UNIVERSITEIT VAN AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Inductive Biases for Morphologically Informed Neural Machine Translation

by

Ivo Verhoeven

13013319

August 3, 2022

48 ECTS
November 2021 - July 2022

Supervisor:
Dr Wilker Aziz

Assessor:
Dr Ekaterina Shutova

Institute for Logic, Language & Computation

This document contains only the first chapter from the full thesis. The full thesis is publicly available at UvA university library.

Contents

Morphological Tagging and Lemmatization in Context

1

This first chapter deals with morphology as a concept, and automated analysis in natural language. The first section provides a whirlwind tour of the field, intending to provide the reader with a working notion of core definitions used throughout the remaining text. Special effort is made in elaborating differences between languages. The second section deals solely with re-implementing contextual joint lemmatizers and morphological taggers. These were deemed necessary for later efforts, but existing solutions proved inadequate. A novel architecture proves competitive with the state-of-the-art, generalizing well to out-of-vocabulary terms. No reference to machine translation or multilingual modelling is made yet, but these mono-lingual systems play a pivotal role in the coming chapters.

1.1 Morphology

Linguistically speaking, the field of morphology is the study of the smallest, most atomic units of language that carry meaning. The units, called morphemes, are present within all words, and largely define their internal structure. More specifically, the field seeks to understand these constituents of a word, their function, and whether their presence is due to grammatical or semantic necessity. Haspelmath and Sims [1] provide two succinct definitions: morphology is either the study of i) systematic covariation in the form and meaning of words, or, ii) the combination of morphemes to yield words. Not unexpectedly then, morphology is considered an important aspect of linguistics:

[1]: Haspelmath et al. (2013), *Understanding morphology*

> Morphology is the conceptual centre of linguistics. This is not because it is the dominant sub-discipline, but because morphology is the study of word structure, and words are at the interface between phonology, syntax and semantics. Spencer and Zwicky [2]

[2]: Spencer et al. (1998), *The Handbook of Morphology*

More intuitive perhaps than the notion of a morpheme is that of a word. For simplicity's sake, a word or token will be defined as some contiguous sequence of characters with some natural boundary pre- and succeeding. As alluded to earlier, a word is in fact already a compound structure. Two distinct variants of a word exist:

- when considering the abstract meaning of a word, one is considering the lexeme. Many words can belong to this lexeme, but are all represented by the same lemma. Lemmas are the items by which a dictionary is indexed, capturing some core concept, condensing a whole set of words into one
- when considering the concrete form of a word or token, one is considering the word-form, surface form or orthographic form of said word. The word-form augments the bare meaning of the lemma with morphemes that carry grammatical function, or alter the word's meaning

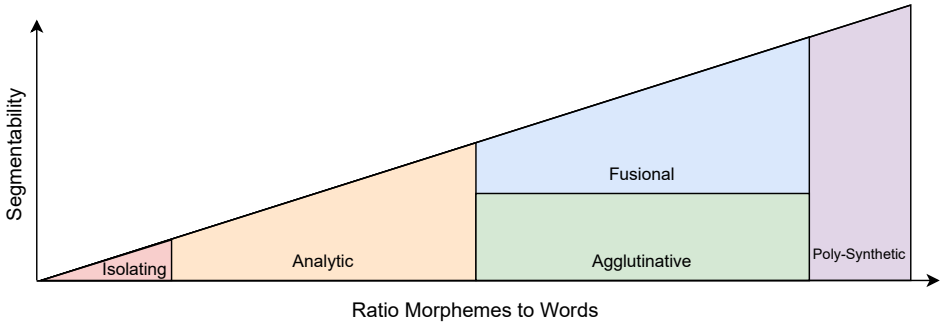

Figure 1.1: The morphological typology landscape and commonly identified subsets within.

In short, we read word-forms, and think in terms of lemmas. Consider, for example, the sentence,

> A snare is for catching rabbits; once you have caught the rabbit, forget about the snare.
> Words are for catching ideas; once you have caught the idea, forget about the words. Zhuang [3]

[3]: Zhuang (300BCE), *Zhuangzi*

In the first line, the words 'catching' and 'caught' are two word-forms referring to the same meaning, one referring to a general act, the other describing the successful completion of that act. Both are concrete instances of the lemma 'catch'. In the second line, 'caught' refers to an entirely different concept (i.e. understanding), and thus a different lexeme, but matches in word-form and maps to the same lemma.

Beyond sage advice, the example illustrates the effect of morphemes. In short, morphemes come in two forms: roots and affixes[1]. The root carries the meaning, the affixes change it to fit in the sentence. If the change is lexically motivated, i.e. to create a new vocabulary item, the word-formation process is one of derivation. If instead it is syntactically motivated, the affix is deemed functional, and it becomes an inflection. For example, to take the lemma 'catch' to 'caught', 'ught' is suffixed to the root 'ca', an inflection required due to a change in tense. All word-forms together, belonging to the same lemma constitute that lemma's paradigm (e.g. 'catch' includes in its paradigm the tense inflected forms 'catch', 'caught', 'caught').

1: Which in turn come in various flavours.

- If preceding the root, it is a prefix
- When succeeding the root, it is a suffix
- In the rare case it is placed within the root, it is an infix
- The opposite of an infix, requiring both a pre- and suffix, is the circumfix

Rules of word-formation, which come naturally to native-speakers, follow (for the most part) strict patterns. Such patterns are typically linked to a word's part-of-speech (PoS). The number and specific instances of those PoS differ somewhat between annotation schemas, but standard entries include nouns, verbs, etc. One important dichotomy between PoS are the open and closed classes, tightly linked to the notion of derivation and inflection, respectively. The former deals primarily with lexical items, and is limited only to our ability to invent new meanings. The latter, in contrast, contains words necessary to make grammatical sentences, with practically no information carried in isolation (for example, conjunctions like 'and' or 'but' or 'or').

1.1.1 Morphological Typology

While understandably important to study for each language separately, differences across languages in their approach to morphology provide a useful framework for classifying languages and specifying their relationship. This subfield is called morphological typology.

Historically, two useful metrics exist for classifying where a language falls in the morphological typology landscape [4]. The first is the degree of synthesis, defined as the number of morphemes per word. Languages with low synthesis are deemed 'analytic', whereas those with relatively high degree, 'synthetic'. For an example clearly showing the difference, compare Haspelmath and Sims [1]'s glossed translations. First an extremely analytic[2] language, Vietnamese, to a moderately analytic language like English,

2: Low synthesis are sometimes categorized as 'isolating'

Vietnamese	Hai	d-ú.a	bo?	nhau	là	ta.i	gia-d-ình	thàng	chông.
Morphemes	two	individual	leave	each other	be	because of	family	guy	husband
English	They divorced because of his family								

Next, an extremely synthetic[3] language like West Greenladic,

West-Greenlandic	Paasi-nngil-luinnar-para	ilaa-juma-sutit.
Morphemes	understand-not-completely-1SG.SBJ.3SG.OBJ.IND	come-want-2SG.PTCP
English	I didn't understand at all that you wanted me to come.	

3: High synthesis outliers are sometimes categorized as 'poly-synthetic'

In essence, the degree of synthesis dictates the allowed complexity of individual words in a sentence. For highly analytical languages, each word is typically composed of a single morpheme, and has a single semantic or syntactical function. In comparison, highly synthetic languages have words consisting of many morphemes, whose combination might allow for many distinct functions. Most major European languages tend to be synthetic, with modern English being one of the few exceptions.

