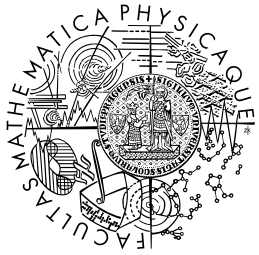


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Jonáš Vidra

Morphological segmentation of Czech Words

Institute of Formal and Applied Linguistics

Supervisor of the master thesis: doc. Ing. Zdeněk Žabokrtský, Ph.D.

Study programme: Computer Science

Study branch: Computational Linguistics

Prague 2018

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

I dedicate this thesis to Zdeněk Žabokrtský and Magda Ševčíková, and I thank them for their support.

Title: Morphological segmentation of Czech Words

Author: Jonáš Vidra

Institute: Institute of Formal and Applied Linguistics

Supervisor: doc. Ing. Zdeněk Žabokrtský, Ph.D., Institute of Formal and Applied Linguistics

Abstract: In linguistics, words are usually considered to be composed of morphemes: units that carry meaning and are not further subdivisible. The task of this thesis is to create an automatic method for segmenting Czech words into morphemes, usable within the network of Czech derivational relations DeriNet.

We created two different methods. The first one finds morpheme boundaries by differentiating words against their derivational parents, and transitively against their whole derivational family. It explicitly models morphophonological alternations and finds the best boundaries using maximum likelihood estimation. At worst, the results are slightly worse than the state of the art method Morfessor FlatCat, and they are significantly better in some settings.

The second method is a neural network made to jointly predict segmentation and derivational parents, trained using the output of the first method and the derivational pairs from DeriNet. Our hypothesis that such joint training would increase the quality of the segmentation over training purely on the segmentation task seems to hold in some cases, but not in other. The neural model performs worse than the first one, possibly due to being trained on data which already contains some errors, multiplying them.

Keywords: morpheme morphology segmentation stemming

Contents

1. Introduction

One of the key notions in structural linguistics, and modern linguistics in general, is the one of arbitrariness – among other uses of the term the arbitrariness of vocabulary. The fact that the vocabulary of a language is arbitrary means that the meaning of a word generally cannot be inferred from its form (i.e. the spoken sound waveforms or the written characters) and vice versa – the user of a language has to learn the form-meaning mapping for each word separately. There are specific cases of non-arbitrariness (e.g. interjections – imitations of natural sounds tend to be similar to those sounds), but in general, "vocabulary is arbitrary" is a fact often repeated in introductory textbooks.

As with all introductory textbook facts, this statement is merely a simplification of reality. Even the founders of modern structural linguistics knew that the meanings of many words may be inferred (de Saussure, 1961). For example, if a student of the Czech language already knows the words *komín* ("chimney"), *zahrada* ("garden"), *kominík* ("chimney sweep"), *zahradník* ("gardener") and *tanec* ("dance"), they may infer the meaning of the word *tanečník* ("dancer") by noticing a recurring pattern in the language: *kominík* and *zahradník* can be decomposed into *komín* + *-ík* and *zahrada* + *-ík*, with some fuzziness – the process is not a simple concatenation, but it is close enough. The *-ík* part of the composite word modifies the meaning of the base word, signifying "a person doing something with the base object". Thus *tanečník* is "a person doing something with a dance" – and a *dancer* seems to be the most likely candidate for such a role.

The preceding paragraph implies that words have internal structure and that this structure is important to users of the language. Such a statement is in line with the findings of modern linguistics (Carstairs-McCarthy, 2005; Haspelmath et al., 2015), and in fact, a whole subfield of linguistics, called morphology, is devoted to studying this structure, its importance and implications (Aronoff; Fudeman, 2011, p. 1).

The importance is not merely theoretical. Different ways of splitting words into some constituent sub-units are employed in many linguistic applications. To give a few examples: hyphenation is used to break long words at the end of a line of text; subword units are used in neural machine translation to compress the dictionary and allow translation of previously unseen words (Sennrich et al., 2016); morphological segmentation à la the motivating example above is used for teaching foreign languages; and some segmentation is employed by search engines to find your desired text in sources where it doesn't occur verbatim, but only in variation (Uyar, 2009).

For each of the examples above, a different method of segmentation is, or may be, used. Hyphenation generally occurs at syllable boundaries in Czech (Martincová et al., 1993), but British English prefers breaks at morphological boundaries (Hladký, 1987)[1]. These often give different results: syllabification of the Czech word *větrat* ("to ventilate") is *vě·trat*, but its morphological segmentation is *větr·a·t*. Translation subword units are often technical and motivated by compression methods (Sennrich et al., 2016). The advantage of *morphological segmentation*, which is to be the focus of this thesis, is that the segments correspond with the meaning of the word.

[1] In English in general, the rules are complex and inconsistent. See e.g. the different hyphenation patterns for the word *interferometer* given by the Merriam-Webster dictionary (*in·ter·fer·om·e·ter*) and the American Heritage dictionary (*in·ter·fe·rom·e·ter*).

1.1 Task statement

The aim of this thesis is to create a linguistically adequate method for automatic segmentation of natural language words into morphs, with a focus on the Czech language. The result is supposed to be used by the DeriNet project (Žabokrtský et al., 2016), which aims to expand its annotation with information about morphematic composition of lexemes in the next major version, 2.0.

The method is to be based on machine learning techniques. In particular, we chose two different algorithms, one supervised, which learns to segment words by looking at correctly segmented examples, and one unsupervised, which segments words by utilizing statistics of probable segments inferred without looking at premade examples. The unsupervised model is based on maximum likelihood probability estimation, bootstrapped using the Expectation-Maximization algorithm. The supervised model is based on neural networks. The examples needed for training the neural network will be created as a part of the thesis, using the unsupervised model.

Note that the difference between supervised and unsupervised learning is quite thin here, because even the unsupervised algorithm utilizes expensive, manually annotated data, thus potentially also qualifying as "supervised". However, the machine learning method used inside the model is of the unsupervised family.

The task itself is found under many different names in literature, and there is some confusion in the naming. Most commonly, it is called morphological (or morphematic, or morpheme) segmentation (or boundary detection, or analysis). Other terms include "segmentation of words" (Creutz, 2003), "discovery of morphemes" (Creutz; Lagus, 2002), "morphological decomposition" (Lehtonen et al. (2011), used especially in the psycholinguistic, cognitive and neurolinguistic circles), "morphological parsing" (Karttunen et al., 2001) and "word segmentation" (Hafer et al., 1974).

Some of these terms are also used for other tasks: the term "word segmentation" is primarily used for the task of segmenting running text into words (i.e. word boundary discovery) and morphological parsing usually means creating parse trees over words, i.e. annotating not only the boundaries, but also the order of affixation or modification. All these names, but predominantly the names which contain the term "morpheme", are also used for the very similar task of segmenting words into canonical segments (i.e. decomposing *beaches* into *beach* + *-s*, or even *men* into *man* + *-s*) or for morpheme analysis (*beach* + PLURAL).

Our task is focused on the surface segmentation, without such canonicalization or high-level annotation of the results – we want to segment the word into morphs represented as contiguous substrings, which, when concatenated together in the original order, form the whole word (i.e. no characters are missing). Stated differently, we want to find segmentation points (segment boundaries) in words.

1.2 Prior publication disclaimer

Parts of the work presented here have already been published elsewhere:

- The Longest Common Substring algorithm and the boundary propagation algorithm were published by Macháček et al. (2018). Despite the co-authorship with Dominik Macháček and Ondřej Bojar, I am the sole author of all the parts of my thesis published therein.

- I submitted a slight modification of the parent-prediction part of the neural network segmentation model to a state-of-the-art Czech lemmatization competition organized as part of the Deep Learning course (NPFL114) at the Charles University, Faculty of Mathematics and Physics during the 2017-2018 academic year.

  The code is based on a template by Milan Straka, containing the data-loading and result-printing code, which was necessary for submission. The architecture of the neural network itself, the code implementing it and its training, optimization and evaluation were selected, written and conducted entirely by me.

1.3 Outline

The first three chapters of this thesis are introductory. Chapter 2 introduces the important terms and concepts, with Section 2.1 focusing on the linguistic theory and Section 2.2 on the definitions used when processing language on a computer.

Chapter 3 presents the data resources used by our models and related work in the form of other segmentation tools.

Chapter 4 introduces our unsupervised segmentation model. There, we explain both our novel probabilistic formulation of the segmentation task using derivational data, and the well-known techniques used to create and train the model.

In Chapter 5, we do the same for the supervised model. Again, the methods used to construct and train the neural network are desribed in referenced literature, but the concrete form of the network and the idea of using derivations to help train a segmentation model are novel.

Chapter 6 contains a desription of the experiments we performed to compare our systems with the state-of-the-art, the results we obtained and their interpretation and explanation.

Chapter 7 concludes the thesis.

2. Basic notions

In this chapter, we introduce and explain basic terms and notions that will be used throughout the rest of the thesis. The first section deals with linguistic theories and the theoretical view on language. In the second section, we define our own terminology for dealing with segmentation on a technical, character-string-wise level.

2.1 Linguistic terminology

Although the aim of this thesis is producing segmentation, we will start our linguistic introduction with derivation, since derivation is the process we will use for inferring segmentation afterwards.

2.1.1 Derivation

Derivation is one of several word-formation processes in natural languages. It creates new words out of existing ones, usually by adding, changing or removing affixes (Dokulil et al., 1986; Aronoff; Fudeman, 2011). For example, we can derive the word *friendly* from the word *friend* by adding the suffix *-ly*. Derived words can themselves serve as bases for further derivation, such as the words *unfriendly* or *friendliness* derived from *friendly*, and so the sets of words related by derivation form directed graphs (Aronoff; Fudeman, 2011). We will use the terms *(derivational) parent* for the base word and *(derivational) child* for the derived word, and together we will call them *derivational pair*.

Derivation is best defined in comparison and contrast with similar language processes: It can be compared with inflection (e.g. *shelf* $\longrightarrow$ *shelves* or *print* $\longrightarrow$ *printed*), a process that is not word-formational, but which is usually very similar in the changes it induces; and with other word-formation processes such as compounding (putting together two independent words, e.g. *skin* + *head* $\longrightarrow$ *skinhead*) or conversion (changing the category of a word without changing its form, e.g. *(a) ship* $\longrightarrow$ *(to) ship (something)*), which induce different changes.

The difference between word-formation (those that create new words) and non-word-formation (those that create forms of the base word) processes is blurry, but some of its aspects are discussed below. The main point is that inflection changes the word to fit grammatical categories in a sentence, while derivation changes the word to fit a desired meaning. Another possible comparison, discussed below, is between derivation as a synchronic view of language, and etymology as diachronic view of language.

As a concept of western linguistic tradition, derivation is easy to identify in languages where utterances break down to words, which are viewed as being made up of morphemes. This is the case in Indo-European languages. In languages which don't have the word as a core concept, derivation is either considered not to exist, or is difficult to define. For example, in Mandarin (Standard) Chinese, most words are single-morpheme and multi-morpheme words are generally considered to be compounds, and thus prototypical derivation through affixation is very limited. See Arcodia (2013) for a discussion of the status of Chinese derivation.

In this thesis, we will focus mostly on Czech, where derivation is well established and mostly well defined, as it has been the object of scientific inquiry for centuries (Dobrovský, 1791). Only this chapter is more diverse as to the considered set of languages and phenomena, because it has the secondary purpose of drawing attention to the possible shortcomings of our methods in multilingual settings.

2.1.2 Morphemes

Derived and compound words are also sometimes called *motivated words*, as opposed to the basal *unmotivated words* (Dokulil et al., 1986). These terms describe the difference in the sign nature of the words – unmotivated words are arbitrary[1], there is no connection between their phonetic or orthographic form and their meaning outside the word itself. The individual phones or letters do not mean anything on their own. To a speaker of the language, motivated words do have such a connection, as they are composed of parts which carry the meaning. For example, the meaning of the word *happiness* can be approximated by taking its motivating word *happy* and modifying it with the affix *-ness*, whose meaning can be inferred by analogy from words such as *closeness*, *holiness* or *darkness* to be "the state of being *X* (the base word)". A speaker can thus infer the meaning of a novel, previously unheard word, by mentally decomposing it into its base parts and relating it to known words with similar compositions.

The smallest such recurring parts which carry meaning, are called *morphemes* (de Saussure, 1961). Their exact properties depend on the theory you follow, but in general, they have the three following ones:

1. Morphemes carry meaning.
2. Morphemes are not subdivisible into smaller meaning-carrying units.
3. Morphemes are repeatable – one morpheme occurs in multiple contexts, i.e. in multiple words.

The third property is especially important for morpheme discovery and is often used both by automatic segmentation systems and as a criterion for manual segmentation (Slavíčková, 1975).

