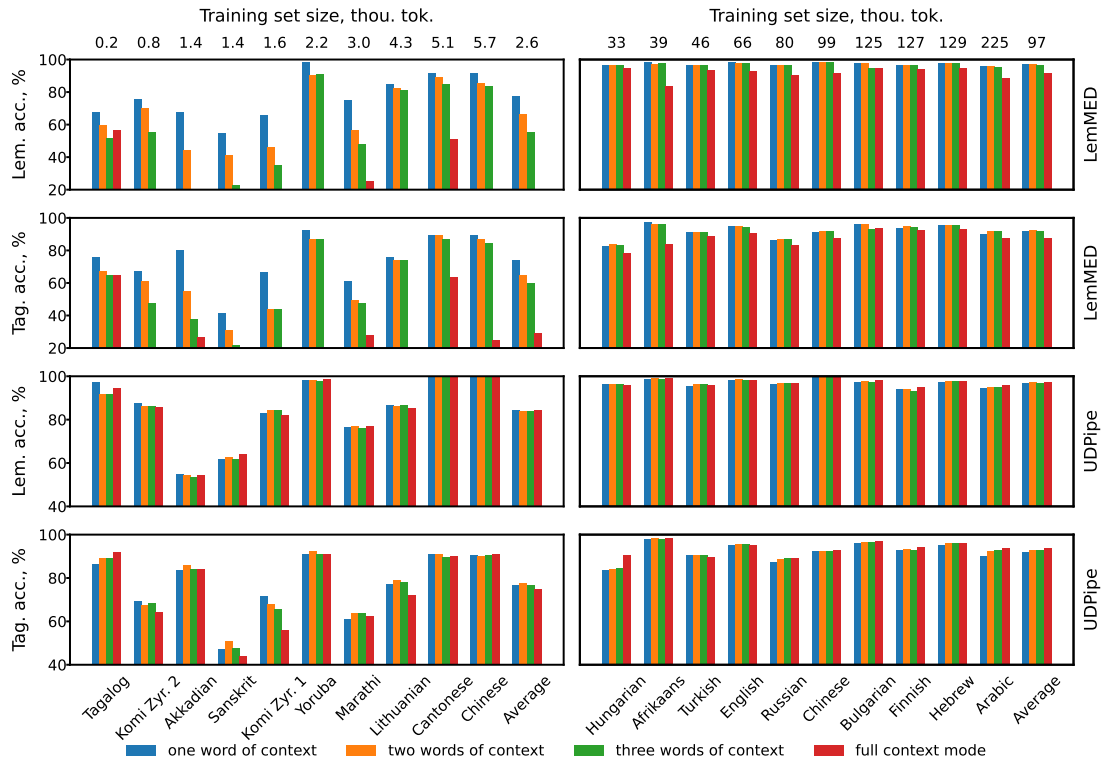


Figure 4.3: Performance of LemMED and UDPipe on smallest (left) and balanced (right) data sets.

For both systems we report test set results, evaluating performance in terms of exact match lemmatization and tagging accuracy. In some experiments for clarity of exposition we use a single metric, analysis accuracy, which deems correct the outputs for which both the lemma and the tag were correctly predicted.

4.3 Results and Discussion

4.3.1 Context Size vs Data Size

One of the main findings of the original work on LemMED was the fact that using local instead of global context improved performance, particularly in data-poor scenarios, where best results were achieved by using just one word of surrounding context. We want to assess this claim for UDPipe that, in contrast to LemMED, uses word-level context. To this end, we start by observing how the behavior of both models changes in context-restricted vs -unrestricted mode with respect to data size.

From Figure 4.3 we can see that, on the smallest treebanks, LemMED, indeed,

achieves best results by using just one word of context. On this data set performance of the models that use more context is considerably lower, especially for morphologically complex languages, suggesting that the latter benefit the most from context restriction. This intuition, however, is not supported by the experiment on the balanced data set with larger treebanks, where the models limited to different amounts of context perform more or less on par with each other, regardless of typological properties of the data. In general, on both data sets LemMED achieves higher accuracy in terms of both, lemmatization and tagging, when using local context.

When it comes to UDPipe, on the smallest data set, context restriction brings noticeable performance gains only on the treebanks of morphologically complex languages and only in terms of tagging accuracy. Thus, context restriction on word level can be useful in data poor scenarios, although benefits are much more modest compared to character-level context restriction. At the same time, limiting context does not hurt performance on the balanced data set, where, with few exceptions, restricted models perform on par with the unrestricted ones on both metrics. This suggests that amount of context has little effect on lemmatization and tagging, which is counter intuitive and requires further investigation.