The second metric of relevance is the segmentability of morphemes, called the degree of agglutination. Low levels of agglutination indicate that morphemes combine in often irregular patterns, and is typical of 'fusional' languages. Juxtaposed are the 'agglutinative' languages, having high levels of agglutination, and highly patterned composition. A prototypical example of an agglutinative language is the concatenative morphology of Turkish. Taken from Comrie [4], the following table provides the noun conjugation of the word 'walk' ('adam'), with hyphens added for effect. Note the highly predictable shift from singular to plural forms, across each casing form. Contrast this to a fusional language like Russian. This

Case/Number	**Singular**	**Plural**
Nominative	adam	adam-lar
Accusative	adam-ı	adam-lar-ı
Genitive	adam-ın	adam-lar-ın
Dative	adam-a	adam-lar-a
Locative	adam-da	adam-lar-da
Ablative	adam-dan	adam-lar-dan

table [4] instead provides a similar paradigm for the Russian nouns 'table' ('stol'), and 'lime tree' ('lipa'). Two words are included to show the

effect of a third variable of complexity interacting with the previous two, namely noun declension types. Note how the casing affixes across the singular and plural forms are often conflated, which in turn also vary across the declension type (I or II).

Declension Type	**I**		**II**	
Case/Number	**Singular**	**Plural**	**Singular**	**Plural**
Nominative	stol	stol-y	lip-a	lip-y
Accusative	stol	stol-y	lip-u	lip-y
Genitive	stol-a	stol-ov	lip-y	lip
Dative	stol-u	stol-am	lip-e	lip-am
Locative	stol-om	stol-ami	lip-oj	lip-ami
Ablative	stol-e	stol-ax	lip-e	lip-ax

Both degrees of synthesis and agglutinivity correlate positively with the notion of morphological complexity. However, the relationship is conceptually not unbounded; the distinction between fusional and agglutinative languages typically only makes sense between synthetic languages. For an overview of the morphological landscape, see Figure 1.1. When talking about morphologically rich languages, one typically refers the demarcation between synthetic and analytic languages, particularly towards poly-synthetic. Again, relative to English, most European languages tend to more complex in their morphology.

1.1.2 Morphological Annotation

Before discussing modelling approaches to morphology, a significant initial hurdle to overcome is developing a labelling scheme that is consistent across languages. At this point, the notion that languages vary drastically according to their morphological complexity should be clear. The dimensions along which that complexity is expressed are morphological features. In this too, there exists a great deal of variation.

To this end, SIGMORPHON[4] has developed the Universal Morphological Feature (UniMorph) schema [5]. Focused entirely on predicting inflected word-forms, the ultimate goal of the project is multilingual lookup of any word-form from the combination of lemma and features. Otherwise, with the lexical item known, one need only choose a single item from its paradigm to construct a grammatically correct token. This work primarily uses data annotated with UniMorph 2 [6]. Recently, the fourth version released, expanding the 23 dimensions to cover 122 million inflections across 182 languages [7]. The full schema, to which many references will be made throughout, made be found in Sylak-Glassman [5].

4: Special Interest Group on Computational Morphology and Phonology

[5]: Sylak-Glassman (2016), 'The composition and use of the universal morphological feature schema (unimorph schema)'

[5]: Sylak-Glassman (2016), 'The composition and use of the universal morphological feature schema (unimorph schema)'

1.2 Automated Morphological Tagging & Lemmatization in Context

In order to produce fluent text, one needs to choose the right words and as seen earlier, choose the right forms of those words. To do so, information from a variety of different sources needs to be combined. Word choice is not merely semantically motivated, but must be altered to conform to syntax and meaning already present within a sentence. Context is thus paramount.

From a modelling perspective, one important implication of word-formation processes are the large output vocabulary sizes. All possible lemmas are able to take on large paradigms. For morphologically rich languages, this phenomenon is especially endemic, with infinitely productive inflection systems. In turn, this leads to specific word forms or even entire morphological rules not being present in training text of morphologically rich languages, despite size. Generalization to out-of-vocabulary terms is thus crucial.

Strong automated morphological analysis systems must thus be able to quickly infer, from context, which features are present for any word, be able to separate those features in word-form space from the underlying lexical item, and ideally do so from limited data, all the while retaining the ability to be infinitely productive.

Running from December 2018 till August 2019, this is precisely the main focus of the second 2019 CoNLL-SIGMORPHON Shared Task McCarthy et al. [8]. Where previous renditions and other tasks focus on automated parsing of paradigms under a variety of constraints, this was the first (and since only) shared task aimed at incorporating contextual information for full sentences. Specifically, for all tokens present in a string, models are expected to produce i) its lemma, ii) the part-of-speech and iii) the morphological features. Due to the task's setup, large neural systems lend themselves especially well.

While unique to SIGMORPHON's workshops and shared tasks, the requirements are similar to the shared tasks presented by SIGNLL[5]'s CoNLL[6] conference. Specifically, the labelled corpora necessary for supervised learning are present in the Universal Dependencies (UD) project. At the time (version 2.3), the project served as a repository of 129 pre-tokenized treebanks across 79 languages. In turn, the languages are spread across a wide array of typological families, with a wide variety of lower resource languages being included. All languages share an annotation scheme, in theory, and are marked for quality to distinguish the annotation's reliability. Furthermore, all treebanks are provided in official train/dev/test splits, enabling robust comparison across systems. Using an automated modification process, the SIGMORPHON shared task requires predicting 2 out of 11 provided labels for all tokens in the treebanks, and are evaluated on the test-set performance averaged over all provided treebanks.

Table 1.1: An annotated sentence from the Dutch LassySmall treebank, showing the tokenized text and the labels to predict.

Token	Lemma	Feats.
In	in	ADP
1425	1425	NUM
ging	gaan	V;SG;PST;FIN
hij	hij	3;PRO;NOM
naar	naar	ADP
Rijsel	Rijsel	SG;PROPN;NEUT
,	,	PUNCT
waar	waar	ADV
hij	hij	3;PRO;NOM
hof-	-	-
schilder	schilder	N;SG;MASC+FEM
⋮	⋮	⋮

[8]: McCarthy et al. (2019), 'The SIGMORPHON 2019 Shared Task: Morphological Analysis in Context and Cross-Lingual Transfer for Inflection'

5: Special Interest Group on Natural Language Learning

6: Conference on Computational Natural Language Learning

1.2.1 Lemmatization as Classification

A clear division in the submitted systems comes from viewing the lemmatization task as either character-based seq2seq generation or token-level classification. With all systems relying on either a recurrent or self-attention based architecture, moving towards an encoder-decoder setup is a natural extension. By limiting the decoder's output vocabulary to a language's alphabet, the system is infinitely productive with few necessary parameters in the final classification head, but suffers in computational complexity with a greatly increased number of units to classify. A common alternative that retains the benefits of an encoder-decoder setup is predicting an edit operation, or set of edit operations, instead of individual characters. Consecutive edit operations can be concatenated together, requiring an additional label but yielding fewer total number of classifications.