All three of these requirements may be violated, though. According to proponents of cognitive and usage-based theories, this is due to the inherent fuziness of language, which is a result of the constant evolution of our mental grammar, as it is being affected by experience and other cognitive factors. As stated by Bybee (2013), "we should not expect linguistic constructs such as segment, syllable, morpheme, word, or construction to have strict definitions, nor do we expect all the manifestations of these constructs in languages to exhibit exactly the same behavior".

Examples of morphemes carrying very little to no discernible meaning are interfixes (connectors) in compounds, such as *speed·o·meter*, or the apparent prefixes in words like *de·ceive*, *de·sist*, *re·ceive* and *re·sist*. Due to this, some authors reject

[1] Onomatopoeic words and most interjections are the exception. They are (extralinguistically) motivated by the natural sounds they imitate. But even they have some degree of arbitrariness: The English imitation of the sound of a cow is /mu:/, while the Czech version is /bu:/, even when imitating the same cow. And some words with onomatopoeic origin have developed to be nearly fully arbitrary, such as *pidgeon*, originally from the onomatopoeic Vulgar Latin *pipio* (de Saussure, 1961).

their morphemic status and analyze these prefixed words as single-morpheme units (Marchand, 1969). This, on the other hand, creates a problem with subdivisibility, as the facts that these subunits can be freely recombined, and that the prefixes occur in contexts where they do have a clear meaning (*re·load*, *re·paint* – "to do something again").

Some linguists propose theories of morphology without morphemes (Neuvel et al., 2002), or suggest that some sub-morpheme language phenomena are difficult to explain in theories involving morphemes (Zingler, 2017). However, apart from trivial arguments such as the one made by us at the beginning of this section, there is also neurolinguistic evidence for morphemes (Lehtonen et al., 2011). We therefore simplify matters by considering their existence to be given.

2.1.3 Morpheme classification

Morphemes can be classified into categories based on their form, function and position.

A basic categorization based on form is into continuous and discontinuous morphemes. Continuous morphemes have contiguous phonological or orthographical representations, i.e. they are not split into multiple parts by phonemes or characters that belong to other morphemes. Discontinuous morphemes are the opposite, they contain spaces for other morphemes or their parts inside them (Bauer, 2014, p. 124).

One possible categorization based on function is into lexical (have meaning by themselves) and grammatical (modify the meaning or inflectional features of lexical morphemes) morphemes – with a whole spectrum in between, for example affixoids (Olsen, 2014).

Another opposition is free vs. bound morphemes – free morphemes may form a single-morpheme word on their own, while bound morphemes must attach to another morpheme or a group of already attached morphemes. Such a group is often called the *base* or *stem*. We will use the term stem throughout the rest of this thesis, knowing that it is a slight departure from the standard usage, which reserves the term for the base to which only inflectional affixes are attached, see below.

Another classification is into roots, which form the base of a word and usually coincide with lexical morphemes; and affixes, which attach to the roots and usually coincide with grammatical morphemes. Continuous affixes can be further divided into prefixes, which stand before the root or stem they modify; suffixes, which stand after it; infixes, which interrupt another morpheme and stand inside it, and interfixes, which stand between two stems in compounds, belonging to neither (Bauer, 2014).

Discontinuous affixes come in many forms, many examples from the complex field of nonconcatenative morphology[2] (Davis et al., 2014) belong to this class. The one most relevant for this thesis is circumfixation, where the affix modifies the stem from both sides at the same time, as if it was simultaneously attaching a prefix and a suffix. Clear examples of circumfixation are rare, because it must be argued why the affix is not composed of its two parts (Bauer, 2014, p. 127). In Czech, examples such as *les* ⟶ *polesí* are sometimes considered to be circumfixation because there is neither **poles* nor **lesí*, but since both the prefixal and the suffixal parts occur on their own in other contexts, they can be also interpreted as two affixations.

[2] Also called parasynthesis.

Other nonconcatenative processes include root-and-pattern derivation[3], common in Semitic languages, where the root is composed of consonants, the affix of vowels, and they intermesh to create a pronounceable form (Shay, 2014). As these processes do not occur in Czech and therefore in our data, we will not consider them further.

2.1.4 Morphs

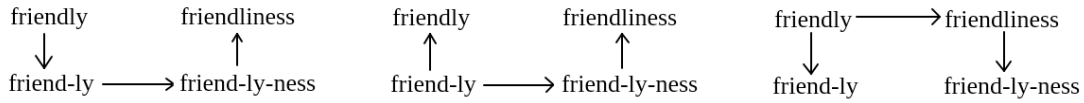
The mental decomposition of a word into morphemes is not problem-free. One of the issues is that morphemes do not have a unique surface representation. A single morpheme may have multiple possible forms, each used in a different context. For example, the English plural morpheme has two variants in writing: *-s* as in *book* → *books*, and *-es* as in *beach* → *beaches*; and three in speech: *[z]* as in *bed* → *beds*, *[s]* as in *book* → *books*, and *[əz]* as in *beach* → *beaches*. Some theories even consider *children*, *cacti*, *sheep* or *women* to contain the same morpheme, just in a form very different from the typical ones (Aronoff; Fudeman, 2011, p. 74; Spencer, 1994), while other will say these are different morphemes or even results of non-morphological processes such as suppletion (complete replacement of one form with another, unrelated form), just with the same semantics (Mel'čuk, 2006, p. 309).

Such variants are called *allomorphs* and their existence is the reason why we distinguish the abstract morpheme as a unit of meaning from its concrete realization, the *morph*.

The selection of allomorphs is not fixed when a morpheme is added to a stem, but the selected allomorph can change after that, when other morphemes are added. For example, in the Czech pair *okno* ("window") → *okenní* ("of the window"), the morph *okn* is replaced with *oken* when the suffix *-ní* is added. An alternative description of the same example is that the morph *okn* is modified with a phonological *alternation* (phonological *change*), namely a vowel insertion. The first description fits with the so-called item-and-arrangement theories, while the second one is in line with item-and-process theories (Hockett, 1954). This thesis will focus on the item-and-process style of describing morphology. Although linguistically, there are several sources of these changes (morphological, orthographical or phonological ones), we will treat them identically and thus denote them simply as *MOP alternations*.

Another issue is that the boundaries between morphs may not always be clear. This point is connected to the previous one, as the fuzziness of the boundaries is, in large part, due to MOP alternation. For example, *pamětník* ("witness$_{\text{Masc.}}$") → *pamětnice* ("witness$_{\text{Fem.}}$") may be segmented as *pamětní·k* → *pamětni·ce*, with an MOP alternation, or as *pamětn·ík* → *pamětn·ice*. Dokulil et al. (1986) calls the *í* or *i* from this example a *submorph* – it is separable on the basis of repeatability, but it does not have a sign nature, as it lacks a clearly defined meaning. They are therefore attached to one of the neighboring morphs, forming an *extended variant* of that morph, and do not stand on their own in segmentation (Dokulil et al., 1986, p. 177). In this case, attaching it to the suffix is the more plausible solution, as there are very few examples of *-k* → *-ce* binding to stems not ending in *-í* → *i* (one would be *pavoučice* ("female spider"), whether it is derived from *pavouk* ("spider") or *pavouček* ("little spider")), making *-ík* → *-ice* the usual form of the morphemes, and there are other words with the stem *pamětn*, such as *pamětný* ("memorable").

[3] Also called templatic morphology.



The view of Slavíčková **The view of Mel'čuk** **The view of Aronoff**

Figure 2.1: The difference in views of selected linguists on derivation and morphemes. The top line shows words, the bottom line their segmentation into morphemes. Arrows indicate the direction of information transfer.

Another prevalent reason for unclear boundaries is fusion of phonemes at morph boundaries, which occurs e.g. in *Rusko* ("Russia") → *ruský* ("russian"), where the *-s-* in *ruský* is a fusion of double *s* from *rus·ský* (Dokulil et al., 1986).

These problems imply that, in many languages, the detection of morphemes or morphs is far from easy. In Czech, there are derivational pairs with many MOP alternations occuring at once, such as *ode·jm·ou·t* ("to confiscate") → *od·ň·a·t·ý* ("confiscated"), where only one morpheme retained its allomorph without changes. Even knowledge of the segmentation of one of the words doesn't necessarily imply we can easily segment the other. This also illustrates that the changes are not local, but it is possible for a change to occur e.g. in a prefix when a suffix is being added.

The significance of morphemes and their relation with derivation varies from author to author. For example, Slavíčková (1975) considers morphemes to be the hidden elemental basis on which derivation is built. Meaning-text theory goes even a step further, considering morph(eme)s[4] to be the basic block – that is, words are not analyzed into morphemes, they are actually built out of them (Mel'čuk, 2006, p. 388). On the other hand, Aronoff (1985) claims that morphemes are secondary – words are formed directly from other words and morphemes are just a secondary theoretical explanation of the process. The difference between these views is seen in Figure 2.1.

This difference stems from the difference in general approaches to language, as Mel'čuk (2006) takes a deductive approach, considering semantics to be primary and the surface phonetical or orthographical form to be derived (Mel'čuk, 2006, p. 12). The apparent reversal of this general deductive trend when considering morphemes and morphs is due to him not considering morphemes to be signs, but a set of signs, which the underlying morph manifests. The view of Dokulil et al. (1986) is largely compatible with this approach.

Slavíčková (1975) takes a slightly different approach, as according to her, the morphemic composition of a word is not given, but a word has to be analyzed to reveal it first. It is possible that this difference is caused by the fact that she was, in fact, trying to analyze Czech words and deduce their morphemic makeup.

The difference is best revealed on border cases, such as cranberry morphemes. These morphemes violate the repeatability property by only appearing in a single word (and optionally its handful of derivatives), and as such, it is difficult to assign them any meaning other than the meaning of the word they are found in. They are revealed as residuals after identifying the other morphemes in the word. A classic example is the word *cranberry*, after which the whole group is named. Its morpheme *berry* is easy to identify by analogy with *blackberry* or *strawberry*, but the residual morpheme *cran* is not found elsewhere. Such morphemes present a greater obstacle

[4] Mel'čuk considers morphs to be primary and defines morphemes on top of them.

for theories à la Mel'čuk (2006), which depend on morphemes being well defined, than for theories of Slavíčková (1975) and Aronoff (1985).

2.1.5 Derivation vs. inflection

Derivation and inflection are both morphological processes that create new lexical forms out of existing ones.

The textbook difference (paraphrased from Aronoff; Fudeman, 2011) between the two is that the units created by derivation are new words, complete with their own semantic and syntactic properties, while the units created by inflection are only forms of the same word and their semantics is either identical, or non-idiosyncratically related, to the semantics of their lemma (the main word form of the word). Typical properties of inflection include peripherality (usually occurs at the beginning or end of a stem), obligatoriness (some inflectional marker must be always present) and non-repeatability (there may be at most one inflectional marker), while for derivation, the opposite properties may hold.

Therefore, the English word pairs *window* → *windows* is an example of inflection, since the addition of the suffix *-s* causes a predictable change in meaning from "one of a thing" to "multiple of that same thing". And the word pair *window* → *windowed* is an example of derivation, because the change from "a thing" to "covered in / containing / fitted with that thing" is specific to the word window and doesn't occur e.g. in the pair *tape* → *taped*, which may mean either "recorded on tape" or "attached with tape".

This definition, of course, depends very strongly on the definition of a word (which will not be discussed further, as we work with existing dictionaries, thus leaving the decision to others); and on the exact amount of idiosyncrasy a relation must accumulate to step over the line from inflectional to derivational. As discussed by ten Hacken (2014), the boundary between them is fuzzy with unclear corner cases. For example, diminution is considered to be a part of inflection by some linguists and a part of derivation by others (ten Hacken, 2014).

Another example is negation in Czech, which is traditionally considered to be an inflectional process and is even listed in inflectional dictionaries (Hajič, 2004), but negated words can e.g. form the basis for derivation, violating the prototypical peripherality (*zneplatnit* ("to invalidate") from *neplatný* ("not valid")), and there are negated words with a large semantic gap from the positive word (*neřád* ("brat") from *řád* ("order, law")) or entirely without positive counterparts (*nemotorný* ("clumsy"); the apparent base word *motor* ("engine") is etymologically unrelated and semantically distant). As reported by Blust (2014), in many Austronesian languages, the distinction between inflection and derivation is even less clear than in Indo-European languages.

Other differences between inflection and derivation are related to this main semantical difference. For example:

- Inflection is paradigmatic, i.e. any word sharing the same paradigm with window will also pluralize, and will do so in the same way. In this case, this paradigm covers all count nouns. Derivation is less regular and words with the same derivational patterns do not form easily identifiable groups. One reason for this is that many derived words are lexicalized and the derivations used to create them are no longer productive. Sometimes, the motivating word may even no longer be a part of the active lexicon, e.g. the Czech word *mnít* ("to

deem"), even though its derivatives like *pomník* ("a memorial") or *domněnka* ("a conjecture") are very common.