In the previous experiment we have performed context-restriction on smaller and larger treebanks, to illustrate the effect of data size. However, not only there is still great variance in size within each data set, but there is also almost no linguistic overlap between the two. The only common language, Chinese, is represented by a treebank of learner essays in one data set and by a treebank of Wikipedia articles in another. Thus, to control for language and domain, from each treebank of the balanced data set, we sub-sample seven training sets, such that each consecutive set contains the previous smaller one. Sample sizes are set to be 1, 2, ... 2^xk ..., 32k, 32k+ tokens, where "32k+" denotes entire training set. We train UDPipe with various context sizes on each sub-sample and compare performance across increasing data size. This time we also experiment with context size of zero (i.e. no context) and evaluate the models and the most frequent baseline in terms of analysis accuracy. Hereinafter, for brevity, we will write model-0, -1, -2, -3 to to the models that use 0, 1, 2, 3 words of context respectively. The unrestricted model will be referred to as model-f.

As it can be seen from Figure 4.4, on all languages, except Chinese and Arabic, until certain amount of training data model-1[6] outperforms model-f. On analytic lan-

[6]For clarity of presentation, we do not plot performance of model-2 and -3, as they follow the same pattern, although perform slightly better than model-1.

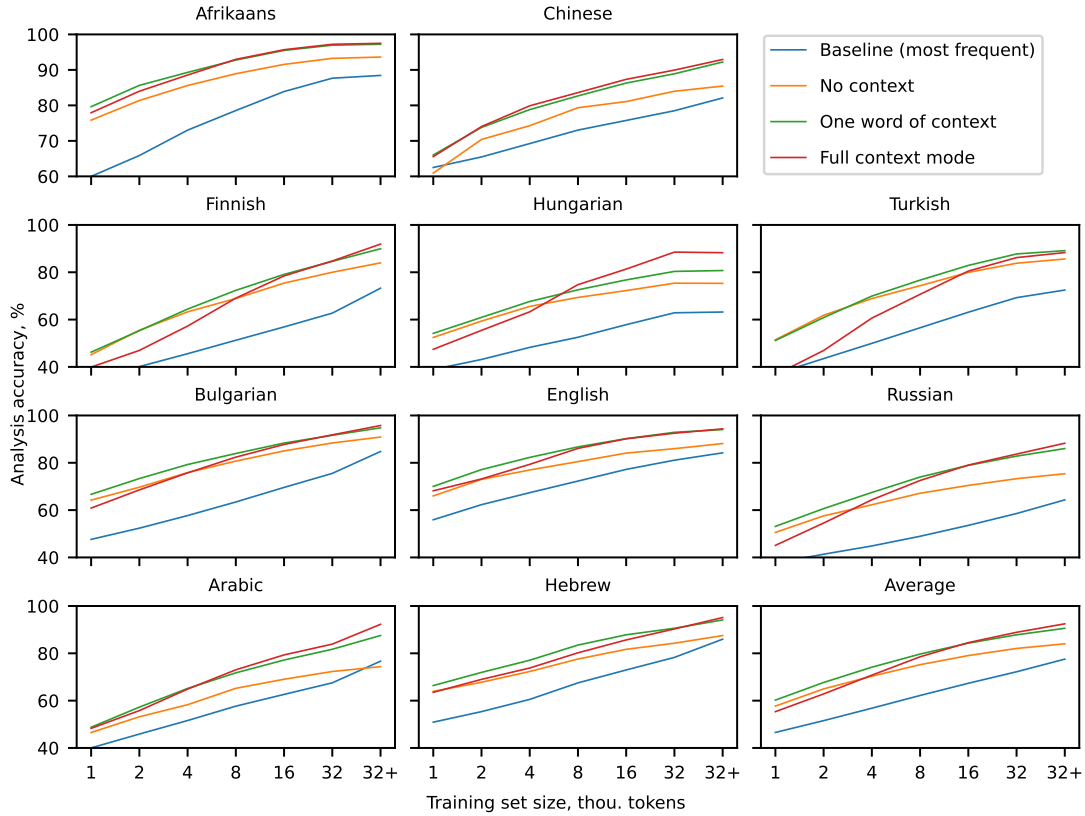


Figure 4.4: Performance of UDPipe vs gradual increase in training sets of the balanced set

guages (top row) this happens to Afrikaans, where a narrow performance gap between the models is maintained until 8k training tokens. Conversely, for Chinese, model-f beats model-1 narrowly starting from 2k training tokens. For both languages, model-0 is consistently worse than the other two models, but better that the baseline.