Ultimately, whether characters or character edits, seq2seq generations of lemmas from word-forms requires successful transfer of word-based context to a character-based decoder. The potentially long character sequences and reduced representational power of the decoder can lead to a bottleneck, or at least severely complicates training and prolongs convergence. A far simpler, but more restrictive, method is predicting entire edit-scripts (a concatenation of all character-based edit actions for the entire token). Perhaps an engineering necessity, this recasting of lemmatization as classification proves an effective simplification, being utilized by both winning systems[7]

First proposed as an automated pre-processing step by Chrupala [9], lemmatization as multi-class classification requires finding for tokens in the train set a minimal or shortest edit script between the lemma and word-form. This script consists of a number of specific operations, yielding a deterministic mapping from a token's word-form to its lemma. Actions, defined at the character level, are typically restricted to skipping ("*"), deletion ("-") or insertion ("+?"), with the last action requiring a specific form for all possible insertion symbols. From the two sequences, and specifically these allowed operations, the classic Myers difference algorithm may be applied [10], finding the minimal edit script by searching the edit graph for the shortest path solution, finding application in (for example) Git's diff function [11].

Going from a shortest edit script between two strings to a lemma edit script between a word-form and lemma representation of the same string, requires a few additional steps. First, the longest common sub-string is found, is deemed the stem, and removed from further consideration. All text preceding the stem is considered a prefix, and all text succeeding is a suffix. For both affixes, the case insensitive shortest edit script is found with the operations detailed above. Finally, for the entire string, the occurrences of capitalized characters are noted, with sequences being compressed to their first character. The concatenation of all three components (casing, prefixes, affixes) is the lemma edit script. As a pre-processing step, this is performed for all tokens present in the dataset, enumerated and converted to a one-hot encoded multiclass label vector. One important special case is when no common sub-string is found. They are considered irregular, and given an edit script that ignores the provided token altogether.

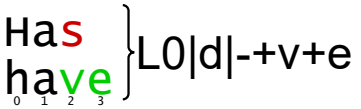

Figure 1.2: An example of a lemma edit script.

7: Compared to systems predicting edit actions, the nearest generative system (CBNU) pre-trains a tiny-transformer exclusively for lemmatization. This yielded a lemmatizer only marginally better than the shared task's baseline, and a morphological tagger significantly worse than the winning systems.

[9]: Chrupala (2006), 'Simple data-driven context-sensitive lemmatization'

[10]: Myers (1986), 'An O(ND) difference algorithm and its variations'

[11]: Coglan (2020), *Building Git*

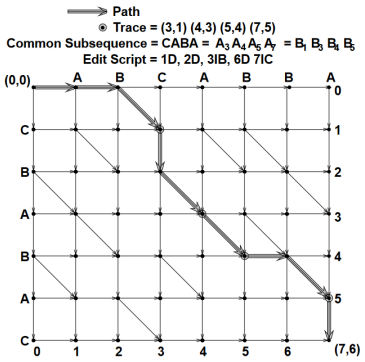

Figure 1.3: The shortest edit script is the shortest path across the edit graph defined on the tokens of both sequences. Taken from [10].

Rule	Count	Examples
L0 | d | d	450024	i→i, the→the, like→like
L0 | d | —	35460	flights→flight, arrives→arrive, later→late
U0,L1 | d | d	27682	President→President, Tuesday→Tuesday
L0 | ign_be	11628	am→be, is→be, ‘m→be
L0 | d | ——	10252	sixth→six, does→do
U0 | d | d	8944	i→I, I→I, AP→AP
L0 | d | ———	6917	returning→return, cheapest→cheap
L0 | ——+b | d	3321	Are→be, ‘re→be, are→be
L0 | d | ———+e	3295	making→make, leaving→leave
L0 | d | —+v+e	2841	has→have, had→have, HAS→have
U0 | ign_I	2745	me→I, Me→I, my→I
L0 | d | ———+y	2039	cities→city, earlier→early, carries→carry
L0 | d | ——+e	1932	ninth→nine, his→he, him→he
L0 | d | —+o*	1646	grew→grow, knew→know, n’t→not
L0 | d | ——+y	1433	paid→pay, said→say, their→they

Table 1.2: The 15 most common lemma edit scripts for English treebanks. The first column gives the script, the second the frequency of the script and the final column some examples from the corpus, as token → lemma.

Overall, while not productive, using lemma edit scripts as labels strikes a nice balance between compressing the lemma space and generalizing to unseen word-forms. Furthermore, the methodology is language agnostic, depending only on the tokenizer. The number of tokens corresponding to a lemma edit script follows a power law, with the vast majority being captured using a small portion of the most common scripts. The most frequent edit scripts tend to be those where the token is already close to the lemma, requiring few to no actions. Consequently, the least frequent edit scripts tend to be for longer words, or ones containing rare character combinations in the affixes.

1.2.2 Architectures

Presented in this subsection are the setups of the 2 winning systems, and a novel architecture leveraging a more recent character-based transformer. All architectures classify a lemma script and a morphological tagset, jointly, for all tokens in a sentence.

UFAL-Prague’s `UDPipe2`

As the name suggests, `UDPipe2` [12] is the second iteration of a recurrent multitask NLP pipeline. In its standard form, it is trained to simultaneously part-of-speech tag, lemmatize and parse dependency relationships between tokens. This setup proved state-of-the-art for a previous CoNLL shared task [13], and needed few modifications for the SIGMORPHON/-CoNLL 2019 shared tasks. At it core, it uses a standard NLP setup. Words are fed through a variety of embeddings, which are passed on to a deep bidirectional recurrent block, before being classified with task-specific multilayer perceptrons (MLPs).

The word-level embeddings consist of three forms: pre-trained language-specific subword aware `fastText` embeddings [14, 15]; contextualized embeddings from feeding the entire sentence into `BERT` [16] and recovering tokens from the averaged BPE representation of the final four layers; and finally, trainable token-specific word embeddings. A fourth type of

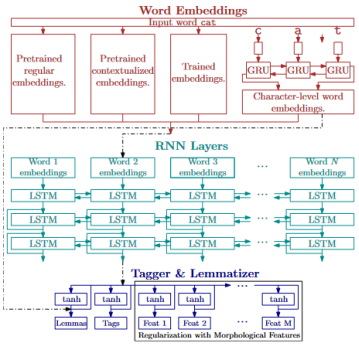

Figure 1.4: The `UDPipe2` architecture presented graphically. Taken from [12].

[12]: Straka et al. (2019), ‘UDPipe at SIGMORPHON 2019: Contextualized embeddings, regularization with morphological categories, corpora merging’

[13]: Straka (2018), ‘UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task’

'word embeddings' are provided in the form of a trainable `char2word` module [17]; tokens are fed as characters into a bidirectional `GRU` [18], with directions being concatenated and projected down before sum-pooling over the time dimension. In turn, the recurrent block consists of a 3 layer bidirectional `LSTM` [19], with skip-connections between the layers [20]. The final block consists of 2 layer MLPs, again with skip-connections, each classifying the produced logits for a separate task. The char2word embeddings are appended to the contextualized token representations prior to lemmatization, adding another skip-connection. As a method of regularization, the morphological tags are classified jointly, but also factored into categories, with the latter loss only being used during training. Regularization proved crucial for generalizability, with dropout being applied throughout and label smoothing when classifying.

`UDPipe2` relies heavily on pre-trained neural modules, at a variety of context levels. These are all simply concatenated before re-contextualizing. As a result, this architecture has proven useful to a number of NLP tasks, with more relevant embeddings being quickly slotted in, as evidenced by Straka and Straková [21], where swapping out `BERT` for `RoBERTa` [22] provided a modest performance bump. Thus, despite having many parameters, relatively few are trainable for the task at hand, while most were previously exposed to large amounts of data with more general objectives. This makes re-training comparatively efficient, and quickly allows for scaling to larger datasets, at the cost of an increased memory footprint and inference latency.