- Inflection doesn't cause a change in the part-of-speech, while derivation may (Aronoff; Fudeman, 2011). This might, however, be a circular definition, since parts-of-speech are usually distinguished on the grounds of common inflectional behavior.

A second definition is considered by ten Hacken (2014) to be better than the one above: "Inflectional morphology is what is relevant to the syntax." However, as reported by Stump (2005), while this definition holds clearly for *contextual* inflection, such as agreement in adjectival number; it is not so clear for *inherent* inflection, e.g. nominal inflection for number.

Despite this fuzziness, the distinction between derivation and inflection still has its merit, as the difference shows up in psycholinguistic studies – see e.g. Feldman (1994), who measured the time it takes subjects to perform simple textual tasks on pairs of words, either related by inflection, by derivation or not related. The study reports that the time is influenced by the type of relatedness and that there is a difference between inflection and derivation.

2.1.6 Derivation vs. other word-formation processes

Word-formation processes differ in the underlying morphological processes they use to create new words. The exact repertoire of word-formation processes varies by author, as does their classification. For example, Aronoff; Fudeman (2011) consider derivation to be an all-encompassing term for all word-formation processes, including compounding, conversion, and even ones that Mel'čuk (2006) considers non-morphological, such as blending, clipping and acronymization. They call the Dokulil's derivation "affixation". This is, however, mostly a terminological issue.

Some authors consider conversion to be a distinct word-formation process, others view it as a special case of derivation using zero affix, some define and distinguish both cases of conversion (Dokulil et al., 1986, p. 201) and yet others view it as a morphological process that can be used in different word-formation processes as well as in inflection (Mel'čuk, 2006). The latter view shows that the word "conversion" has multiple meanings – one relates to derivation, the other one, used by Mel'čuk, relates to affixation.

In our models and algorithms, we will not consider compounding at all, conversion will be interpreted as zero affixation, and other processes, to the extent they appear in our training data, will be ignored.

2.1.7 Derivation vs. etymology

Both derivational morphology and etymology study the origins of words. The difference is in the type of origin they consider. Etymology is the study of how words came to be; how they were first created and introduced into the language. Derivation is the formation process that takes place in the speaker's minds. This means that etymology is diachronic, while derivation is synchronic. In basic cases, such as those given in previous sections, these processes match.

But there are many examples of differences. To give one of them, the word *hamburger* etymologically comes from *Hamburg*, the name of the city in northern Germany – but after the shortened form *burger* was created along with *beefburger*, *cheeseburger* etc., the original form was reinterpreted and in the minds of many modern speakers, it is created via compounding from the words *ham* and *burger* (Trips, 2014).

Similar reinterpretation is not limited to compounding, but can be found in derivation as well. For example, the Czech words *odér* ("odor") and *deodorant* ("deodorant") are etymologically loanwords from French. But they may be interpreted as being (indirectly) derivationally related without the round-trip to and from French, by analogy with words such as *mutant* ("mutant"), *emigrant* ("émigré") or *konzultant* ("consultant").

It is also possible to reinterpret certain cases of back-formation (affix removal) as standard affixation by flipping the direction of derivation. This approach further disconnects derivation from etymology, but regularizes the language processes, and thanks to this, it is taken by some creators of derivational databases (Ševčíková et al., 2016). A good guiding principle to such direction reversal is one of frequency: it has been reported (Furdík, 1978; Panocová, 2017) that derivational parents are generally more common than their children.

2.2 String-wise view

In this section, we look at the technical application of linguistic terms introduced in the previous section. Since our resources are all textual, we will focus on the written side of language rather than on the phonological one. In Subsection 2.2.1, we define a technical interpretation of the linguistic terms. In Subsection 2.2.2 and Subsection 2.2.3, we outline the concept of using derivational data for inferring segmentational information.

2.2.1 Definitions

Alphabet, Word, Language

Language is built on top of an alphabet Σ, which is a finite nonempty set of characters. A word w is defined as a nonempty string (a nonempty sequence of characters) from Σ, i.e. $w = (l_1, l_2, ..., l_n) \in \Sigma^+$. A language $\mathcal{L}$ is then a set of words: $\mathcal{L} \subset \Sigma^+$. These definitions are mostly consistent with the standard formal grammars definition (Barták, 2013), except that we do not allow the empty word to be in the language.

These definitions are a simplification, because in natural languages, the alphabet is not well-defined. For example, texts in any given language may contain words written in a foreign language using their native alphabet, and in the case of Czech, there is the digraph *ch*, which is usually considered a single letter (e.g. *pře·chod·n·ý* ("transient") is 8 letters long), but is hard to distinguish from the letter combination *c*+*h*, found e.g. in *na·nic·hod·n·ý* ("not good for anything"). We consider *ch* to always be two characters regardless of context.

Also, the definition of a language as a set of words is problematic, because it implies that people with even slightly different vocabularies speak different languages, and that language changes every time a neologism is introduced. This issue manifests itself in our EM model, where any word not encountered during training (out of

vocabulary word, OOV) has to be left unsegmented.

Substrings

To cut out parts of strings, we define the following three functions:

substrings(w) is a multiset of all contiguous substrings of w (including the empty ones) generated by iterating over all possible positions,

prefixes(w) is a set of all contiguous substrings of w (including the empty one) that start at the beginning of the string, and

suffixes(w) is a set of all contiguous substrings of w (including the empty one) that terminate at the end of the string,

as given by the following definitions, where (l_x, l_{x-1}) denotes the empty string:

$$\forall w = (l_1, l_2, ..., l_n):$$
$$\text{substrings}(w) = \{(l_i, ..., l_j) \mid i \in \{1, ..., n+1\}, j \in \{i-1, ..., n\}\} \tag{2.1}$$
$$\text{prefixes}(w) = \{(l_1, ..., l_j) \mid j \in \{0, ..., n\}\} \tag{2.2}$$
$$\text{suffixes}(w) = \{(l_i, ..., l_n) \mid i \in \{1, ..., n+1\}\} \tag{2.3}$$

Since substrings(w) is a multiset, substrings("eye") = {"eye", "ey", "ye", "e", "y", "e", "", "", "", ""}

We will also sometimes use λ as a symbol to denote the empty string.

Derivation

Words can be either unmotivated or motivated. Motivated words have exactly one derivational parent, with which they are related. Unmotivated words have no parent. Either kind may have any number of derivational children. This means that words can be ordered into directed derivational trees, with an unmotivated derivational root and motivated branches and leaves.

Morphs

Each word is composed of one or several morphs, which are nonempty disjoint (non-overlapping) contiguous substrings of characters, whose concatenation forms the whole word. Therefore, a morph must be a nonempty substring of a word. Since a morph must be contiguous, circumfixes are interpreted as a pair of morphs, one on each end of the word. Zero morphs are not allowed.

The fact that a suffix or a prefix may be empty, while morphs must be nonempty, seems to be contradictory at the first glance. It is better understood when looking at morphs as strings obtained by splitting a word by segmentation points. Since the beginning and end of a word already contain an implicit segmentation point, the discovery of an empty affix does not change the list of segmentation points and therefore the morph composition of the word. Prefixes and suffixes are understood as substrings of the word, not necessarily as morphs in their own right – an affix may span several morphs as well as none.

Stems

This segmentation of a word into its morphs is created by a peripheral affixation process, which means that the immediate composition of each word is a stem and an optional single affix at each side of the stem. The stem must be nonempty (i.e. must contain at least a single morph), while the affixes may be empty, signifying the lack of an affix. Both may be multi-morph, must not overlap with one another, and together, they must cover the entire word.

The stem of a motivated word is shared with its derivational parent, i.e. it is descended from a contiguous subset of its derivational parent's morphs. This sharing-descendence relation doesn't necessarily mean that the child's stem is copied from the parent verbatim, as it may be modified by morphological, orthographical and phonological alternations (MOP alternations). Also, the corresponding subset of the parent's morphs need not be its stem as per the definition above, e.g. in *jaro* ("$\text{spring}_{\text{Noun}}$") → *jarní* ("$\text{spring}_{\text{Adj}}$") → *jarně* ("$\text{spring}_{\text{Adv}}$"), the stem of *jarní* is *jar*, but the mapped part from *jarně* is *jarn*. We will call this parent's subsequence *mapped stem*.

Due to the peripherality of affixation, given a particular word $w \in \mathcal{L}$ of length $|w| = n$, a segmentation of w into its (optional) prefix, stem and (optional) suffix can be expressed as finding two boundaries $\{b_1, b_2\} \in \mathbb{N}_0, b_1 < b_2 \leq n$, which segment the word into its prefix $p = (l_1, ..., l_{b_1})$ (nonexistent if $b_1 = 0$), stem $r = (l_{b_1+1}, ..., l_{b_2})$ (always nonempty) and suffix $s = (l_{b_2+1}, ..., l_n)$ (nonexistent if $b_2 = n$).

2.2.2 Stemming using derivations

The definition of a stem given above contains the base of a possible stemming algorithm. If we could find the contiguous set of morphs descended from the derivational parent, we would obtain the stem. The rest of the word's morphs then belong to the affixes. An algorithm that performs exactly this search for common stems and changed affixes is presented in Chapter 4.

2.2.3 Segmentation using stemming

With a working stemming algorithm, we can to transform information about stems into information about morphs. This is done using a secondary resegmentation algorithm. After stemming a word, one or two morph boundaries are usually already present and correct[5]. However, the stem itself is often a multi-morph string, and sometimes even the affixes can be multi-morph, as in *z·vrát·i·t* ("to overturn") → *z·vrác·e·n·ý* ("perverse"), where the suffixes *it* and *ený* contain two and three morphs, respectively, and the stem *zvrác* is composed of two. We can, however, subdivide stems by *repeated stemming*, i.e. by looking at the segmentation of a stem in the word's derivational parent or children and transferring the boundary. See Figure 2.2 for an example.

2.2.4 Type and token frequencies

Both our models utilize information about frequencies of observed phenomena, such as morphs, affixes, stems, individual characters or character substitutions. The

[5] An exception would be e.g. conversion, such as *strážný* ("tutelary") → *strážný* ("a guardian"), where the stem forms the whole of both words and thus no boundary is discovered.

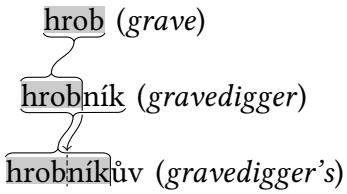


Figure 2.2: An illustration of how repeated stemming yields a morphological segmentation. Stems are indicated by gray background, morph boundaries between stems and affixes by a solid vertical line. Pairs of connected braces indicate how a stem of the word below maps to a part of the word above. A dashed vertical line in the bottom word indicates how its stem can be subdivided using a morph boundary information from the middle word.

probabilistic model directly, in the form of internal count tables, the neural model indirectly, as phenomena frequently encountered during training have a greater chance of influencing its parameters than infrequent ones.

We have an option of changing the frequencies of inputs our models encounter by training on different datasets or by changing the dataset. One decision that is often done is training on word type frequencies or on word token frequencies. A type frequency is 1 for each word that exists in a dictionary (except for homonyms, which share one word form among several conceptually different lexemes) and 0 for unseen words, while a token frequency of each type is given by its frequency in a corpus.

The distribution of type frequencies is therefore uniform – as if we had a corpus where every plausible word type was encountered equally often. Such a corpus is absolutely unrealistic, because token frequencies in real world corpora generally follow a power law distribution: several word types are encountered very frequently and most types are rare (Zipf, 1932).

The chosen mode of training influences the results. As shown by previous studies (Creutz; Lagus, 2005) for similar algorithms, training on type frequencies increases the recall of the algorithm by a large margin, while simultaneously decreasing precision by a little bit, when compared to training on token frequencies. Overall, using type frequencies in training seems to be a better fit for our model.

An even better mode of operation is to interpolate between type and token frequencies, as reported e.g. by Goldwater et al. (2005), who justifies the better results obtained by such interpolation by comparing the algorithm with the Kneser-Ney smoothing method for interpolating between frequencies of n-grams of varying lengths (Kneser et al., 1995).

However, as both our models train on derivational pairs of words, there is the question of how to estimate the frequency of the pair. As explained in Subsection 2.1.7, the direction of derivation generally goes from the more common to the rarer word, but there are exceptions to this rule. Such exceptions may point to a higher degree of lexicalization of the derivational child, and therefore to lesser productivity of the process used to derive the child from its parent. It may be undesirable to increase the frequency of such examples.

3. Related work

In this chapter, we will first introduce derivational and segmentational data resources usable for training and evaluating our algorithms, and then related segmentation systems of other authors.

As the aim of this thesis is to segment Czech words, the focus of the first part of this chapter is to introduce the resources necessary for that, namely the DeriNet network (Žabokrtský et al., 2016) in Subsection 3.1.1, and the evaluation data set from the Retrograde morphemic dictionary of Czech language (Slavíčková, 1975) in Subsection 3.2.3.