As we have seen from the previous experiment, the gap in performance between models -1 and -f is the largest and is maintained for longer on agglutinative languages (second row from the top). Here, even model-0 outperforms model-f almost as much and as for long as model-1. Considering that these languages are the most morphologically complex (have the highest enumerative complexity, cf Table. 4.1) in the data set, the model might need more data just to learn something about morphotactics, i.e. word structure. Fusional languages (third row from the top) behave similar to the agglutinative ones, but improvement of models-0 and -1 over model-f is less pronounced. For two templatic languages (bottom row) there appears to be no common pattern in performance. On Arabic model-f beats model-1 starting from 8k training tokens, and is clearly better when trained on all data (32k+). In contrast, on Hebrew model-1 out-

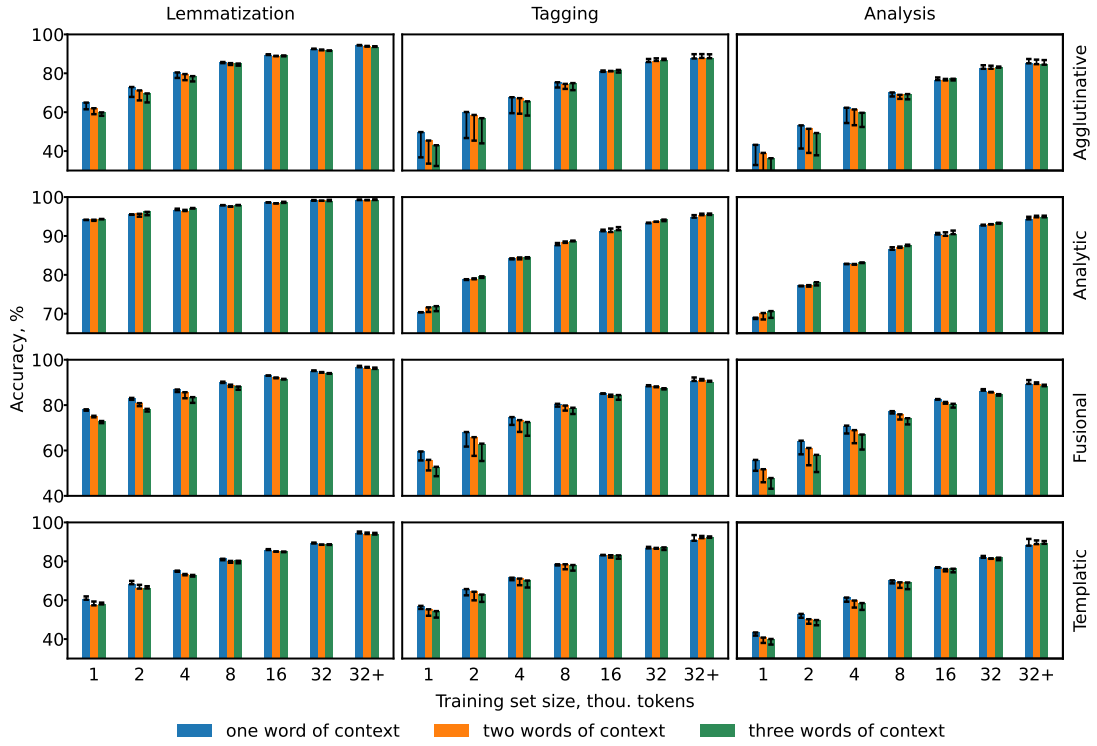


Figure 4.5: Performance of the restricted models on orphan tokens. Error bars indicate net increase/decrease in performance of the corresponding unrestricted models.

performs model-2 by almost constant score up until 32k training tokens. Model-0 and the baseline, however, behave more-or less similarly in both cases.

From this experiment we can see that, depending on typological properties of the data, using local context can be more or less beneficial, when training data is scarce. However, with the exception of Hungarian and Arabic, limiting context size does not seem to result in heavy performance loss, suggesting that global context does not offer anything which cannot be learned from local context. To see if this is indeed the case, in the next experiment we investigate the relationship between context size and syntactic dependency range.

4.3.2 Context Size vs Dependency Range

An obvious weakness of context restriction is that the model gets denied access to contextual cues, especially useful for tagging, that get left out of the context window. To quantify how much accuracy is potentially lost this way, we design the following experiment. First, we measure the accuracy of the restricted models *only* on the tokens, whose dependency heads are left outside of the context window. We will refer to

such as orphan tokens. Then, on the same tokens we measure the accuracy of the corresponding unrestricted models, which, by definition, can use contextual cues from the whole sentence to analyze those tokens. From UD relations list we exclude those, which "are not dependency relations in the narrow sense",[7] e.g. punct, dep, parataxis, etc.