[21]: Straka et al. (2020), 'UDPipe at EvaLatin 2020: Contextualized embeddings and treebank embeddings'

Charles-Saarland's `UDIFY`

Kondratyuk presents `UDIFY` [23, 24] as a system very similar to `UDPipe2`. Instead of relying on 4 somewhat similar word-embedders, only `BERT` and char2word are kept. Furthermore, `BERT` is made trainable, with token embeddings being created by attending over all self-attention layers, keeping only the first BPE. Where `UDPipe2` uses a tightly joined recurrent block, `UDIFY` separates these into two smaller `LSTM`s: one for morphological tagging and one for lemmatization. Otherwise, differences are minor at best.

Previous research into transformer-based architectures have already indicated the highly hierarchical representations built by `BERT` and the like; lower layers tend to specialize in classically upstream tasks, whereas upper layer representations contain more upstream, context-dependent information [25]. Layer attention, a simple static vector of linear mixing coefficients, as utilized by `UDIFY` could lead to automatically detecting and leveraging these specializations. This effect is then compounded by passing gradients back to the decoder-only block. While consisting of relatively many trainable parameters, and sacrificing the latency gains of self-attention modules by introducing recurrent blocks throughout, `UDIFY` remains fully language agnostic, with the individual components all being shown to scale to multilingual settings. Kondratyuk leverages this property well, with optimal performance only being achieved when pre-training on all available languages, before fine-tuning to each treebank in isolation.

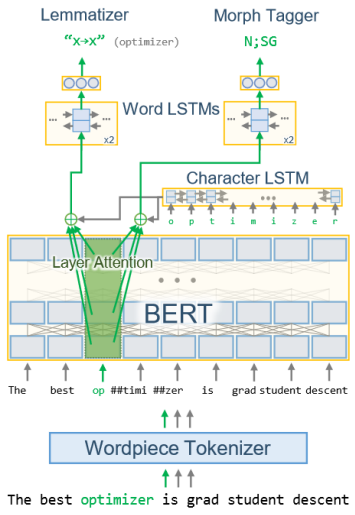

Figure 1.5: The `UDIFY` architecture presented graphically. Taken from [23].

[23]: Kondratyuk (2019), 'Cross-lingual lemmatization and morphology tagging with two-stage multilingual BERT fine-tuning'

[24]: Kondratyuk et al. (2019), '75 languages, 1 model: Parsing universal dependencies universally'

[25]: Tenney et al. (2019), 'BERT rediscovers the classical NLP pipeline'

DogTag

In the years since, transformer architectures have gone from wildly impressive newcomers to virtually ubiquitous, with improvements being made along the way. While the hidden states contain some useful representations for lemmatization and annotation, they remain largely unaware of morphological features in word-form space, due to commonly used sub-word tokenizers. One specific architecture that proves an exception to this rule is Clark et al.'s `CANINE`[26]. Using a series of convolutions for down- and up-sampling, they manage to train a deep transformer stack on character input and output. Much like `BERT`, the first token of the transformer stack contains a semantic representation of the entire sentence, but only the upsampled character representations are used for the masked language modelling objective. Via a relatively simple added module, they produce a model that is entirely tokenizer free, vocabulary free, almost inherently multilingual and capable of handling long sequences - all with 30% fewer parameters than sub-word alternatives[8]. However, while provably impressive at natural language understanding tasks, due to the character based MLM objective, the final layer is by necessity aware of morphological word-formation processes. Hence, the final layer representations likely contain information pertinent to tasks like lemmatization and morphological feature prediction.

[26]: Clark et al. (2022), 'Canine: Pre-training an efficient tokenization-free encoder for language representation'

`DogTag` follows this line of reasoning: a joint lemmatization and morphological feature prediction model in the style of `UDPipe2` or `UDIFY`, but with `CANINE` as the sole feature extractor. The contextualized character representations produced by `CANINE` are collated using multi-head attention [27] between the character sequence and a learnable query matrix (of length 1). Beyond the multiple heads (whose number correlated positively with performance), three separate character collators are trained: two for directly feeding into lemmatization and morphological feature prediction, and one being fed into a bidirectional, multi-layer residual `LSTM` for recontextualization of the token representations. These representations are then concatenated with the task specific collations. A simple `MLP` is trained to classify tokens, with training only regularization coming from a factored prediction. While initial experiments keep `CANINE` fixed, fine-tuning the feature extractor proved beneficial, although unstable at times.

8: However, given the auto-regressive modelling of characters, the authors do show a 50% increased sentence throughput.

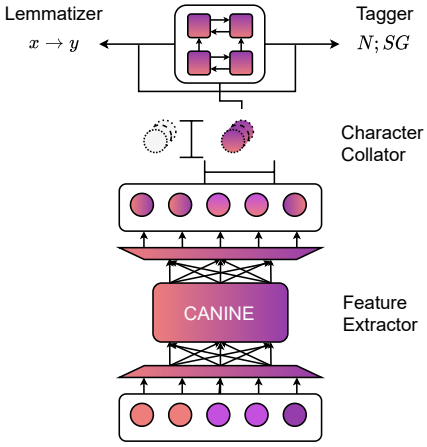

Figure 1.6: The `DogTag` architecture presented graphically.

1.2.3 Methods

Data

Since the 2019 CoNLL-SIGMORPHON Shared Task, the UD treebanks have seen numerous revisions and updates, both in languages used and novel. Specifically, the used version stems from 2.3, with version 2.9 and 2.10 being made available in November 2021 and May 2022 respectively. One particularly relevant change are differing train/test splits, making direct comparisons practically impossible. For many treebanks, the annotations have been brought to align more closely to the UD standard.

To leverage the improved data quality, treebanks from version 2.9 were used. The `UPOS` and `XPOS`, carrying the Part-of-Speech (PoS) and morphological feature tag sets respectively, were converted to Universal

Table 1.3: Differences between UD and UM annotations. Taken from [28].

Schema	Annotation
UD	VERB MOOD=IND | NUMBER=SING | PERSON=3 | TENSE=IMP | VERBFORM=FIN
UniMorph	V;IND;PST;3;SG;IPFV

Morphology (UM) [6] tagsets using the same methodology as presented by McCarthy et al. [8]. The work by McCarthy et al. [28] presents an automated conversion system that merges PoS with morphological features, with language specific combinations being considered, leading to increased cross-lingual agreement. They further show improved tagging recall scores compared to simply using the UD features. Lemmas were left intact, despite systematic differences across tree-banks, and all other provide features were left in CoNLL-U format. For an extended overview and explanation of all UniMorph morphological tags, see Sylak-Glassman [5].

[28]: McCarthy et al. (2018), 'Marrying Universal Dependencies and Universal Morphology'

[5]: Sylak-Glassman (2016), 'The composition and use of the universal morphological feature schema (unimorph schema)'

Unlike the shared task, this re-implementation focuses on language-level training, concatenating available treebanks. One unavoidable source of systematic errors are differences in annotation methodologies across different treebanks. While the UD project has prescribed a standard, ultimately their primary function is collating available treebanks, leaving manual annotation to separate authors. This leaves mixtures of treebanks, and in turn entire languages, riddled with noisy labels. Some treebanks especially do not lend themselves to the task of automated morphological tagging and lemmatization due to incomplete or unconventional schemas. Therefore, UD provides quality ratings on their website, loosely based on the unattached undirected attachment score (UUAS) of the provided treebanks, with scores close to 0 indicating particularly poor treebanks. To avoid inclusion of harmful treebanks, a lower quality limit of 0.2 (1 out of 5 stars) is adhered to. For an overview of available language corpora, their size and provenance, see Appendix A Table A.1.