However, as our algorithm is not specific to Czech and should have some degree of language independency, we also list foreign-language datasets that may be of use. As we haven't found another language with the necessary combination of large derivational data and segmentational data in a compatible format, the list is mostly meant as an overview of related work.

3.1 Derivational data

Since derivational data primarily expresses links between words, it typically comes in the form of graph databases with various levels of complexity. In the recent years, several projects aiming to gather such data were started for different languages. These projects differ in format, size, scope, complexity and completeness of their annotation, see Kyjánek (2018) for a list.

We can identify four major formats of the data (Kyjánek, 2018):

1. Directed trees. This is the format of the DeriNet database, which lists a single derivational parent for each motivated lexeme and no parent for unmotivated ones, thus forming a directed tree structure (Žabokrtský et al., 2016). This is the format most suitable for our algorithms.

2. General graphs. This format is used in DErivBase (Zeller et al., 2014), in which nodes are connected by edges without a restriction on the structure.

3. Family listing. This format, used in DErivCELEX (Shafaei et al., 2017) and DerivBase.hr (Šnajder, 2014) is technically not graph-based, because databases in this format merely groups derivationally related words from a single derivational family (what would be a single tree in DeriNet) into a flat list. Therefore, it is unsuitable for our purposes, as our algorithms require overt derivational links.

3.1.1 DeriNet

DeriNet is a network of derivational relations between Czech words, structured into directed trees (Žabokrtský et al., 2016). The newest version (as of June 2018), 1.5.1, contains 1 011 965 words connected with 785 525 derivational relations, it is therefore by far the largest derivational resource known to us (Kyjánek, 2018).

The database has a high quality of annotation. In version 1.0, it had a precision of 99% and recall of 88% as measured on the derivational relations (Vidra, 2015). The

recall has grown since then and the latest estimates are 99% for precision and 91% for recall.

The annotation is synchronic and attempts to be regular, i.e. whenever there is a choice of multiple parents, DeriNet tends to choose the one that follows the choices done for similar derivational pairs, even when another parent could be a more natural choice in the particular case. This means that e.g. etymological back-formation is systematically reversed. Such annotation is in line with e.g. the view of Mel'čuk (2006), who argues that back-formation is not a morphological process due to its inherent diachrony, while its reverse may be morphologically more sound (Mel'čuk, 2006, p. 310).

3.1.2 DErivBase

DErivBase (Zeller et al., 2014) is a database that covers 280 336 German words, out of which 65 420 words have some derivational links listed, and they are grouped into 20 371 families.

The families are not trees, but general graphs – a pair of nodes from a single connected subgraph may have more than one path connecting them, and there may be directed cycles. The edges are scored to indicate their probability, since they are autogenerated by a machine learning algorithm based on manually-specified rules.

It is theoretically possible to approximate derivational tree extraction from the database by using the supplied edge-wise quality scores with an algorithm for finding optimal spanning arborescences (the directed analog of the minimum spanning tree), e.g. the Chu-Liu-Edmonds' algorithm (Chu et al., 1965), but this was, to our knowledge, not attempted.

DErivBase.hr is a variant of DErivBase for Croatian (Šnajder, 2014), containing 100 000 words grouped into 56 000 families. It also does not contain trees, as it annotation consists of clustering of words into derivational families. The reported quality of the resource is quite low, with 81.2% precision and 76.5% recall.

3.1.3 Sloleks

Sloleks (Dobrovoljc et al., 2015) is a tree-structured database of Slovenian derivations, listing, in its version 1.2, 100 784 words with 65 951 derivational relations. There is a compatible lemmatizer available, Obeliks (Grčar et al., 2012), but no segmentation data are available for Slovene.

3.2 Segmentation data

Segmentation data also comes in different flavors, differing in the richness of their annotation. Some datasets annotate only the morph boundaries, other add morpheme information on top of that, some only annotate the morpheme composition without presenting the immediate morph segmentation.

3.2.1 Morpho Challenge datasets

The data most widely used by projects with aims similar to ours is part of the Morpho Challenge, an unsupervised morpheme analysis competition held in 2005 and

annually from 2007 to 2010 (Kurimo et al., 2010). The main task in the competition is morpheme analysis, not segmentation, but they also publish segmentation data for three languages: English, Finnish and Turkish, together with morpheme annotation.

3.2.2 Bázový morfematický slovník češtiny

Limited information about Czech segmentation is available in Šiška (1998): It lists selected root morphemes grouped by their semantics, putting root allomorphs together. For each such list of morphs, a list of exemplar lexemes is given. However, as the the book is intended for human audience, not for machine learning, these lists are short, exhaustive segmentations are not present and the dictionary is not machine tractable.

3.2.3 Retrograde morphemic dictionary of Czech language

The dictionary by Slavíčková (1975) is more relevant, as it directly lists segmentations of words into morphs and the segmentations given there correspond to the desired output of our tool. The dictionary is not available in machine-tractable electronic form, but the book was recently scanned by Jaroslava Hlaváčová (Slavíčková, 2018), who also rewrote the full complement of 14 581 verbs listed therein into a machine-readable text file (Slavíčková et al., 2017).

We have further processed this text file to make it compatible with the set of words used in DeriNet by:

- Removing the reflexive *se* found with some words. We do not consider it to be a part of the verb, but a standalone pronoun.
- Treating slash-markings the same way as standard segmentation point markings. These are used for several different purposes in the dictionary, mostly to mark and distinguish homonymous morphs with complementary distributions – if the same string may denote both an affix and root morph, the affix morph is marked with slashes at both sides.
- Removing the word-final *-i*, which is no longer used to mark infinitives in modern Czech and the vocabulary of DeriNet does not contain it.

For an excerpt of the dictionary, see Figure A.1 in Attachment A.1.

3.2.4 Hutmegs

Hutmegs (Creutz; Lindén, 2004) is a segmentational resource for Finnish and English. It lists segmentations for 1 400 000 Finnish words and 120 000 English ones. It is not freely available and a license must be purchased for non-academic use. This is one reason not to use it in our project, as it could influence the licensing of DeriNet. Excerpts from Hutmegs are available in the Morpho Challenge data. As we do not have a large enough Finnish nor English derivational resource, we did not attempt to obtain this dataset.

3.2.5 CELEX

The CELEX database (Baayen et al., 1995), available for English, German and Dutch, contains morphological segmentation data, but it doesn't list surface segmentations into morphs, only segmentations into derivational morphemes. This means that inflectional morphs are omitted altogether and derivational morphs are converted to a canonical form; e.g. *blockieren* is segmented as *Block·ier* and *blenden* as *blind*.

Also, for some words, the segmentational field is filled, but the segmentation is incomplete or the word is left unsegmented: e.g. *abgesagt* is listed as a single segment, although it can be (as far as my German skills can tell) segmented as *ab·ge·sag·t*.

It does list surface stem information, but that is for the standard, inflectional meaning of "stem": The word without its inflectional affixes. If there is no inflectional affix, the stem is equal to the lemma, even when there are derivational affixes left.

Therefore, is not usable as gold-standard segmentation data source for our models.

3.3 Corpora

We use the Morfessor FlatCat model (Grönroos et al., 2014) as a state-of-the-art comparison target for evaluation. This model is optionally trained on token frequencies (see Subsection 2.2.4 for an explanation) obtained from a corpus. We chose SYN2010 (Křen et al., 2010) as the corpus to train on. It is a corpus of contemporary written Czech, based in roughly equal amounts on broadsheet newspaper articles from the years 2005-2010, scientific literature and literary fiction. The corpus should be representative of the Czech language (Křen et al., 2010).

For training, we stripped punctuation and words containing non-letters from the corpus. This filtering was performed by keeping only words that satisfy the following Python3 regular expression: `^[^\W\d_]*$`.

The filtered corpus contains 8 182 869 sentences with 99 604 643 word tokens belonging to 1 580 543 word types. The number is higher than for DeriNet, because it counts inflected word forms, while DeriNet only contains lemmas. This discrepancy is intentional, as we want to test Morfessor both on lemmas from DeriNet and on inflected forms from a corpus and compare both scores.

3.4 Segmentation methods

Since our task is to create an automatic system for morphological segmentation, it is important to look at other segmentation systems. This section is meant to give an inexhaustive overview of the method types similar to our systems, to allow for a comparison with current and previous state of the art.

For a long time, the focus of the field was on unsupervised systems (Ruokolainen et al., 2016). The reasons usually given are cheap applicability to multiple languages, including underresourced ones, closeness to other similar problems such as word segmentation (tokenization of text or recorded speech into words) and independence from linguistic theories (Creutz; Lagus, 2002). The latter reason is, in fact, surprisingly important, because as we explain in Section 2.1, many theories are very vague on how to recognize morphemes and morphs.

3.4.1 Character successor frequency

An early unsupervised method of segmentation is based on character-based n-gram successor frequency (Harris, 1955; Déjean, 1998) or entropy (Hafer et al., 1974). The theory behind this approach is, that morphs are common clusters of characters. When sequentially scanning a word, the next character inside a cluster is easily predictable from the preceding context, while the first character of the next cluster is not, because the vocabulary of morphs is relatively limited, while the possible morph bigram combinations are numerous. Therefore, when the entropy (or nearly equivalently, the successor frequency) of the next character as predicted by an n-gram model raises above a certain absolute or relative threshold, or peaks, a morph boundary separates the next character from its predecessor. This technique is also used in the more recent system Affisix (Hrušecký et al., 2010).

3.4.2 Minimum description length models

Several newer methods are based on the minimum description length principle (Rissanen, 1996; Goldsmith, 2001), which states that the best description (the best model) of some data is the description that best compresses the data. The principle balances exhaustiveness with complexity, minimizing the sum of description size and residual unexplained data size – i.e. the principle minimizes the number of bits needed to store the trained model and the training corpus processed by the model. In this way, the minimum description length principle is related to Occam's razor or Maximum entropy models.

One popular unsupervised system using this principle is Morfessor (Creutz; Lagus, 2005) and its derivatives, which used it directly in its first implementation, Morfessor Baseline (Creutz; Lagus, 2002). The algorithm works by taking input words one by one, trying different splits of the word into two segments and selecting the one with the minimum cost (the cost being measured as the complexity of the morph lexicon plus of the recorded decisions splits), or leaving the word unsegmented if the division would increase the total description length. If a split was made, the two segments are then segmented recursively using the same algorithm.

Since this incremental model could produce suboptimal segmentations for words seen early in the training phase, when the morph lexicon is nearly empty, a word is resegmented each time it is encountered, even if it was previously seen, and the whole dictionary is reread periodically.

An updated version of the Morfessor algorithm, called Morfessor FlatCat (Grönroos et al., 2014), is a modern state-of-the-art segmentation method and as such, we will use it as the target to compare our system against. It uses semi-supervised learning (e.g. utilizes a small amount of hand-annotated data in addition to an unannotated corpus) to return a segmentation closer to the linguistically appropriate one. Over the Morfessor Baseline system, it categorizes morphs into prefixes, stems and suffixes and adds an n-gram *morphotactics model* for a better estimation of which morphemes may follow which ones.

3.4.3 Bayesian inference

Another popular principle for unsupervised segmentation systems is Bayesian inference. Although other methods also use some principles of statistical inference, in-

cluding our EM model and the Minimum description length models described above, there is a group of systems based directly on Bayesian inference and the associated set of algorithms, such as Gibbs sampling (Geman et al., 1984). These methods work by looking at the data D as on a set of statistical variables with the observable values (e.g. the values of individual characters) fixed to the values of the individual data items, specifying a generative probabilistic model of the data (including the hidden variables H, e.g. the segmentation points), and then estimating the hidden variables by repeatedly sampling individual data points from the model, taking the current setting of all other variables in the model as fixed. The sampling is made possible through decomposition using the Bayes rule as follows, making the posterior $\mathsf{p}(H \mid D)$ easier to compute using the easily-computed (if taking one variable at a time) likelihood $\mathsf{p}(D \mid H)$ and estimatable prior $\mathsf{p}(H)$:

$$\mathsf{p}(H \mid D) = \frac{\mathsf{p}(D \mid H)\,\mathsf{p}(H)}{\sum_{H'} \mathsf{p}(D \mid H')\,\mathsf{p}(H')} \propto \mathsf{p}(D \mid H)\,\mathsf{p}(H) \tag{3.1}$$

Because the variables are sampled one at a time, the individual successive samples are not independent of one another, but this issue can be sidestepped by only recording every n^{th} sample (e.g. for $n = 1000$), not remembering samples in between, decreasing the interference between successive recorded samples.

Models in this class include newer versions of Morfessor (since one formulation of its Minimum description length model has been proven to be equivalent to a Bayesian Maximum a posteriori model (Creutz; Lagus, 2005)) and models of Goldwater et al. (2005), Kurfalı et al. (2017) and Can et al. (2018).