Figure 4.5 shows the results of the experiment across three metrics averaged over languages grouped by typological properties. First let us note, that apart from two smallest agglutinative data samples, in terms of lemmatization f-models seem to perform slightly better on all orphan tokens, regardless of data size and typology. Likewise, analysis accuracy by and large mirrors that of tagging. Thus, we will focus on the latter.

As we can see, on agglutinative languages f-models actually perform worse, up until 8k training tokens, with net decrease ranging from 20% to 10% on data spanning 1k to 4k training tokens. Starting from 16k tokens, however, f-models display modest improvement of up-to 2% on 32k+ samples. This trend is more or less repeated by the performance of f-models on fusional and templatic languages, with smaller decrease/increase in accuracy. On the analytic languages, f-models perform slightly worse, by about 2%, on the sample of 1k training tokens and gradually improve after that. Overall, we can conclude that for morphologically complex languages, in the face of data scarceness, the unrestricted models do not seem to be able to make good use of long-range contextual cues. However, unlike the restricted models, they are able to utilize those contextual cues, given enough training data.

4.3.3 Training vs Inference Context Size

In this section we perform a follow-up experiment to the previous one, where we saw that the unrestricted models struggled with orphan tokens in data poor scenarios, whereas the restricted models did very well. What if that has nothing to do with the way how the models were trained, i.e. in restricted or unrestricted manner, and instead depends on context size during inference?

To answer this question, to each model trained with a given context size we have fed as input test sets of every other context size. Thus for each treebank, excluding Arabic,[8] we ended up with 16 results per a training set sample. Figure 4.6 shows average

[7] `https://universaldependencies.org/u/dep/all.html`

[8] We had technical issues with this particular treebanks, where this cross-validation experiment worked on some sample sizes and crushed on others.

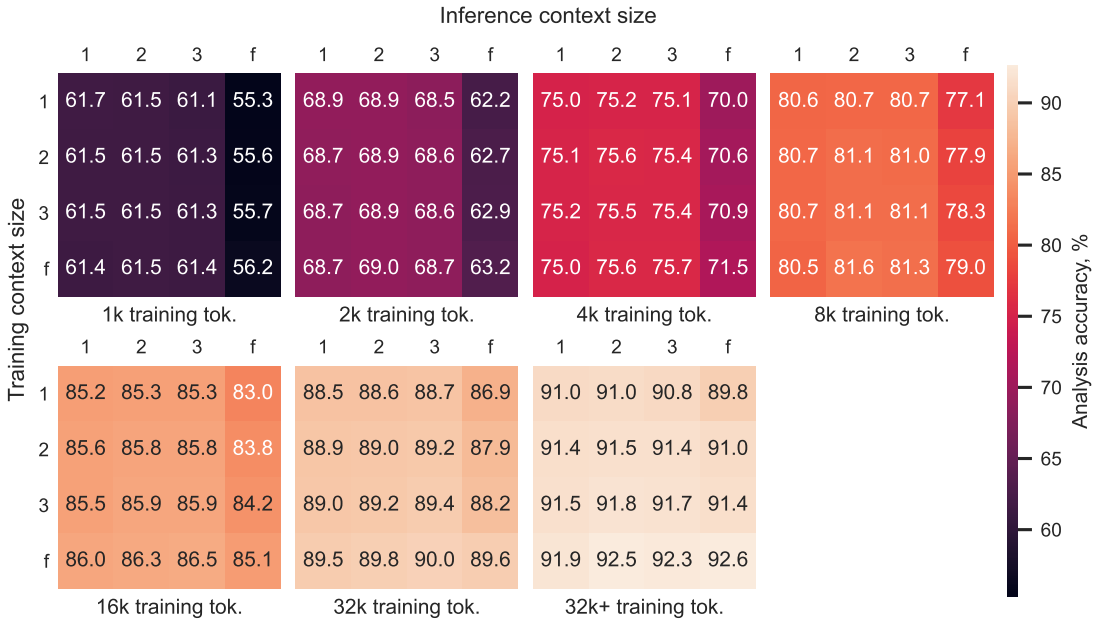


Figure 4.6: Performance of the models across training and inference context sizes.

analysis accuracy computed across all treebanks and training-inference context size combinations. If the best results are produced by training and predicting on the same context-sizes we expect to see diagonal values to be the largest across corresponding rows and columns. Surprisingly we find that it is not the case, certainly not for the f-model. It appears that, up until 32k+ training tokens, the models trained on global context benefit from being fed context-restricted input. So much so that they perform at least as good as and, starting from 16k training tokens, even better than the models whose training and inference context sizes match. Consider 16k training tokens heat map. Here f-model achieves 86.0% accuracy on one-word-of-context input, while f-1 model achieves only 85.2% on its "native" input. Similarly f-model outperforms models-2 and -3, when fed corresponding context-restricted input.