Experimenting with all available languages is prohibitively expensive. Instead, 8 (primarily Indo-European) languages were chosen for a good mixture of typological families, morphologically complexity. See Table 1.4.

Table 1.4: Chosen languages. Columns give their typological families, the language name and their position on the morphological spectrum (Analytic (Ana.), Synthetic (Syn.), Fusional (Fus.) and Agglutinative (Agg.).

Fam.	Lang.	Type
Germanic	Dutch	Syn.
	English	Ana.
Romance	French	Fus.
Slavic	Czech	Fus.
	Russian	Fus.
Semitic	Arabic	Fus.
Turkic	Turkish	Agg.
Uralic	Finnish	Agg.

Implementation Differences & Hyperparameters

Given new datasets and a slightly altered end-goal, namely strong language-specific morphological taggers for a downstream task, the use of intention behind using pre-defined architectures is not replication of earlier results, but merely re-implementation of strong baselines. Furthermore, the cited systems were each developed using NLP-specific deep learning frameworks, whereas their re-implemented versions are built using PyTorch [29] only. Some implementation details differ:

- **Factoring**: both `UDPipe2` and `UDIFY` use factored (i.e. the morphological tags split into separate categories) tag sets as a training-only form of regularization. With the new datasets, this proved difficult to re-implement, as some tags were not linked to any category, and some to multiple. Instead, models were trained to classify present morphological categories along side all tags.
- **Sparse Embeddings**: support for sparse embeddings remains lacking in PyTorch, and lead to instabilities. Instead, for embedding layers, back-propagated gradients were dense with a high β_2 value (0.999) for the Adam optimizer [30] instead.
- **Additional Regularization**: the most prevalent issue in test-set performance appears to be overfit to the training dataset. While

both cited architectures use high levels of dropout throughout, additional regularization techniques proved beneficial. Of note are masking of entire words and characters when feeding input sequences to pre-trained embedders

- **Competition tricks**: rather than focusing on building systems capable of winning on a per-treebank basis, the systems are designed for performance on a single language. This invalidates some training strategies employed by the original authors. For example, no additional per-treebank fine tuning is done, nor is ensembling of systems, nor language specific hyperparameter searches.

Despite these alterations and augmentations, the hyperparameter sets used were taken directly from the original papers. Overall, the systems proved reasonably robust to most choices, and no language-specific changes were introduced. All results and corresponding hyperparameter choices may also be found in the Weights & Biases [31] dashboard used for experiment tracking[9]. Used code, datasets and model checkpoints has been documented and open-sourced [10], and should allow for easy replication.

9: https://wandb.ai/verhivo/morph_-tag_lemmatize

10: https://github.com/IvoOVerhoeven/morph_-tag_lemmatize

1.2.4 Results

Overall Test Set Performance

In the style of the shared task, overall test set performance metrics are provided in Table 1.5. The values are computed using all available sentences in the official test splits of the used datasets. Again, direct comparison is not possible, but these do provide comparisons between systems, and ballpark estimates of how these systems compare to their original counterparts. For `UDPipe2` and `UDIFY` the self-reported metrics are provided, averaged over the languages included here. System wide means are also provided, averaged over all used languages[11].

11: This practise has come under scrutiny, with high-resource language families unfairly inflating average scores. Averaging over typological family averages can change the systems' rankings [32].

Both `UDPipe2` and `UDIFY` are clearly strong baselines, extending their impressive competition results to updated datasets and a slightly altered training procedure. Give `UDPipe2`'s heavy reliance on pre-trained resources to provide morphologically or context aware word embeddings, the model requires only a fraction of the training time used by its rivals, making its performance all the more striking. In the officially published results, it already achieved the best lemmatization performance, and was the second best tagger, with notable improvement from corpora merging. Here, it's primarily the morphological tagging performance that stands out. Overall, however, `UDIFY` appears to be the winner. It sets the system-wide highest metrics for most languages, and does so with a respectable margin. Strangely enough, on some of the higher resource languages its performance falters relative to the rest. As such, its across language performance is on par with `UDPipe2`.

Comparatively, `DogTag` uses only a fraction of the parameters, and in the `DogTag`-Fixed variant, trains only a fraction of that fraction. Allowing the fine-tuning of the `CANINE` backbone proves important, bringing the system to close to SoTA. Even without the fine-tuning, `DogTag` proves a strong lemmatizer especially. Overall, it seems the best lemmatizer, and

Table 1.5: Test set performance of `UDPipe2` and `DogTag` (with pre-trained `CANINE` weights, and mono-lingual finetuned variants). Metrics are provided per-language, and mean aggregated per system in the MEAN row. All standard deviations were below 5e-3, and are omitted for brevity's sake. Numeric columns provide, in order, the 0-1 accuracy of a tokens predicted lemma, the Levenshtein distance between predicted and ground-truth lemma, the 0-1 set accuracy of tokens predicted morphological feature set, the F1 score for morphological tags (presented as micro/macro averaged), and finally the throughput in tokens per second, as measured on a NVIDIA GTX 1080Ti GPU, with batches of 2048 tokens and at most 248 sentences. Bold values indicate best across systems. Arrows ↑↓ indicate whether higher or lower values are desired, respectively. For `UDPipe` and `UDIFY` the self-reported competition results are presented as a *rough* baseline. For `UDIFY`, the multilingual with mono-lingual fine-tuning models are used as baselines. Given the differences in training, and the different datasets, direct comparison is not recommended. Instead, these present an upper limit to the performance of the replicated models.