3.4.4 MorphoChains

The method which is perhaps the closest to our systems is the MorphoChains model (Narasimhan et al., 2015) and its derivatives (Bergmanis et al., 2017). The morphological chains used by these models are approximations of derivational relations discovered by an unsupervised, orthography-based model. Each chain is an ordered list of morphologically related words which starts from the one with the fewest morphemes, adding affixes or compound parts at each step. One word may belong to multiple chains, effectively creating graphs which should be similar to the trees found in DeriNet, since there is usually at most one linguistically plausible preceding word in the chain.

Narasimhan et al. (2015) build the chains using several features, including the affix change from the preceding word (including some model of MOP alternations) and semantic similarity measured by word embedding vectors (Mikolov et al., 2013). An example chain given in Narasimhan et al. (2015) is *nation* ⟶ *national* ⟶ *international* ⟶ *internationally*, which shows very large similarity with our derivational trees. The individual chain steps need not be substrings of one another, as the model can predict more complex changes including reduplication and other alternations.

The papers do not contain an analysis of the predicted chains, so it is hard to say how they compare with a manually annotated derivational network such as DeriNet. Given the experience of Lango et al. (2018), who attempted semi-supervised construction of derivational networks for Polish and Spanish, we know this task is possible, but the quality of the chains will probably not be as high as the quality of manually annotated data. On the other hand, the general graph formulation of the chains

Narasimhan et al. (2015) use is more general than the directed trees used in DeriNet, making it possible for them to cover phenomena not captured in DeriNet, such as compounding. They report that the model "tends to prefer a single parent for every word", but it is hard to tell what exactly "tends to" means and to what extent the model learns to identify compounds.

A similar but simpler system was created by Üstün et al. (2016), who also use word embeddings to find semantically similar parents, but do away with the complex chains and only look for possible parents among substrings of the word.

4. EM-based segmentation model

In this chapter, we will introduce three algorithms for stemming Czech words and an algorithm for morphological segmentation that uses stemming and derivational links to segment words into morphemes.

The theory from Section 2.1 tells us that a pair of Czech words connected by a derivational link should usually[1] share a common stem and only differ in its affixes.

Since we have a database of such derivational links, we can use it to look for the common parts of a word pair, which we will consider to be a stem of the derivational child, and the differing parts, which we will consider to be (potentially multi-morph) affixes. Three concrete algorithms for such stemming, one simple and two more advanced, are explained below.

4.1 Common substring stemming algorithm

The simplest algorithm for identifying a stem is string equality comparison. We can find the longest common contiguous substring of a derivational pair without attempting to identify or model any morphological, orthographical and phonological (MOP) alternations occuring inside the morphs or at their boundaries. Such algorithm is obviously deficient, but since alternations are relatively rare, it should identify the correct stem in the majority of cases. See Figure 4.1 for examples of correct and incorrect stemming using this technique.

4.2 Stemming using a MOP alternation model

Because of the deficiencies of the previous algorithm in handling morphological, orthographical and phonological alternations, we have devised a more complex algorithm to cover them. It is a probabilistic algorithm that explicitly models derivation by modelling both the process of affixation and of stem alternations. It therefore models the fact that the morphological composition of a motivated word is strongly dependent on the composition of its derivational parent.

[1] Suppletion is an uncommon exception.

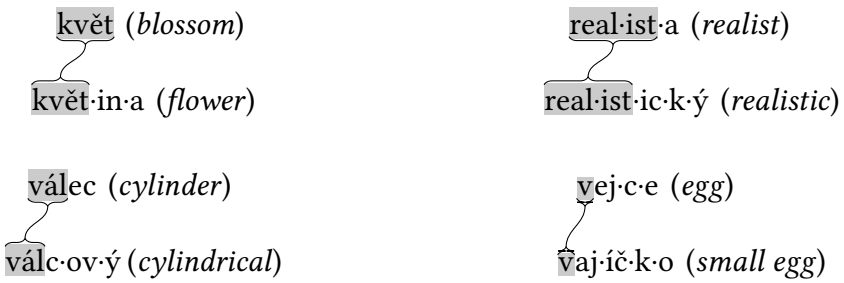


Figure 4.1: Four examples of stemming using the common substring algorithm, the top two being correct, the bottom two incorrect. The segmentation point marks (·) indicate the gold standard segmentation. The grey background and braces indicate the common (mapped) stems between the two words.

The affixation and alternation processes are, as a simplification, considered to be independent of each other. The selection of prefixes and suffixes is, however, modelled jointly, thus explicitly modelling complex processes such as circumfixation.

We can therefore model stemming as a segmentation of a word w into its stem and affixes as follows:

$$\begin{aligned}
&\text{segmentation of } w = r \in \Sigma^+;\ p, s \in \Sigma^* : \\
&\qquad (prs = w \wedge p \text{ is a prefix} \wedge r \text{ is a stem} \wedge s \text{ is a suffix}) && (4.1)\\
&\qquad = \operatorname*{argmax}_{p,r,s:\ prs=w} \mathsf{p}(w \text{ has prefix } p \wedge w \text{ has stem } r \wedge w \text{ has suffix } s) && (4.2)\\
&\qquad = \operatorname*{argmax}_{p,r,s:\ prs=w} \mathsf{p}(w \text{ has prefix } p \wedge w \text{ has suffix } s) \cdot \mathsf{p}(w \text{ has stem } r) && (4.3)
\end{aligned}$$

The individual models for affixes and stems are described in the next two subsections.

4.2.1 Modelling affixation

Affixation is modelled differently for motivated and unmotivated words. Unmotivated words are considered to be unaffixed and their stem-internal morphemic structure is discovered by secondary resegmentation only.

For motivated words, affixation is modelled using joint probabilities of the affixes of the child and the residual affixes of the parent, which can be derived from affix probabilities of the child using the law of total probability (LTP).

LTP states that given a countable set of measurable events $B_i : i \in \{1, 2, \ldots\}$, that form a partition of a sample space, the following equation holds for any event A: $\mathsf{p}(A) = \sum_i \mathsf{p}(A \cap B_i)$. In other words, the probability of an event can be measured piecewise, as long as we make sure to count all the pieces and not count any piece more than once.

Using LTP, the following decomposition is possible:

$$\begin{aligned}
&\forall w \in \Sigma^+, w \text{ has a parent}, w' = \text{parent}(w), \forall p \in \text{prefixes}(w), \forall s \in \text{suffixes}(w) : \\
&\qquad \mathsf{p}(w \text{ has prefix } p \wedge w \text{ has suffix } s) \overset{\text{LTP}}{=} \sum_{\substack{p' \in \text{prefixes}(w') \\ s' \in \text{suffixes}(w')}} \mathsf{p}\begin{pmatrix} w \text{ has prefix } p \wedge w \text{ has suffix } s \wedge \\ w' \text{ has prefix } p' \wedge w' \text{ has suffix } s' \end{pmatrix} && (4.4)\\
&\qquad\qquad = \sum_{\substack{p' \in \text{prefixes}(w') \\ s' \in \text{suffixes}(w')}} \mathsf{p}((p', s') \to (p, s)) && (4.5)
\end{aligned}$$

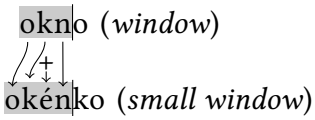


Figure 4.2: An example of a stem mapping. The gray areas contain the characters which are mapped from the derivational parent's mapped stem (top) to its child's stem (bottom). The *-o* and *-ko* are suffixes of the parent and child, respectively, and thus play no role in the mapping. Arrows from letter to letter indicate substitutions, the plus sign indicates insertion of the vowel *é*.

4.2.2 Modelling MOP alternations

For stems, an equation analogous to Equation (4.5) holds:

$$\forall w \in \Sigma^+, w \text{ has a parent}, w' = \text{parent}(w), \forall r \in \text{substrings}(w), |r| > 0:$$

$$\mathsf{p}(w \text{ has stem } r) \overset{\text{LTP}}{=} \sum_{\substack{r' \in \text{substrings}(w') \\ |r'|>0}} \mathsf{p}(w \text{ has stem } r \wedge w' \text{ has mapped stem } r') \tag{4.6}$$

$$= \sum_{\substack{r' \in \text{substrings}(w') \\ |r'|>0}} \mathsf{p}(r' \to r) \tag{4.7}$$

But it is necessary to decompose the probabilities further, because the value of this formula cannot be directly estimated in a real-world segmentation system due to data sparsity. Affixes are broadly shared by many different words, making their counts in the data reliable indicators of their probability, but a particular stem is seen at most a couple of times – in fact, many stems are only found in a single word. Due to this, we will estimate the stem probability using a character-wise string mapping algorithm from the parent mapped stem to the child stem.

The algorithm works by scoring pairwise alignments between the mapped stem of the parent and the stem of the child. An alignment between two strings $s = (s_1, s_2, \ldots, s_n)$ and $t = (t_1, t_2, \ldots, t_m)$ is a sequence $c_1, c_2, \ldots, c_k$ of single-character changes, where each change is either a substitution $s_i \to t_j$, insertion $\lambda \to t_j$ or deletion $s_i \to \lambda$, and the indices i and j in the list of changes form contiguous sequences $1 \ldots n$ and $1 \ldots m$ respectively (i.e. transpositions are not allowed). See Figure 4.2 for an example of such stem mapping. Additionally, it is not allowed for an insertion to immediately (without an interleaving substitution) follow a deletion, since the alignments $(x \to \lambda, \lambda \to y)$ and $(\lambda \to y, x \to \lambda)$ represent an identical process and this duplication would make it harder to enumerate the different unique aligments. By fixing the canonical order as $(\lambda \to y, x \to \lambda)$, each possible stem mapping process has a single unique alignment.

How many alignments are there between strings s and t? If we consider a model with only insertions and deletions, without substitutions, there is only one – insert all, then delete all. If we introduce substitutions, the count can be decomposed as follows: First, we choose $k \in \mathbb{N}$ indicating how many substitutions there will be in the alignment – there are $\min(|s|, |t|) + 1$ possible choices. Second, we choose k characters from s and k characters from t to be substituted. This fully determines which gets substituted for which, because they may only be mapped in order, as there is no way our algorithm can produce a transposition. The order of (optional) insertions

and deletions between each pair of substitutions is also determined by the substitutions and there is only one way of placing them (see the model without substitutions above). The final equation for the alignment count is:

$$|\text{alignments}(s,t)| = \sum_{k=0}^{\min(|s|,|t|)} \binom{|s|}{k} \cdot \binom{|t|}{k} \tag{4.8}$$

Since different alignments are mutually exclusive (the stem and its corresponding mapped stem cannot be aligned in more than one way), we can calculate the probability of a mapping between the mapped stem and stem by summing the probabilities of all possible alignments. The probability of a word's stem candidate is therefore:

$$\begin{gathered}\forall w \in \Sigma^+, w \text{ has a parent}, w' = \text{parent}(w), \forall r \in \text{substrings}(w), |r| > 0: \\ \mathsf{p}(w \text{ has stem } (t_1, t_2, \ldots, t_m)) \overset{\text{LTP}}{=} \sum_{\substack{r' \in \text{substrings}(w') \\ |r'|>0 \\ r'=(s_1,s_2,\ldots,s_n)}} \mathsf{p}((s_1, s_2, \ldots, s_n) \text{ maps to } (t_1, t_2, \ldots, t_m))\end{gathered} \tag{4.9}$$

An alignment is scored by calculating its probability, which we define as a product of the probabilities of the individual changes, i.e. we simplify calculations by assuming independence between the characters in the stem.

$$\mathsf{p}(c_1, c_2, \ldots, c_k) = \mathsf{p}(c_1 \cap c_2 \cap \ldots \cap c_k) \tag{4.10}$$

$$= \mathsf{p}(c_1) \cdot \mathsf{p}(c_2) \cdot \ldots \cdot \mathsf{p}(c_k) \tag{4.11}$$

$$= \prod_{i \in \{1 \ldots k\}} \mathsf{p}(c_i) \tag{4.12}$$

The final probability is given by the following recurrent algorithm obtained by repeatedly taking all the possible first changes out of the sum and product. There are two equations, one for the case where an insertion can follow, the other (p_D) for the case where a deletion has occured and thus no insertions are allowed up to the nearest substitution.