This result supports our pro-restriction argument laid out in Chapter 1. There, citing recent findings in context-restricted language modeling, we speculated that, LSTMs might have some "effective range of history", beyond which there is not much to be learnt from context. We also suggested that, perhaps, this effective range might shrink with decrease in training set size, because context becomes sparser. Clearly, until data is sufficiently large, unrestricted models perform better on local context, suggesting that they learn more from local rather than distant context.

4.4 Summary

In this work we have investigated the application of word-level context restriction strategy to an LSTM-based joint morphological analyzer. In accordance with our experiments on character-level context, we have found that in data-poor scenarios context-restriction improves performance of word-level models as well, especially on morphologically complex languages. We have shown that, models trained on global context do capture long-range dependencies that context-restricted models miss, but they do that only when given enough training data. Finally, we find that by restricting context during inference, it is possible to achieve the same or even better results than by restricting context during training. This is a particularly interesting finding, which we would like to investigate in the future. All of the findings were made for a model that makes no use of any kind of pre-trained embeddings. We would also like to see, how incorporation of such embeddings affects our findings.

Chapter 5

Conclusion

Context is essential for many NLP tasks. In the early days of the discipline, usage of context was almost exclusively limited to a few preceding or surrounding words due to both design of models and data sparseness issues. With ubiquitous adoption of neural approaches, however, the situation became completely reversed, as usage of global (sentence-level) context became a de facto standard. In this thesis we evaluated utility of local context for neural models on the task of morphological analysis. Our findings suggest that context restriction can be beneficial on both character- and word level.

In Chapter 1 we presented an argument for context restriction, citing work on language modeling which suggest that LSTM models might have certain effective range, beyond which context might be less valuable. We proceed by discussing some of the challenges of contextual morphological analysis, its importance for down-stream applications, and fate it shared with other NLP tasks in being denied restriction to local context.

Chapter 2 provided relevant background on morphology and morphological typology, as well as morphological analysis and related tasks. We show that few exceptions existing neural approaches do not employ context restriction.

In Chapter 3 we presented our character-level model and reported that it: (i) achieves both better performance and efficiency in context restricted mode; (ii) should be limited to as little context as possible in low resource settings; (iii) can compete with world-level context models that do not use pre-trained contextualized word embeddings; (iv) cannot be improved via incorporation of pre-trained contextualized word embeddings.

In Chapter 4 we confirmed some of our previous findings for word-level context restriction. Namely, we showed that local context can indeed be useful, but only for agglutinative and fusional languages in low resource scenarios. We tried to explain

some of the limitations of context restriction and concluded that not having access to long range dependencies hurts performance when data is sufficiently large. We also found support for our argument on "effective range" of LSTMs, by showing that in low resource scenarios, models trained on global context work better on restricted input.

5.1 Future Work

Immediate follow up work might include further investigation on augmenting LemMED with word embeddings. One suggestion would be to use a single embedding not per character or per word, as was tried, but per window. A possible implementation might include an additional biRNN layer over the words in the window, which outputs a single vector containing word-level context information. This vector can then be inserted after the last character embedding in the window. This should be tried with both regular and contextualized embeddings. Also, existing framework for incorporation word embeddings might be tried with regular embeddings too.

In terms of, word-level context, the experiment with inference-time context restriction should be tried on larger context windows, to see if there is a pattern of growth and slump. This could shed more light on the "effective range" speculations. Additionally, since the results we presented were averaged across languages, per-language investigation should be conducted to see, if there are any language-specific peculiarities.

On a more general level, it could be useful to see if our findings can be confirmed across sequence labeling tasks and different architectures, that involve other RNN implementations. Perhaps, starting with something easier, say POS-tagging, would allow for designing better controlled experiments.

Bibliography

Ackerman, F. and Malouf, R. (2013). Morphological organization: The low conditional entropy conjecture. *Language*, pages 429–464.

Aduriz, I., Urkia, M., Alegria, I., Artola, X., Ezeiza, N., and Sarasola, K. (1997). A spelling corrector for Basque based on morphology. *Literary and Linguistic Computing*, 12(1):31–38.