Model	Language	Lemma Acc. ↑	Lev. Dist. ↓	Morph. Set Acc. ↑	Morph. Tag F1 ↑	Throughput ↓
UDPipe2	Arabic	0.93	0.21	0.90	0.96/0.85	**2313**
	Czech	0.98	0.03	0.92	0.98/0.90	**2930**
	Dutch	0.94	0.12	0.95	0.97/0.93	**3223**
	English	**0.97**	**0.05**	**0.92**	**0.96/0.90**	**2977**
	Finnish	0.82	0.44	0.81	0.92/0.62	**2633**
	French	**0.98**	**0.04**	**0.92**	**0.97**/0.87	**3716**
	Russian	**0.97**	**0.06**	**0.92**	**0.97/0.88**	**2759**
	Turkish	0.91	0.19	0.77	0.89/0.58	**1828**
	MEAN	**0.94**	0.14	**0.89**	**0.95**/0.82	**2797**
	[12]	0.96	0.11	0.95	0.98/	–
UDIFY Mono	Arabic	**0.94**	**0.18**	**0.93**	**0.96/0.88**	2135
	Czech	**0.99**	**0.02**	**0.95**	**0.98/0.95**	2413
	Dutch	**0.95**	**0.09**	**0.96**	**0.97/0.96**	2507
	English	0.93	0.12	0.82	0.89/0.82	2445
	Finnish	**0.88**	**0.26**	**0.92**	**0.96/0.84**	2106
	French	0.94	0.18	0.93	0.96/0.88	2135
	Russian	0.92	0.14	0.77	0.91/0.80	2242
	Turkish	**0.94**	**0.13**	**0.83**	**0.92/0.74**	1371
	MEAN	**0.94**	0.14	**0.89**	0.94/**0.86**	2169
	[23]	0.95	0.11	0.95	0.98/	–
DogTag Fixed	Arabic	0.85	0.45	0.76	0.90/0.77	1851
	Czech	0.93	0.12	0.72	0.90/0.81	2255
	Dutch	0.87	0.27	0.79	0.87/0.83	2255
	English	0.93	0.12	0.79	0.88/0.81	2361
	Finnish	0.67	0.85	0.55	0.77/0.52	1620
	French	0.95	0.09	0.84	0.93/0.81	1652
	Russian	0.91	0.16	0.73	0.89/0.78	2110
	Turkish	0.79	0.46	0.60	0.81/0.53	1714
	MEAN	0.86	0.32	0.72	0.87/0.73	1977
DogTag Mono	Arabic	0.93	0.20	0.87	0.94/0.84	1851
	Czech	0.98	0.03	0.90	0.97/0.93	2279
	Dutch	0.93	0.13	0.92	0.95/0.94	2256
	English	**0.97**	0.06	0.89	0.94/0.87	2282
	Finnish	0.85	0.32	0.85	0.94/0.81	1915
	French	**0.98**	**0.04**	**0.92**	**0.97/0.88**	2623
	Russian	**0.97**	**0.06**	0.88	0.96/**0.90**	2135
	Turkish	0.92	0.18	0.78	0.91/0.72	1714
	MEAN	**0.94**	**0.13**	0.88	**0.95/0.86**	2132

while it lags on morph tag set accuracy, the F1 scores stand out. This might indicate slightly better capacity at handling rare features.

There exist notable differences within languages, but across systems. The three smallest corpora, Arabic, Finnish and Turkish[12], yield lower lemmatization scores, and substantially lower morphological tagging results. For Finnish and Turkish especially, the macro averaged F1 scores lag behind (behind the '/'). Again, low values on especially this metric are symptomatic of not being able to handle infrequent label instances.

12: Interestingly, these also represent the included agglutinative languages.

More disappointing was the lack of improvement from the 2 stage multi-lingual pre-training followed by mono-lingual fine-tuning setup, as prescribed by `UDIFY`. Results may be found in Appendix A Table A.2. The `UDIFY` shows definite improvement, approaching the self-reported global averages. Relative to their monolingual training, however, the improvement is smaller. It could be that treebank merging already provides much of the benefit that including other languages from the same typological family provides. Otherwise, the system might be close to a feasible upper limit on performance. For `DogTag`, hardly any improvement was booked overall, and the Turkish system even showed decline. Potential explanations might include a lack of consistency in the language merged labels, or requiring merging of the dataset beyond the typological family.

Generalization to Out-of-vocabulary Terms

While the use of hidden test set does emulate the system's performance on a natural language corpus of the same language, it does not directly test the generalizability of the systems to new word-forms and new lemmas. Especially for morphologically complex languages, this ability is a necessity for downstream use. Thus, moving beyond the analysis provided by the task organizers, the generalizability of the `UDIFY` and `DogTag` are put to the test. Two forms of generalization are tested for, i) comparing seen word-forms to unseen word-forms of known lemmas, and ii) comparing seen lemmas to unseen ones. Both test to which degree the models can leverage provided information beyond their capacity to memorize word-forms. Such a test ideally checks both the lemmatization and morphological tagging capacity. The chosen metrics, one for each task, are the Levenshtein distance and the intersection over union (IoU) of the produced tag set. The latter is a more fine-grained measure of token-level accuracy than presented in Table 1.5.

Table 1.6 displays the results per-language, averaged. The mean generalization gap between seen and unseen is provided in terms of the within group variance via Cohen's d[13], with ideal values being 0 (no difference between seen and unseen). To test the differences systems, p is provided, indicating the one-sided probability that the d value for `UDIFY` and `DogTag` are statistically significant[14].

The Levenshtein distance between predicted and ground-truth lemmas shows a counter-intuitive pattern when comparing generalization to new word-forms and to new lemmas. In the former situation, only the particular inflection is unfamiliar. One would expect a successful lemmatizer to be able to disentangle the affix from a lemma, with affixes generally being shared across paradigms. However, when comparing

13: Rather than reporting a standard statistic for difference of means, like Student's or Welch's t-test, Cohen's d [33] is used; the difference in means in units of the pooled group standard deviation. Unlike the aforementioned statistics, Cohen's d provides sensible values when variances between groups differ and populations are of unequal sizes. This is the case for comparing seen word-forms/lemmas to unseen ones. The former is far more frequent than the latter. Cohen's d has as sampling distribution a standard normal distribution, and the variance is approximately known [34].

14: While hypothesis testing is hinted at, please note that p here is not used as a formal hypothesis test. Instead, it merely provides an indication that the observed effect is actually present. In other words, it indicates the probability that the same test repeated with a new set split would yield $d_{\texttt{DogTag}} - d_{\texttt{UDIFY}} = 0$

Table 1.6: Testing the generalization capacity of the best systems from Table 1.5. Given are, per-language and per-system, the average metric value for tokens that fall in the seen and unseen categories. Arrows ↑↓ indicate whether higher or lower values are desired, respectively, and a ◇ indicates 0 is ideal. The difference between these is given in units of the pooled standard deviation, a statistic known as Cohen's d. The p value provides the probability that the difference $d_{\texttt{DogTag}} - d_{\texttt{UDIFY}}$ being positive is an artefact of the variance inherent to each values, with * indicating $p < 5e-2$, a standard hypothesis testing acceptance rate.

		Lev. Distance ↓							**Morph. IoU ↑**						
		UDIFY			DogTag			p	UDIFY			DogTag			p
	Lang.	**Seen**	**Unseen**	$d\diamond$	**Seen**	**Unseen**	$d\diamond$		**Seen**	**Unseen**	$d\diamond$	**Seen**	**Unseen**	$d\diamond$	
Word-form	Arabic	0.10	1.51	−2.01	0.11	1.74	−2.17	1.000	0.97	0.90	0.44	0.95	0.81	0.70	1.000
	Czech	0.01	0.13	−0.56	0.02	0.16	−0.54	0.099	0.98	0.94	0.43	0.96	0.91	0.41	0.090
	Dutch	0.04	0.58	−1.37	0.06	0.65	−1.18	0.001*	0.98	0.90	0.55	0.96	0.79	0.80	1.000
	English	0.09	0.99	−1.81	0.05	0.51	−1.13	0.000*	0.88	0.50	1.27	0.93	0.76	0.71	0.000*
	Finnish	0.06	0.69	−0.91	0.09	0.77	−0.85	0.021*	0.97	0.89	0.43	0.94	0.84	0.43	0.466
	French	0.03	0.29	−0.84	0.04	0.35	−0.92	0.914	0.97	0.92	0.38	0.96	0.88	0.47	0.946
	Russian	0.10	0.61	−0.85	0.05	0.29	−0.56	0.000*	0.90	0.77	0.54	0.95	0.91	0.22	0.000*
	Turkish	0.09	0.39	−0.51	0.13	0.56	−0.61	1.000	0.91	0.82	0.32	0.88	0.79	0.31	0.299
	Mean	0.06	0.65	−1.11	0.07	0.63	−1.00	-	0.95	0.83	0.55	0.94	0.84	0.50	-
Lemma	Arabic	0.14	1.23	−1.34	0.17	1.34	−1.37	0.686	0.96	0.75	1.23	0.94	0.70	1.15	0.050*
	Czech	0.01	0.15	−0.61	0.03	0.17	−0.53	0.000*	0.98	0.90	0.79	0.96	0.85	0.76	0.099
	Dutch	0.05	0.57	−0.98	0.08	0.64	−0.89	0.007*	0.98	0.87	0.64	0.95	0.80	0.64	0.562
	English	0.10	0.55	−0.79	0.06	0.23	−0.39	0.000*	0.87	0.62	0.81	0.93	0.80	0.50	0.000*
	Finnish	0.15	1.11	−1.10	0.19	1.08	−0.95	0.000*	0.96	0.91	0.28	0.93	0.86	0.27	0.381
	French	0.03	0.16	−0.40	0.04	0.16	−0.35	0.140	0.97	0.84	0.83	0.96	0.82	0.76	0.073
	Russian	0.13	0.50	−0.60	0.06	0.28	−0.51	0.000*	0.89	0.69	0.81	0.94	0.81	0.72	0.000*
	Turkish	0.11	0.92	−1.30	0.16	1.02	−1.18	0.001*	0.90	0.75	0.56	0.87	0.71	0.55	0.400
	Mean	0.09	0.65	−0.89	0.10	0.62	−0.77	-	0.94	0.79	0.74	0.94	0.79	0.67	-