$$\begin{aligned}&\mathsf{p}_D((s_1, s_2, \ldots, s_n) \text{ maps to } (t_1, t_2, \ldots, t_m)) \\ &\quad = \mathsf{p}(s_1 \to t_1) \cdot \sum_{(c_2,\ldots,c_k) \text{ are changes of } (s_2,\ldots,s_n) \text{ to } (t_2,\ldots,t_m)} \prod_{i \in \{2 \ldots k\}} \mathsf{p}(c_i) \\ &\quad + \mathsf{p}(s_1 \to \lambda) \cdot \sum_{(c_2,\ldots,c_k) \text{ are changes of } (s_2,\ldots,s_n) \text{ to } (t_1,t_2,\ldots,t_m)} \prod_{i \in \{2 \ldots k\}} \mathsf{p}_D(c_i)\end{aligned} \tag{4.13}$$

$$\begin{aligned}&= \mathsf{p}(s_1 \to t_1) \cdot \mathsf{p}\left((s_2, \ldots, s_n) \text{ maps to } (t_2, \ldots, t_m)\right) \\ &\quad + \mathsf{p}(s_1 \to \lambda) \cdot \mathsf{p}_D\left((s_2, \ldots, s_n) \text{ maps to } (t_1, t_2, \ldots, t_m)\right)\end{aligned} \tag{4.14}$$

$$\begin{aligned}
&\mathsf{p}((s_1, s_2, \ldots, s_n) \text{ maps to } (t_1, t_2, \ldots, t_m)) \\
&\qquad = \sum_{(c_1,c_2,\ldots,c_k) \text{ are changes of } (s_1,s_2,\ldots,s_n) \text{ to } (t_1,t_2,\ldots,t_m)} \prod_{i \in \{1\ldots k\}} \mathsf{p}(c_i) \qquad (4.15) \\
&= \mathsf{p}(s_1 \to t_1) \cdot \sum_{(c_2,\ldots,c_k) \text{ are changes of } (s_2,\ldots,s_n) \text{ to } (t_2,\ldots,t_m)} \prod_{i \in \{2\ldots k\}} \mathsf{p}(c_i) \\
&\quad + \mathsf{p}(\lambda \to t_1) \cdot \sum_{(c_2,\ldots,c_k) \text{ are changes of } (s_1,s_2,\ldots,s_n) \text{ to } (t_2,\ldots,t_m)} \prod_{i \in \{2\ldots k\}} \mathsf{p}(c_i) \\
&\quad + \mathsf{p}(s_1 \to \lambda) \cdot \sum_{(c_2,\ldots,c_k) \text{ are changes of } (s_2,\ldots,s_n) \text{ to } (t_1,t_2,\ldots,t_m)} \prod_{i \in \{2\ldots k\}} \mathsf{p}_D(c_i) \qquad (4.16) \\
&= \mathsf{p}(s_1 \to t_1) \cdot \mathsf{p}\left((s_2, \ldots, s_n) \text{ maps to } (t_2, \ldots, t_m)\right) \\
&\quad + \mathsf{p}(\lambda \to t_1) \cdot \mathsf{p}\left((s_1, s_2, \ldots, s_n) \text{ maps to } (t_2, \ldots, t_m)\right) \\
&\quad + \mathsf{p}(s_1 \to \lambda) \cdot \mathsf{p}_D\left((s_2, \ldots, s_n) \text{ maps to } (t_1, t_2, \ldots, t_m)\right) \qquad (4.17)
\end{aligned}$$

With the recursion stopping conditions for when either or both strings are exhausted defined as follows:

$$\mathsf{p}((s_x, s_{x+1} \ldots, s_y) \text{ maps to } ()) = \prod_{i \in \{x\ldots y\}} \mathsf{p}(s_i \to \lambda) \qquad (4.18)$$

$$\mathsf{p}(() \text{ maps to } (t_x, t_{x+1} \ldots, t_y)) = \prod_{i \in \{x\ldots y\}} \mathsf{p}(\lambda \to t_i) \qquad (4.19)$$

$$\mathsf{p}(() \text{ maps to } ()) = 1 \qquad (4.20)$$

$$\mathsf{p}_D((s_x, s_{x+1} \ldots, s_y) \text{ maps to } ()) = \prod_{i \in \{x\ldots y\}} \mathsf{p}(s_i \to \lambda) \qquad (4.21)$$

$$\mathsf{p}_D(() \text{ maps to } (t_x, t_{x+1} \ldots, t_y)) = 0 \qquad (4.22)$$

$$\mathsf{p}_D(() \text{ maps to } ()) = 1 \qquad (4.23)$$

4.2.3 Joint modelling of affixation and alternation

Since both models defined above use the law of total probability to introduce a dependency on the derivational parent, it is possible to model the parent's segmentation jointly in the combined model. This gets rid of some of the independence assumptions made in Equation (4.3) and prevents each part of the model from segmenting the parent in a different way. The combined model therefore is:

$$\begin{aligned}
&\forall w \in \Sigma^+, w \text{ has a parent}, w' = \text{parent}(w): \\
&\text{segmentation of } w = \underset{\substack{r \in \Sigma^+; \\ p,s \in \Sigma^*: \\ prs = w}}{\text{argmax}} \mathsf{p}(w \text{ has prefix } p \wedge w \text{ has stem } r \wedge w \text{ has suffix } s) \\
&\qquad = \underset{\substack{r \in \Sigma^+; \\ p,s \in \Sigma^*: \\ prs = w}}{\text{argmax}} \sum_{\substack{r' \in \Sigma^+; \\ p',s' \in \Sigma^*: \\ p'r's' = w'}} \mathsf{p}((p', s') \to (p, s)) \cdot \mathsf{p}(r' \to r) \qquad (4.24)
\end{aligned}$$

4.3 Model inferrence algorithms

After designing our model in the previous section, we have to train it by finding the best configuration of its parameters, i.e. estimating the probabilities of affixes and MOP alternations.

In Subsection 4.3.1, we will define how to estimate model parameters from annotated data. Since we do not have such data for supervised learning, an Expectation-Maximization loop described in Subsection 4.3.2 will be used to bootstrap our model without requiring it.

4.3.1 Estimating maximally likely parameters

Let's assume that we have annotated data, which lists the following information:

- For each derivational pair, all possible (stringwise, not linguistic) segmentations of the child into its potential stem and affix combinations.
- For each such segmentation, all possible alignments of the stem and affixes to the corresponding substrings of the parent word.
- For each such alignment of the stem, the per-character mapping of the stem to the parent's mapped stem.
- For each such affix alignment and stem mapping, the probability that the derivational pair is segmented and has stems mapped this way.

We assume that for each derivational pair, there is exactly one correct segmentation, therefore all the probabilities of all alignments and mappings of any given derivational pair should sum up to 1. Our model doesn't guarantee this, requiring normalization when predicting by dividing the probability of each alignment and mapping by the sum of probabilities of all alignments and mappings present for that derivational pair.

We can then estimate the parameters of our model by accumulating the individual probabilities from these annotated data by using the maximally likely estimate. If $\mathsf{c}(s \to t)$ denotes the sum of word segmentation probabilities of alignments in which each occurence of the $s \to t$ MOP alternation was found, the probability of an alternation is estimated by normalizing this sum by the total sum of probabilities of all possible alternations in the data as follows:

$$\forall s, t \in (\Sigma \cup \{\lambda\}) \colon \mathsf{p}(s \to t) = \frac{\mathsf{c}(s \to t)}{\sum_{u,v \in (\Sigma \cup \{\lambda\})} \mathsf{c}(u \to v)} \tag{4.25}$$

Similarly, if $\mathsf{c}(p', s' \to p, s)$ denotes the sum of word segmentation probabilities of alignments which contain the $p', s' \to p, s$ affix change (e.g. alignments where the parent has prefix p' and suffix s' and the child has prefix p and suffix s), the probability of an affix change can be estimated by normalizing over all possible affix changes:

$$\forall p', s', p, s \in \Sigma^+ \colon \mathsf{p}((p', s') \to (p, s)) = \frac{\mathsf{c}((p', s') \to (p, s))}{\sum_{u', v', u, v \in \Sigma^+} \mathsf{c}((u', v') \to (u, v))} \tag{4.26}$$

In the real model, we use Laplace smoothing with a smoothing term α[2] tunable by a hyperparameter of the model to improve training stability in the early iterations. Smoothing redistributes probability mass from the more probable cases to the less probable and helps to avoid degenerate cases where some probability collapses to 0. This requires us to only process changes which were already encountered in the data (the sets "mop-changes" and "affix-changes") instead of all possible strings, since otherwise the denominator of the affix change equation would not be finite for $\alpha > 0$. Therefore, the model with smoothing looks as follows:

$$\forall s, t \in \text{mop-changes}\colon\ \mathsf{p}(s \to t) = \frac{\mathsf{c}(s \to t) + \alpha}{\sum_{u,v \in \text{mop-changes}} (\mathsf{c}(u \to v) + \alpha) + \alpha} \tag{4.27}$$

$$\forall p', s', p, s \in \text{affix-changes}\colon\ \mathsf{p}((p', s') \to (p, s)) = \frac{\mathsf{c}((p', s') \to (p, s)) + \alpha}{\sum_{u',v',u,v \in \text{affix-changes}} (\mathsf{c}((u', v') \to (u, v)) + \alpha) + \alpha} \tag{4.28}$$

Notice that the stem mapping counts are calculated over individual mappings, while affix change counts are calculated over affix alignments to avoid giving greater weight to affixes of longer words.

4.3.2 Expectation-Maximization inferrence loop

If we had manually annotated gold-standard data, we could calculate the maximally likely estimates of these probabilities directly, using frequencies of occurences in the data. However, we do not have such data available, as even segmentation datasets such as the one by Slavíčková et al. (2017) do not list stems and their alignment to the mapped stem of the derivational parent, only full segmentations with no derivational relations.

Due to the lack of data, we train the algorithm by an unsupervised Expectation-Maximization (EM) loop (Dempster et al., 1977) on derivational data only, without using the gold-standard segmentation at all. EM is a well known technique for bootstrapping a full model from a trivial one. It works by alternating an Expectation phase, where an existing model (a trivial one in the first iteration, the one from the last iteration in subsequent ones) is used to annotate a dataset, with a Maximization phase, in which the annotated data is used to infer the parameters of a new, hopefully better, model. In our case, this estimation is done by maximum likelihood as per the previous subsection. The phases are alternated until the model converges, i.e. stops changing.

Of course, such an arrangement only works under specific circumstances. The model can easily converge to a local optimum, or even start diverging and worsening its performance. In our case, divergence is common and had to be avoided through careful model reformulation. Divergence happens, because there are two driving forces – the stem mapping and the affixes – which have to be well-balanced at each iteration. If they are not, the model converges on a wrong, degenerate solution determined by the stronger force.

If the stem mapping probabilities are too low, the model will compensate by selecting longer affixes and shortening the stems to increase the overall word proba-

[2] Often denoted by λ, but we chose α to differentiate it from the empty string.

bility, thus converging to a model equivalent to the Common substring stemming algorithm from Section 4.1, or, in case the probabilities are really extreme, to a model where each stem is just a single character and the rest of the word is considered an affix.

If, on the other hand, the stem mapping probabilities are too high, the model will be able to select shorter (thus presumably with more occurences and more probable) affixes, eventually converging to a state where the whole word is considered to be a stem and the affixes are empty.

4.4 Alternative model

A problematic part of this model is the competition between a substitution from s to t and the equivalent pair of a deletion of s and an insertion of t. If we calculate the probabilities as joint probabilities, the deletions and insertions will overpower the substitutions, because substitutions are normalized over both s and t, while deletions and insertions are only normalized over one of them, the other character is λ – and a λ is effectively found between any pair of characters in the data, so a deletion or insertion is seen much more often. Consequently, the affix probabilities will be off, because the model will favor shortening the stem and deleting the superfluous characters over using a longer stem with a change in it.

We attempted to fix this shortcoming by introducing a single-character context into the deletions and insertions, i.e. modelling them as "delete character s_i from the parent if the next character of the child is t_j" and "insert character t_j to the child if the next character in the parent is s_i", but this model suffered from data sparsity, as the deletions and insertions are quite rare and usually occur just before or after a morpheme boundary, where (in line with the observations of Harris, 1955) there are many possible successor or predecessor characters. Basing them on the previous character instead of the next one improves the quality only a bit.

We finally solved this problem by modifying the probabilistic formulation of the model. Instead of modelling substitutions using joint probabilities $\mathsf{p}(s, t)$, we model them as conditional probabilities $\mathsf{p}(t \mid s)$. Insertions and deletions are still modelled the old way. The only change required to implement this new model is in the maximum likelihood estimation, where the estimation of substitution probability is calculated as follows:

$$\forall s, t \in \text{mop-changes}\colon\ \mathsf{p}(s \to t) = \frac{\mathsf{c}(s \to t) + \alpha}{\sum_{v \in \text{mop-changes}} (\mathsf{c}(s \to v) + \alpha) + \alpha} \tag{4.29}$$

A possible interpretation of this model is, that we calculate $\mathsf{p}(s, t)$ as $\mathsf{p}(t \mid s) \cdot \mathsf{p}(s)$, with $\mathsf{p}(s) = 1 \forall s$. But ultimately, there is not mathematical reasoning or probabilistic interpretation behind this model, it was simply empirically found to perform better than the original one.

4.5 Implementation

We have implemented the algorithm in Python3, the code is available from `https://github.com/vidraj/segmentace`.