Agirre, E., Alegria, I., Arregi, X., Artola, X., de Ilarraza Sanchez, A. D., Maritxalar, M., Sarasola, K., and Urkia, M. (1992). XUXEN: A Spelling Checker/Corrector for Basque Based on Two-Level Morphology. In *ANLP*, pages 119–125. ACL.

Aiken, B., Kelly, J., Palmer, A., Polat, S. O., Rama, T., and Nielsen, R. (2019). SIGMORPHON 2019 Task 2 system description paper: Morphological analysis in context for many languages, with supervision from only a few. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 87–94, Florence, Italy. Association for Computational Linguistics.

Albright, A. (2000). The Productivity Of Infixation In Lakhota.

Arad, M. (2005). *Roots and Patterns: Hebrew Morpho-syntax*, volume 63.

Assylbekov, Z., Washington, J., Tyers, F., Nurkas, A., Sundetova, A., Karibayeva, A., Abduali, B., and Amirova, D. (2016). A free/open-source hybrid morphological disambiguation tool for Kazakh.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Beesley, K. R. (1991). Computer Analysis of Arabic Morphology: A two-level approach with detours.

Beesley, K. R. and Karttunen, L. (2003). *Finite state morphology*. CSLI Publications, Stanford, Calif.

Bender, E. M. (2013). *Linguistic Fundamentals for Natural Language Processing: 100 Essentials from Morphology and Syntax*. Morgan Claypool Publishers.

Bergmanis, T. and Goldwater, S. (2018). 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)*, pages 1391–1400, New Orleans, Louisiana. Association for Computational Linguistics.

Cardenas, R., Borg, C., and Zeman, D. (2019). CUNI–malta system at SIGMORPHON 2019 shared task on morphological analysis and lemmatization in context: Operation-based word formation. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 104–112, Florence, Italy. Association for Computational Linguistics.

Chakrabarty, A., Pandit, O. A., and Garain, U. (2017). Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1481–1491, Vancouver, Canada. Association for Computational Linguistics.

Chaudhary, A., Salesky, E., Bhat, G., Mortensen, D. R., Carbonell, J., and Tsvetkov, Y. (2019). CMU-01 at the SIGMORPHON 2019 shared task on crosslinguality and context in morphology. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 57–70, Florence, Italy. Association for Computational Linguistics.

Chelba, C., Norouzi, M., and Bengio, S. (2017). N-gram language modeling using recurrent neural network estimation. *CoRR*, abs/1703.10724.

Chrupała, G., Dinu, G., and van Genabith, J. (2008). Learning morphology with Morfette. In *LREC 2008*.

Chrupała, G. (2006). Simple data-driven context-sensitive lemmatization. *Proces. del Leng. Natural*, 37.

Cotterell, R., Kirov, C., Hulden, M., and Eisner, J. (2018a). On the complexity and typology of inflectional morphological systems. *CoRR*, abs/1807.02747.

Cotterell, R., Kirov, C., Hulden, M., and Eisner, J. (2019). On the complexity and typology of inflectional morphological systems. *Transactions of the Association for Computational Linguistics*, 7:327–342.

Cotterell, R., Kirov, C., Sylak-Glassman, J., Walther, G., Vylomova, E., McCarthy, A. D., Kann, K., Mielke, S. J., Nicolai, G., Silfverberg, M., Yarowsky, D., Eisner, J., and Hulden, M. (2018b). The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27, Brussels. Association for Computational Linguistics.

Cotterell, R., Kirov, C., Sylak-Glassman, J., Walther, G., Vylomova, E., Xia, P., Faruqui, M., Kübler, S., Yarowsky, D., Eisner, J., and Hulden, M. (2017). CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver. Association for Computational Linguistics.

Cotterell, R., Kirov, C., Sylak-Glassman, J., Yarowsky, D., Eisner, J., and Hulden, M. (2016). The SIGMORPHON 2016 shared Task—Morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Erjavec, T. and Dzeroski, S. (2004). Machine learning of morphosyntactic structure: Lemmatizing unknown slovene words. *Applied Artificial Intelligence*, 18:17 – 41.

Eryiğit, G. and Oflazer, K. (2006). Statistical dependency parsing for Turkish. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Ezeiza, N., Alegria, I., Arriola, J., Urizar, R., and Aduriz, I. (1998). Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 380–384, Montreal, Quebec, Canada. Association for Computational Linguistics.