the mean unseen distances, it becomes readily apparent that word-form generalization yields worse lemmatization performance. Two possible explanations for this effect are:

1. unseen lemmas are easier to inflect or belong to classes that are. For example, no training set could contain all possible proper nouns, but these tend to inflect in a rigid pattern. Phrased otherwise, new lexical items are likely to come from the open class of words, indicating words carrying predominantly semantic information, and tend to be inflected according to learned patterns
2. this is an artefact of limiting the lemma generation to existing lemma scripts. During training, the model is encouraged to associate a word-form to a set of classes, and neglect all scripts not immediately relevant. When presented with a new word-form, the system only chooses a lemma edit script associated with similar words in the training data

The only two languages where is not the case, Finnish and Turkish, are agglutinative. It could be that the prototypical easy segmentability present in this type of languages is at play, although currently evidence is too weak to draw any concrete conclusions. When considering the morphological tagging performance metrics, this relationship is not present, with word-form generalization scoring higher for some languages, and lower for others.

Regardless, DogTag's lemmatization proves to be more robust to OOV terms, both in the sense of surface-forms of words it has already seen,

and altogether new lexical items. While its performance lags for most languages, the difference in performance does not, implying that a general increase to DogTag's lemmatization capacity should yield better performance on unseen word-forms and lemmas also. This effect also extends to morphological tagging, although to a lesser degree. It is already clear that UDIFY is the better morphological tagger overall (see Table. 1.5)), and this extends to unseen word-forms and lemmas also. When comparing generalizbility, DogTag appears to be marginally better for both forms of generalization. While some differences exist between languages, these match the performance differences already noted earlier.

Overall, while possessing far fewer parameters, DogTag shows it is capable of leveraging character-level information in order to generalize as well as or better than UDIFY.

1.3 Discussion

By this point, a notion of what morphological word-formation processes are and how these differ across languages, should have been made clear. More importantly, modelling these linguistic phenomena using dedicated architectures is covered in detail. These systems are evaluated both in terms of general test-set performance, but also on their capacity to lemmatize and annotate novel word-forms and lexical items. All trained systems, both ones re-implemented and novel, prove their competence on both these tasks, with strong performance across a wide variety of languages.

To a lesser extent, this information was already available after the CoNLL/SIGMORPHON 2019 shared task, and is verified to function on general language corpora without the context of a competition. These already make such systems valuable tools for researchers, as will be evidenced in the later chapters of this thesis. However, despite little additional research post-competition, this does not imply this task is 'solved'. While for many languages scores appear to be approaching an upper limit on performance, lower-resource or morphologically rich (and especially both) languages lag behind, considerably. The multilingual pre-training prescribed by UDIFY could improve this facet especially, and with large pre-trained transformers forming the modelling backbone for all considered languages, larger and more varied data might narrow this gap. As done with DogTag or DogTag in later iterations, replacing the transformer backbone with newer, more morphologically aware self-attention architectures, might already yield a better inductive bias. Especially when using character-based transformers, one could step away from lemmatization by classification, rephrasing it as another seq2seq task and yielding systems better at handling open vocabularies. Plenty of additional future research presents itself. For example, typological property prediction during multilingual training has shown a beneficial effect on UDIFY [35]. In summary, while already impressive, new techniques from other NLP domains should be implemented for automated morphological tagging and lemmatization also, likely bridging the gap between higher and lower resource languages.

Appendix

Morphological Tagging and Lemmatization in Context

A

Table A.1: Language merged UD treebanks, filtered by having at least 1 start of quality. Gives the number of constituent treebanks, the number of sentences (thousands), the number of tokens (thousand), the length of the set of genres present in the treebanks, and the average quality in stars.

Language	Family	Treebanks	Sentences (k)	Tokens (k)	Genres	Stars
Afrikaans	IE, Germanic	1	2	49	2	3.5
Arabic	Afro-Asiatic, Semitic	1	8	242	1	3.0
Armenian	IE, Armenian	1	3	52	6	4.0
Belarusian	IE, Slavic	1	25	305	7	4.5
Bulgarian	IE, Slavic	1	11	156	3	4.0
Catalan	IE, Romance	1	17	537	1	4.0
Croatian	IE, Slavic	1	9	199	3	4.0
Czech	IE, Slavic	4	127	2204	5	4.0
Dutch	IE, Germanic	2	21	307	1	2.5
English	IE, Germanic	6	38	608	2	3.0
Estonian	Uralic, Finnic	2	37	511	4	4.0
Finnish	Uralic, Finnic	1	15	202	6	3.5
French	IE, Romance	5	25	559	4	3.5
Galician	IE, Romance	1	1	23	1	3.5
German	IE, Germanic	2	206	3687	3	4.0
Greek	IE, Greek	1	3	62	3	3.5
Icelandic	IE, Germanic	2	51	1142	4	3.0
Indonesian	Austronesian, Malayo-Sumbawan	2	7	148	2	3.5
Irish	IE, Celtic	1	5	116	5	2.0
Italian	IE, Romance	5	34	737	3	3.5
Japanese	Japanese	2	16	344	2	2.0
Latin	IE, Latin	1	9	242	2	4.0
Latvian	IE, Baltic	1	16	266	5	3.5
Lithuanian	IE, Baltic	2	4	75	4	2.5
Norwegian	IE, Germanic	2	38	612	3	4.0
Polish	IE, Slavic	2	39	478	5	4.0
Portuguese	IE, Romance	1	9	211	1	4.0
Romanian	IE, Romance	3	40	937	2	4.0
Russian	IE, Slavic	3	110	1813	1	4.0
Serbian	IE, Slavic	1	4	98	1	4.0
Slovak	IE, Slavic	1	11	106	3	3.5
Slovenian	IE, Slavic	2	11	170	3	3.5
Spanish	IE, Romance	1	18	555	1	4.0
Swedish	IE, Germanic	1	5	91	3	3.5
Tamil	Dravidian, Southern	1	1	9	1	2.5
Telugu	Dravidian, South Central	1	1	6	1	1.0
Turkish	Turkic, Southwestern	6	73	640	2	3.5
Welsh	IE, Celtic	1	2	41	5	2.5

Table A.2: Multilingual pre-training with monolingual fine-tuning.