Several implementation tricks were necessary to obtain a realistically short run time and reasonably small memory profile. These are described in this section.

4.5.1 Limiting MOP alternation search space

As the size of the search space of stem mappings is exponential to the lengths of the stems, it is necessary to search only the reasonably viable areas. Similar classes of string matching algorithms, such as Levenshtein distance calculation (Levenshtein, 1966), or sequence alignment algorithms such as the Needleman-Wunsch algorithm (Needleman et al., 1970), Smith-Waterman algorithm (Smith et al., 1981), Dynamic time warping techniques (Sakoe et al., 1978) or Viterbi decoding of pair hidden Markov models (Viterbi, 1967) avoid this high cost, because they only calculate the best result. We have to score all possible alignments in order to update the count tables during the Expectation phase of the EM loop.

We have, however, experimentally confirmed that the probability distribution among the alignments is far from uniform. Most of the probability mass is concentrated on the several best alignments and the probabilities fall off quickly even with untrained or highly smoothed probability tables. This is because the best alignments are those that prefer substitutions over a pair of insertion and deletion, and even if the probabilities of those are close to uniform, a pair of insertion and deletion introduces one extra multiplicand strictly less than one (in fact, on the order of $\frac{1}{|\Sigma|}$) over the equivalent substitution, lowering the overall probability.

Empirically, the 100^{th} best alignment's score is often already on the order of 10^{10} lower than the 1^{st} best one. Such low scores can't meaningfully impact the outcomes, since the data is only 10^6 items large. The 1000^{th} best alignment's score is effectively zero.

We implemented a beam search algorithm which discards the lowest scoring hypotheses during computation. When filling in the transition table during stem mapping, at each step, only the top scoring k hypotheses are kept. The top scoring k hypotheses selected at the end of the procedure are guaranteed to be identical to the top scoring k hypotheses produced by an unrestricted search by analogy with the Viterbi HMM decoding algorithm (Viterbi, 1967), to which is our algorithm analogous when $k = 1$.

We have empirically observed that on a subsampled DeriNet dataset, the algorithm with a beam width of 100 converges to a result nearly identical to an unrestricted algorithm. Any difference seems to be due to rounding errors on floating-point numbers. The processing speed increased from approximately 5 words per minute to approximately 750 words per minute, as measured on an Intel Xeon E5-2690 processor running at 2.90GHz.

There are other possibilities of speeding the algorithm up: The simplest would be to only calculate alignments that are mostly 1:1, without many deletions and insertions, i.e. only calculate alignments in a band near the diagonal of the scoring matrix, similarly to the Dynamic time warping optimization of Sakoe et al. (1978). This method, however, can not be guaranteed to be globally optimal for all edge cases and since we also need to compare string of widely unequal lengths, the window size would have to be dynamic, otherwise it could happen that no path exists due to the window being too small for the step size.

Other optimizations used with Dynamic time warping techniques could also be

applicable – see Salvador et al. (2007) for an overview. In our case, the beam search sped the program up enough.

4.5.2 Joint Expectation and Maximization

The EM loop has two distinct phases: Expectation, where a model is applied on a dataset to obtain tagged data, and Maximization, where the tagged data is used to estimate new model parameters. Normally, these would be implemented as two distinct phases in the code as well.

In our case, this is not feasible. Even with the reduction of alternation search space from exponential to $\mathcal{O}(k^2)$ for a single stem, the algorithm would still require $\Omega(\sum_{w \in \mathbf{W}} |w|^3 |\text{parent}(w)|^3)$ memory to store the intermediate results. The algorithm can be made to run without globally storing any segmentational info outside the count tables by counting the results as soon as a hypothesis is created. Local intermediate storage is still needed, because the word segmentation probabilities have to be normalized to 1 before updating the count tables.

4.5.3 Affix probabilities preinitialization

Another necessary optimization to lower memory requirements is limiting the affix table size. If the algorithm ran unguided, the affix tables would gradually fill up with all combinations of string prefixes and affixes of every derivational pair, exhausting all available memory, as there is $\sum_{w \in \mathbf{W}} |w|^2 |\text{parent}(w)|^2$ many of them[3].

We do this by exploiting the Common substring stemming algorithm given in Section 4.1. There are two reasons why we hope this is an adequate solution:

1. We expect that every linguistically plausible affix combination occurs at least once without any MOP alternation.

2. The algorithm wouldn't be able to learn the affixes undiscovered by the Common substring algorithm anyway.

The first expectation may not be entirely correct, mostly because of backformation. For example, the pairs *anektovat* ⟶ *anexe* and *reflektovat* ⟶ *reflexe* require an affix pair of *-ovat* ⟶ *-e*, which is not found anywhere without an MOP alternation – there are similar pairs, e.g. *hypertrofovat* ⟶ *hypertrofie*, *transmitovat* ⟶ *transmise* and many different examples like *insinuovat* ⟶ *insinuace*, *abstrahovat* ⟶ *abstrakce* or *expedovat* ⟶ *expedice*, but the pair *-ovat* ⟶ *-e* will not be recognized in any of them by the Common substring algorithm.

The second expectation was not rejected by our experiments on small hand-crafted datasets. As such, these results may not be entirely representative of the algorithm's behavior on large data, but the hypothesis is difficult to test otherwise.

By preinitializing the affix tables with affixes found by the Common substring algorithm, disabling smoothing of affix counts and not recording zero-count updates in the probability tables, we can avoid recording information about most implausible affixes.

[3]Back of the envelope calculation: If we assume 8 characters per word on average, there are approximately $10^6 \cdot 8^2 \cdot 8^2$ substrings, each pointed to by an 8B pointer. Just these pointers will occupy over 30 GiB of space.

When using the preinitialization, it is necessary to also pretrain the MOP alternation probabilities – otherwise the untrained MOP probabilities would drag the affix probabilities far away from their pretrained values, partially offsetting the effect of pretraining. To do this, we perform several iterations over the data, updating only the MOP probability tables while holding the affix ones fixed.

A side effect of this preinitializationis speeding up the convergence rate. Testing on a mid-size dataset subsampled from DeriNet indicates that the algorithm needs 20 iterations to converge without initialization, but only 5 iterations for full convergence after preinitializing the tables.

4.5.4 Handling inflected words

Since we have both a derivational database and segmented data for Finnish, we wanted to evaluate our algorithm on that language. To do that, we needed to process inflected word forms, since the segmented data from MorphoChallenge is not composed of lemmas only. We have therefore added the ability to use the lemmatizers MorphoDiTa (Straková et al., 2014) for Czech and UDPipe (Straka et al., 2017) for all languages from the Universal Dependencies set of corpora (McDonald et al., 2013).

When using the lemmatizer, the input is interpreted as sentences in the selected format and is first lemmatized. Both the form and the output lemma (or lemmas in case of MorphoDiTa, which is able to give multiple candidates) are searched for in the database of segmentations. If the form is found, its segmentation is given as-is. If the form is not found, but the lemma is, the difference between the form and its lemma is analyzed first, segmentation of the lemma is transferred to the form using the secondary resegmentation algorithm, and the output is produced based on this segmentation.

5. Neural networks

Artificial neural networks (NNs) are, as the name suggests, collections of interconnected artificial neurons. They can be, among other uses, used as a supervised machine learning model for approximating unknown real-valued functions.

In this thesis, we construct a neural network for joint prediction of derivational relations and morphemic structure of words and discuss its performance in comparison to an equivalent network trained to solve either the former or the latter task only. The basics of neural networks and their training procedures are explained below and the structure of our neural-network-based predictor is described in Section 5.9.

Although NNs were conceived as an analogue of the biological neural tissue (McCulloch et al., 1943), they are defined as mathematical models and can be explained without referencing the complex field of neurology. Because of this, we will omit the modifier *artificial* in the following sections and the rest of this thesis, as we will not talk about animal neural systems at all.

We will start the explanation of artificial neural network's function by introducing the artificial neuron.

5.1 Basic neurons

A basic neural cell is a unit with n inputs $x_1, ..., x_n \in \mathbb{R}$, $n + 1$ internal (hidden) variables $w_0, ..., w_n \in \mathbb{R}$ defining its response profile, an *activation function* $f \in \mathbb{R} \to \mathbb{R}$, and a single output $h \in \mathbb{R}$. The output is a sum of the inputs weighted by the hidden variables $w_1, ..., w_n$, to which a biasing term w_0 is added and f is applied. The cell therefore computes the following function:

$$h = f\left(w_0 + \sum_{i=1}^{n} w_i \cdot x_i\right) \tag{5.1}$$

If we define x as $(1, x_1, ..., x_n)$ and w as $(w_0, ..., w_n)$, we can do without an explicit biasing operation and the output of a neuron becomes simply

$$h = f(w \cdot x) \tag{5.2}$$

In practice, many different functions are used as the activation function and the chosen one may be determined through experimentation or simply picked arbitrarily. The main restrictions are that the function should be continuous and non-linear to take advantage of multiple successive neurons (see the next section for the basic reason and Hornik (1991) for more details) and that it must be differentiable and smooth for training (see section 5.7 for details). Common ones include the hyperbolic tangent function (tanh), the logistic function (σ) and the rectified linear unit (ReLU, defined as $\text{ReLU}(x) = \max(0, x)$ with the derivative defined as 0 for $x = 0$). See Figure 5.1 for a comparison of their plots.

5.2 Interconnected neurons

Although even a single neuron may be used as a classifier on its own, in practice, neural networks are built out of many interconnected neurons. Historically, different

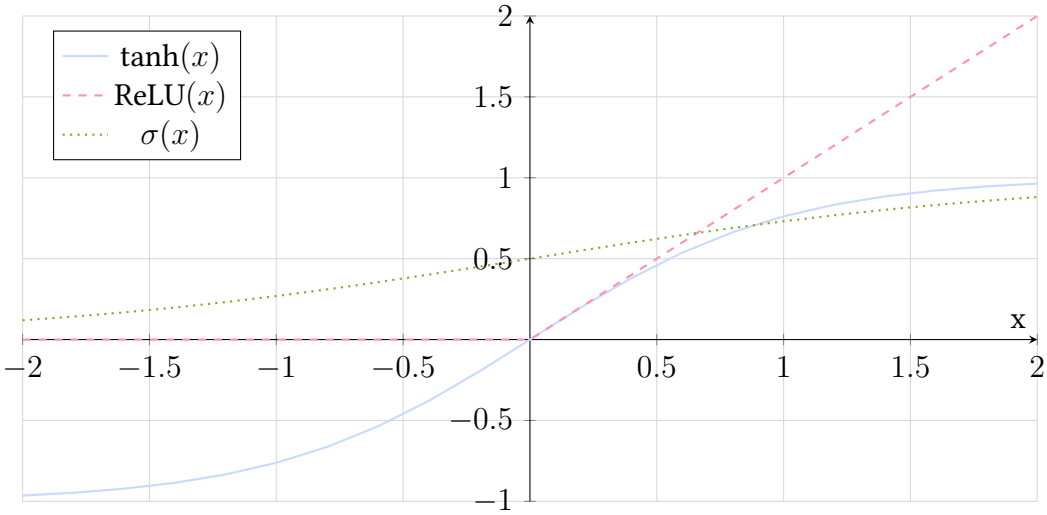


Figure 5.1: Plots of three common activation functions for neural networks.

wirings of individual units were used, but today, the most common practice is to build networks out of layers. One layer contains many instances of a single neuron type, all wired identically. While a single neuron computes a function with signature $\mathbb{R}^n \to \mathbb{R}$, a layer computes a function $\mathbb{R}^n \to \mathbb{R}^m$, allowing for easier processing of vectors. If $n = m$, other optimizations are possible, such as residual connections – these allow the network to bypass the layer, making training quicker and allowing practical training of networks with many consecutive layers (Srivastava et al., 2015).

The layering of neurons is the reason for the non-linearity requirement for activation functions. If the activation function were linear, putting multiple neurons in a sequence would not inrease the predictive power over a single layer. See the following derivation for two neurons, one with inputs $x_1, ..., x_n$, weights $w_0, ..., w_n$ and activation function $f(x) = a \cdot x + b$, the second one connected to its output with weights v_0 and v_1 and activation function g:

$$h = f\left(w_0 + \sum_{i=1}^{n} x_i \cdot w_i\right) = a\left(w_0 + \sum_{i=1}^{n} x_i \cdot w_i\right) + b$$
$$o = g(v_0 + v_1 \cdot h) \tag{5.3}$$

$$o = g\left(v_0 + v_1\left(a\left(w_0 + \sum_{i=1}^{n} x_i \cdot w_i\right) + b\right)\right) \tag{5.4}$$

$$o = g\left(v_0 + v_1 \cdot a \cdot w_0 + v_1 \cdot a \cdot \sum_{i=1}^{n}(x_i \cdot w_i) + v_1 \cdot b\right) \tag{5.5}$$

$$o = g\left(v_0 + v_1 \cdot a \cdot w_0 + v_1 \cdot b + \sum_{i=1}^{n} x_i \cdot v_1 \cdot a \cdot w_i\right) \tag{5.6}$$

If we define $w'_0 = v_0 + v_1 \cdot a \cdot w_0 + v_1 \cdot b$ and $\forall i \in 1...n\colon w'_i = v_1 \cdot a \cdot w_i$, we get:

$$o = g\left(w'_0 + \sum_{i=1}^{n} x_i \cdot w'_i\right) \tag{5.7}$$

Which is a single neuron calculating a function equal to the two neurons defined previously. Therefore, multiple layers of neurons with a linear activation function cannot compute functions different from a single layer.