Forcada, M. L., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Sánchez-Martínez, F., Ramírez-Sánchez, G., and Tyers, F. M. (2011). Apertium: A free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

Fromkin, V., Rodman, R., Hyams, N., Collins, P., Amberber, M., and Cox, F. (2017). *An introduction to language*. Cengage Learning, 11th edition.

Goldwater, S. and McClosky, D. (2005). Improving statistical mt through morphological analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, page 676–683, Vancouver, British Columbia, Canada.

Greenberg, J. H. (1960). A quantitative approach to the morphological typology of language. *International Journal of American Linguistics*, 26(3):178–194.

Habash, N. and Rambow, O. (2005). Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan. Association for Computational Linguistics.

Hajic, J. (2000). Morphological tagging: Data vs. dictionaries. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Hajic, J. and Hladka, B. (1998). Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

Hakkani-Tür, D. Z., Oflazer, K., and Tür, G. (2002). Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36(4):381–410.

Haspelmath, M. (2002). *Understanding Morphology*.

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

Johnson, C. D. (1972). *Formal Aspects of Phonological Description*. Mouton, The Hague. Monographs on Linguistic Analysis No. 3.

Kanerva, J., Ginter, F., and Salakoski, T. (2021). Universal lemmatizer: A sequence-to-sequence model for lemmatizing universal dependencies treebanks. *Natural Language Engineering*, 27(5):545–574.

Kann, K., McCarthy, A. D., Nicolai, G., and Hulden, M. (2020). The SIGMORPHON 2020 shared task on unsupervised morphological paradigm completion. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 51–62, Online. Association for Computational Linguistics.

Kann, K. and Schütze, H. (2016). Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 555–560, Berlin, Germany. Association for Computational Linguistics.

Kaplan, R. M. and Kay, M. (1981). Phonological rules and finite-state transducers. Paper presented at the Annual meeting of the Linguistics Society of America. New York.

Karttunen, L. and Beesley, K. R. (2001). A short history of two-level morphology.

Khandelwal, U., He, H., Qi, P., and Jurafsky, D. (2018). Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, Melbourne, Australia. Association for Computational Linguistics.

Kirov, C., Cotterell, R., Sylak-Glassman, J., Walther, G., Vylomova, E., Xia, P., Faruqui, M., Mielke, S., McCarthy, A., Kübler, S., Yarowsky, D., Eisner, J., and Hulden, M. (2018). UniMorph 2.0: Universal morphology. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Koehn, P. and Hoang, H. (2007). Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Kondratyuk, D. (2019). Cross-lingual lemmatization and morphology tagging with two-stage multilingual BERT fine-tuning. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 12–18, Florence, Italy. Association for Computational Linguistics.

Koskenniemi, K. (1983). Two-Level Model for Morphological Analysis. In *IJCAI*.

Linden, K., Silfverberg, M., and Pirinen, T. (2010). Hfst tools for morphology – an efficient open-source package for construction of morphological analyzers.

Ling, W., Dyer, C., Black, A. W., Trancoso, I., Fermandez, R., Amir, S., Marujo, L., and Luís, T. (2015). Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Makhambetov, O., Makazhanov, A., Sabyrgaliyev, I., and Yessenbayev, Z. (2015). Data-Driven Morphological Analysis and Disambiguation for Kazakh. In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing*, pages 151–163, Cham. Springer International Publishing.

Malaviya, C., Wu, S., and Cotterell, R. (2019). A simple joint model for improved contextual neural lemmatization. *CoRR*, abs/1904.02306.

McCarthy, A. D., Vylomova, E., Wu, S., Malaviya, C., Wolf-Sonkin, L., Nicolai, G., Silfverberg, M., Mielke, S. J., Heinz, J., Cotterell, R., and Hulden, M. (2019). The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244, Florence, Italy. Association for Computational Linguistics.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In Kobayashi, T., Hirose, K., and Nakamura, S., editors, *INTERSPEECH*, pages 1045–1048. ISCA.

Mueller, T., Schmid, H., and Schütze, H. (2013). Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332. Association for Computational Linguistics.

Müller, T., Cotterell, R., Fraser, A., and Schütze, H. (2015). Joint Lemmatization and Morphological Tagging with Lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, Lisbon, Portugal. Association for Computational Linguistics.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Oflazer, K. (1994). Two-level Description of Turkish Morphology. *Literary and Linguistic Computing*, 9(2):137–148.

Oflazer, K. and Guzey, C. (1994). Spelling correction in agglutinative languages. In *Fourth Conference on Applied Natural Language Processing*.