Model	Language	Lemma Acc.	Lev. Dist.	Morph. Set Acc.	Morph. Tag F1	Throughput
UDIFY Multi+Mono	Arabic	0.94	0.17	0.93	0.97/0.88	2313
	Dutch	0.96	0.08	0.96	0.97/0.97	2507
	English	0.97	0.05	0.93	0.96/0.90	2445
	Finnish	0.91	0.19	0.93	0.97/0.89	1915
	Turkish	0.94	0.13	0.83	0.92/0.73	1407
	Mono. Mean	0.93	0.16	0.89	0.94/0.85	2113
	Mean	0.94	0.12	0.92	0.96/0.87	2117
DogTag Multi+Mono	Arabic	0.93	0.20	0.88	0.94/0.84	1851
	Finnish	0.87	0.29	0.86	0.94/0.83	2106
	Turkish	0.91	0.19	0.78	0.90/0.72	1714
	Mono. Mean	0.90	0.23	0.83	0.93/0.79	1827
	Mean	0.90	0.23	0.84	0.93/0.80	1890

References

[1] Martin Haspelmath and Andrea Sims. *Understanding morphology*. Routledge, 2013 (cited on pages 1, 3).

[2] A. Spencer and A.M. Zwicky. *The Handbook of Morphology*. Blackwell Handbooks in Linguistics. Wiley, 1998 (cited on page 1).

[3] Zhou Zhuang. *Zhuangzi*. A translation from Chinese Thought: From Confucius to Cook Ding by Roel Sterckx, page 152. 300BCE (cited on page 2).

[4] Bernard Comrie. *Language universals and linguistic typology: Syntax and morphology*. University of Chicago press, 1989 (cited on page 3).

[5] John Sylak-Glassman. 'The composition and use of the universal morphological feature schema (unimorph schema)'. In: *Johns Hopkins University* (2016) (cited on pages 4, 10).

[6] Christo Kirov et al. 'UniMorph 2.0: Universal Morphology'. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018 (cited on pages 4, 10).

[7] Khuyagbaatar Batsuren et al. 'UniMorph 4.0: Universal Morphology'. In: *arXiv preprint arXiv:2205.03608* (2022) (cited on page 4).

[8] Arya D. McCarthy et al. '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*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 229–244. DOI: 10.18653/v1/W19-4226 (cited on pages 5, 10).

[9] G.A. Chrupala. 'Simple data-driven context-sensitive lemmatization'. English. In: *Procesamiento del Lenguaje natural, Revista* 37 (2006) (cited on page 6).

[10] Eugene W Myers. 'An O(ND) difference algorithm and its variations'. In: *Algorithmica* 1.1 (1986), pp. 251–266 (cited on page 6).

[11] James Coglan. *Building Git*. James Coglan, 2020 (cited on page 6).

[12] Milan Straka, Jana Straková, and Jan Hajič. 'UDPipe at SIGMORPHON 2019: Contextualized embeddings, regularization with morphological categories, corpora merging'. In: *arXiv preprint arXiv:1908.06931* (2019) (cited on pages 7, 12).

[13] Milan Straka. 'UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task'. In: *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 197–207. DOI: 10.18653/v1/K18-2020 (cited on page 7).

[14] Piotr Bojanowski et al. 'Enriching Word Vectors with Subword Information'. In: *arXiv preprint arXiv:1607.04606* (2016) (cited on page 7).

[15] Armand Joulin et al. 'Bag of Tricks for Efficient Text Classification'. In: *arXiv preprint arXiv:1607.01759* (2016) (cited on page 7).

[16] Jacob Devlin et al. 'Bert: Pre-training of deep bidirectional transformers for language understanding'. In: *arXiv preprint arXiv:1810.04805* (2018) (cited on page 7).

[17] Yoon Kim et al. 'Character-aware neural language models'. In: *Thirtieth AAAI conference on artificial intelligence*. 2016 (cited on page 8).

[18] Kyunghyun Cho et al. 'On the properties of neural machine translation: Encoder-decoder approaches'. In: *arXiv preprint arXiv:1409.1259* (2014) (cited on page 8).

[19] Sepp Hochreiter and Jürgen Schmidhuber. 'Long short-term memory'. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cited on page 8).

[20] Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. 'Residual LSTM: Design of a deep recurrent architecture for distant speech recognition'. In: *arXiv preprint arXiv:1701.03360* (2017) (cited on page 8).

[21] Milan Straka and Jana Straková. 'UDPipe at EvaLatin 2020: Contextualized embeddings and treebank embeddings'. In: *arXiv preprint arXiv:2006.03687* (2020) (cited on page 8).

[22] Yinhan Liu et al. 'Roberta: A robustly optimized bert pretraining approach'. In: *arXiv preprint arXiv:1907.11692* (2019) (cited on page 8).

[23] Dan Kondratyuk. '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*. 2019, pp. 12–18 (cited on pages 8, 12).

[24] Dan Kondratyuk and Milan Straka. '75 languages, 1 model: Parsing universal dependencies universally'. In: *arXiv preprint arXiv:1904.02099* (2019) (cited on page 8).

[25] Ian Tenney, Dipanjan Das, and Ellie Pavlick. 'BERT rediscovers the classical NLP pipeline'. In: *arXiv preprint arXiv:1905.05950* (2019) (cited on page 8).

[26] Jonathan H Clark et al. 'Canine: Pre-training an efficient tokenization-free encoder for language representation'. In: *Transactions of the Association for Computational Linguistics* 10 (2022), pp. 73–91 (cited on page 9).

[27] Ashish Vaswani et al. 'Attention is all you need'. In: *Advances in neural information processing systems* 30 (2017) (cited on page 9).

[28] Arya D. McCarthy et al. 'Marrying Universal Dependencies and Universal Morphology'. In: *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 91–101. DOI: 10.18653/v1/W18-6011 (cited on pages 9, 10).

[29] Adam Paszke et al. 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035 (cited on page 10).

[30] Diederik P Kingma and Jimmy Ba. 'Adam: A method for stochastic optimization'. In: *arXiv preprint arXiv:1412.6980* (2014) (cited on page 10).

[31] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/ (cited on page 11).

[32] Antonios Anastasopoulos. *A note on evaluating multilingual benchmarks*. URL: http://www.cs.cmu.edu/~aanastas/evaluating%5C_multilingual.html (cited on page 11).

[33] Jacob Cohen. *Statistical power analysis for the behavioral sciences*. Routledge, 2013 (cited on page 13).

[34] Larry V. Hedges. 'Distribution Theory for Glass's Estimator of Effect Size and Related Estimators'. In: *Journal of Educational Statistics* 6.2 (1981), pp. 107–128. (Visited on 07/01/2022) (cited on page 13).

[35] Chinmay Choudhary. 'Improving the Performance of UDify with Linguistic Typology Knowledge'. In: *Proceedings of the Third Workshop on Computational Typology and Multilingual NLP*. Online: Association for Computational Linguistics, June 2021, pp. 38–60. DOI: 10.18653/v1/2021.sigtyp-1.5 (cited on page 15).