Oflazer, K. and Kuruoz, I. (1994). Tagging and morphological disambiguation of Turkish text. In *Fourth Conference on Applied Natural Language Processing*, pages 144–149, Stuttgart, Germany. Association for Computational Linguistics.

Oh, B.-D., Maneriker, P., and Jiang, N. (2019). THOMAS: The hegemonic OSU morphological analyzer using seq2seq. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 80–86, Florence, Italy. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the*

2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Roche, E. and Schabes, Y. (1997). *Finite-State Language Processing*. Language, Speech, and Communication. The MIT Press.

Sak, H., Güngör, T., and Saraçlar, M. (2007). Morphological disambiguation of Turkish text with Perceptron algorithm. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 107–118. Springer.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Shadikhodjaev, U. and Lee, J. S. (2019). CBNU system for SIGMORPHON 2019 shared task 2: a pipeline model. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 19–24, Florence, Italy. Association for Computational Linguistics.

Sproat, R. (1991). Review of "PC-KIMMO: A Two-Level Processor for Morphological Analysis" by Evan L. Antworth. Summer Institute of Linguistics 1990. *Comput. Linguist.*, 17(2):229–231.

Straka, M., Hajic, J., and Straková, J. (2016). UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In Chair), N. C. C., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Straka, M., Straková, J., and Hajic, J. (2019). UDPipe at SIGMORPHON 2019: Contextualized embeddings, regularization with morphological categories, corpora merging. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 95–103, Florence, Italy. Association for Computational Linguistics.

Straková, J., Straka, M., and Hajič, J. (2014). Open-source tools for morphology, lemmatization, POS tagging and named entity recognition. In *Proceedings of 52nd*

Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 13–18, Baltimore, Maryland. Association for Computational Linguistics.

Sulubacak, U., Eryigit, G., and Pamay, T. (2016). Imst: A revisited turkish dependency treebank.

Toleu, A., Tolegen, G., and Makazhanov, A. (2017). Character-aware neural morphological disambiguation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 666–671, Vancouver, Canada. Association for Computational Linguistics.

Tsarfaty, R., Seddah, D., Goldberg, Y., Kübler, S., Candito, M., Foster, J., Versley, Y., Rehbein, I., and Tounsi, L. (2010). Statistical Parsing of Morphologically Rich Languages (SPMRL): What, How and Whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, SPMRL '10, pages 1–12, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tür, G., Hakkani-Tür, D., and Oflazer, K. (2003). A statistical information extraction system for Turkish. *Natural Language Engineering*, 9(2):181–210.

Tyers, F., Washington, J., Çöltekin, Ç., and Makazhanov, A. (2017). An assessment of Universal Dependency annotation guidelines for Turkic languages.

Üstün, A., van der Goot, R., Bouma, G., and van Noord, G. (2019). Multi-team: A multi-attention, multi-decoder approach to morphological analysis. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 35–49, Florence, Italy. Association for Computational Linguistics.

Vania, C., Grivas, A., and Lopez, A. (2018). What do character-level models learn about morphology? the case of dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2573–2583, Brussels, Belgium. Association for Computational Linguistics.

Vania, C. and Lopez, A. (2017). From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 2016–2027, Vancouver, Canada. Association for Computational Linguistics.

Wiemerslage, A., McCarthy, A. D., Erdmann, A., Nicolai, G., Agirrezabal, M., Silfverberg, M., Hulden, M., and Kann, K. (2021). Findings of the SIGMORPHON 2021 shared task on unsupervised morphological paradigm clustering. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 72–81, Online. Association for Computational Linguistics.

Wolf, T., Lhoest, Q., von Platen, P., Jernite, Y., Drame, M., Plu, J., Chaumond, J., Delangue, C., Ma, C., Thakur, A., Patil, S., Davison, J., Scao, T. L., Sanh, V., Xu, C., Patry, N., McMillan-Major, A., Brandeis, S., Gugger, S., Lagunas, F., Debut, L., Funtowicz, M., Moi, A., Rush, S., Schmidd, P., Cistac, P., Muštar, V., Boudier, J., and Tordjmann, A. (2020). Datasets. *GitHub. Note: https://github.com/huggingface/datasets*, 1.

Yildiz, E. and Tantuğ, A. C. (2019). Morpheus: A neural network for jointly learning contextual lemmatization and morphological tagging. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 25–34, Florence, Italy. Association for Computational Linguistics.

Yildiz, E., Tirkaz, C., Sahin, H. B., Eren, M. T., and Sonmez, O. O. (2016). A morphology-aware network for morphological disambiguation. In *Thirtieth AAAI Conference on Artificial Intelligence*.