The same derivation cannot be generally done when the activation functions are non-linear, and it has been both theoretically and empirically confirmed that networks with multiple layers can approximate more general functions than single-layer ones (Cybenko, 1989; Barron, 1993).

5.3 Processing sequences

When we need to model sequences with interdependent steps, as is common in natural language processing, recurrent neural network cells (RNN cells) are often used.

These cells process input as a series of *timesteps* by having an internal state, which is updated at each timestep and copied over to the next one and which is used in addition to the cell input to produce output.

RNNs were possibly first used in practice by Hopfield (1982), but emerged in their modern form with the invention of Long-Short Term Memory (LSTM) cells by Hochreiter et al. (1997), which addressed problems with training the older models. An even newer type of RNN cell, slightly simpler in its internal structure than the LSTM, is the Gated Recurrent Unit (GRU) cell (Cho et al., 2014), used in this project.

The GRU cell has two inner gates, the reset gate r and the update gate z, that modify its memory storage, allowing the cell to select what is kept and what is updated. Otherwise, it contains a basic neuron, which differs from a non-RNN neuron only in the fact that it computes its output based on the memory as well as the input. See Figure 5.2 for a diagram of the unit.

The reset gate r produces a single number between 0 and 1 based on the current input and memory, which multiplies the previous memory state coming into the basic neuron. It therefore modifies the strength of the memory's influence. The update gate z mixes the output o of the basic neuron with the previous memory using a convex combination, thus selecting whether the memory will be kept as is or overwritten with the basic neuron's output. The output of the update gate h serves as the GRU output. Unlike LSTM cells, which can use different values for output and memory transfer, the GRU cell outputs just a single value, which is used both as the cell output and as the memory to be read in the next step.

The GRU cell produces its output based on the following equations:

$$r = \sigma(w_r \cdot x + u_r \cdot h_{t-1}) \tag{5.8}$$

$$z = \sigma(w_z \cdot x + u_z \cdot h_{t-1}) \tag{5.9}$$

$$o = f(w_o \cdot x + u_o \cdot r \cdot h_{t-1}) \tag{5.10}$$

$$h_t = z \cdot h_{t-1} + (1 - z) \cdot o \tag{5.11}$$

5.4 Classification

Since neural networks have real-valued output, the usual way of using neural networks for classification is to predict a probability distribution over the possible

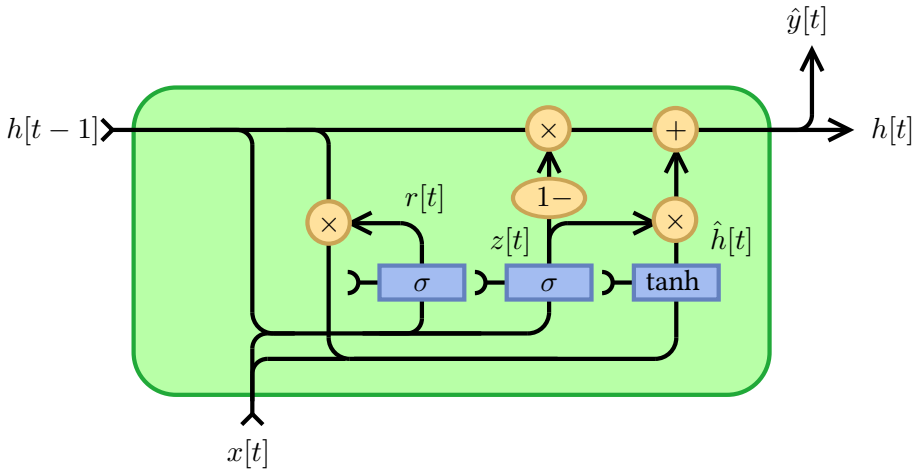


Figure 5.2: Diagram of the Gated Recurrent Unit (Cho et al., 2014). Image courtesy of Jeblad at Wikimedia Commons, licensed under the CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)

classes. For each class, a single neuron on the output layer with identity activation function predicts its score, interpreted as an unnormalized log probability of the class. Therefore, to predict one of k classes, an output layer with k neurons is used. The scores of all classes are then normalized to form a probability distribution by the softmax $\mathbb{R}^n \to \mathbb{R}^n$ function, defined as follows:

$$\forall i \in \{1...n\}\colon \operatorname{softmax}(x)_i \stackrel{def}{=} \frac{e^{x_i}}{\sum_{j\in\{1...n\}} e^{x_j}} \tag{5.12}$$

The softmax function ensures that each output is between 0 and 1 and that they sum up to 1, which are the necessary and sufficient requirements for a probability distribution. To perform classification of a single example, it is possible to simply pick the class corresponding to the maximum value – which is equivalent to picking the maximum valued class before softmax – but softmax is useful in training, where its properties allow for easy comparison of the predicted output to the true distribution.

5.5 Structured prediction

Prediction of structured output, in our case segmentation into morphs, can be done by predicting the boundaries between morphs. For each character position in a word, the model can output true or false depending on whether there is a morph boudary before that character. Since the sequences we are predicting are quite short and the boundaries are relatively well-defined, there is no need for using e.g. CRF-based prediction (Yao et al., 2014).

5.6 Sequence prediction

With RNNs, it is possible to process sequences by decomposing the input sequence to individual timesteps (in our case, invariably, characters) and feeding the characters to the RNN one by one. The output of such a model is a sequence of individual responses. This is fine in the case of the morph structure prediction described above,

but for our auxiliary task of derivational parent prediction, this model is limiting, because it only allows generating outputs of length equal to the input.

When the expected output is shorter than the input, this is not a problem, because we can introduce a special padding character which will bring the two sequences to the same length and will be deleted from the output in a post-processing step.But there are cases where the derivational parent is longer than its child, e.g. *anektovat* ("to annex") → *anexe* ("annexation"). Therefore, we need a way to predict sequences of arbitrary length.

5.6.1 Encoder-decoder model

One way of solving this sequence transduction problem (also called sequence-to-sequence, seq2seq) is the encoder-decoder model (Sutskever et al., 2014; Cho et al., 2014). In it, the network is internally divided into two parts, both composed of an RNN: The encoder, which reads the input, processes it and distills the information found therein to a single data point (usually a real-valued vector of fixed length, obtained as the last hidden state of the last layer of the encoder); and the dynamic decoder, which is seeded by this data (usually its first hidden state is set to it) and generates the output character-by-character until it generates an end-of-sequence token, in a manner similar to the text generation model of Graves (2013). This way, the length of output is disconnected from the length of input.

The prediction the dynamic decoder makes in timestep t is fed back to it as the input of timestep $t + 1$. This backfeed was necessary in the model of Graves (2013), which worked with text directly and was not primed by an encoder, and was inherited by the later encoder-decoder models as an optimization that allows the model to both train and predict better.

Also, the backfeed makes it possible to optimize the selection of a sequence of values by using e.g. beamsearch for finding the optimal sequence, which does not have to be the sequence with the optimal beginning, found by greedily picking the most probable value at each timestep. We can try picking a different one instead, feeding it to the next timestep of the dynamic decoder to inform the model of our choice, and see if the final sequence gets higher overall probability.

5.6.2 Attention-based model

A weakness of the basic encoder-decoder model described above lies in the limited interface between the encoder and the decoder. Because the network is forced to compress all necessary information into a fixed-size representation, it is clear that above a certain length and complexity of the input sequence, the representation can not be lossless and the interface becomes a bottleneck.

One solution was offered by Bahdanau et al. (2014), who introduced an *attention mechanism* that allows the decoder to utilize not only the hidden state of the encoder, but also its outputs, while keeping the lengths of the input and output sequences unrelated.

The attention mechanism is an extra piece of a non-recurrent neural network, which accepts the decoder hidden state and a single timestep of the encoder outputs as input, and produces a single number which acts as the weight of this encoder output timestep. The network is independently applied to each timestep of the encoder, the

output weights are normalized using softmax and the results are used as weights for a weighted sum of encoder outputs. This sum is then presented to the decoder as part of its input, together with the decoder output from the previous timestep.

This arrangement allows the network to look at the relevant parts of the encoded sequence and is especially beneficial if there is some kind of correspondence between the input and output sequences. This is the case for the derivational parent prediction task, where the stem of the derivational child can be viewed as being copied from the derivational parent and changed by MOP alternations; see Figure 4.2 for an example of this process.

5.7 Training the networks

Unlike the aforementioned EM algorithm, which is trained directly by collecting statistics from the training dataset and using them as hidden variables, neural networks are nowadays trained by gradient descent with backpropagation, i.e. by presenting them examples of input together with the expected output values, quantifying the errors the network makes and compensating for the error by modifying the hidden variables to cancel it out (Werbos, 1982).

The process of backpropagation is similar to a debugging technique used by programmers to pinpoint the source of error in a program – start at the place where the fault manifests itself and notice the source must lie either at the point of manifestation, or before it. That is, the problem is either in the code currently being examined, or the code is correct, but was fed wrong inputs from an erroneous code executed before. If the examined code is found to be correct, look at all its inputs and examine the code that produced those inputs for errors. Continue backtracking until the error is found.

The main difference between debugging and training NNs is that we cannot generally identify "correct" and "erroneous" parts of the network, so instead of finding a part of the network to blame and amending that, we spread the blame across all parts of the network that can be held responsible, in proportion to how they contributed to the difference between the actual and expected result. The proportion of blame can be determined by differentiating the difference between the output values of the individual neurons and the expected values, and the necessary amendment is given by the gradient descent process.

Technically, this process is implemented by first defining and calculating a *loss function*, which measures the error the network makes when predicting output. The loss function we use with our categorical prediction model is cross entropy between the correct and predicted outputs, defined as follows:

$$H(y', y) \stackrel{def}{=} -\sum_{i=1}^{|y|} y'_i \cdot \log_2 y_i \tag{5.13}$$

5.8 Preventing overfitting

A common problem of neural networks, and machine learning in general, is overfitting. Overfitting occurs when the model learns to predict the training set based on non-generalizable clues – the performance on the training set rises, but performance

during prediction decreases. To alleviate overfitting, several techniques are used, two of which are described below.

5.8.1 Dropout

Dropout is simply turning off some connections in a neural network, typically half of them at each timestep (Hinton et al., 2012). According to its original authors, it works by preventing coadaptation of neurons in the network – dropout ensures that each neuron keeps a number of sources for its information instead of relying on just a handful of signals.

5.8.2 Residual connections

Residual connections are connections across a neuron, that allow information to pass through without being processed (He et al., 2015). The information bypassing the neuron is simply summed with its output. Because of this piecewise summation, it requires that the input and output sizes of the layer be equal. It helps training by allowing the loss to backpropagate more easily, without it tending to zero or to infinity as it crosses more an more layers (known as the problem of vanishing or exploding gradients).

Although it helps the most on very deep neural networks, we have found it to be useful even on our, rather small, model.

5.9 Our model

The main aim of this thesis is to build a model for supervised morphological segmentation of Czech words. The two variants of the EM algorithm described in Section 4.2 and Section 4.4 are able to segment only words present in the DeriNet dictionary or recognizable by the MorphoDiTa or UDPipe lemmatizers. A more general solution, capable of segmenting arbitrary words, was desired. The chosen solution is a neural network, trained on data obtained from the EM algorithm.

In alignment with the overarching theme of this thesis, we attempted to utilize derivational information when predicting morphemic segmentation, hoping it would help the network learn the proper boundaries. The way we do this is by constructing a network that learns to jointly predict morphemic structure and derivational parents of words. We prepared several variants of the architecture to compare their relative qualities. The basic network is similar in all cases and consists of an encoder composed of several layers of RNNs, and decoders for predicting either the derivational parent, segmentation, or both. The architecture of the individual pieces is described in the next subsections and a diagram of it is in Figure 5.3.

The encoder is shared and its architecture is static across all our experiments. It is composed of one layer of bidirectional GRU RNN, followed by a configurable number of unidirectional GRU RNNs. The last state of the last layer is transferred to the parent decoder described below. The input of the encoder consists of individual characters, embedded using a character embedding matrix.

The segment prediction is done using an architecture identical to the encoder, with a bidirectional GRU RNN followed by several optional unidirectional layers. In our

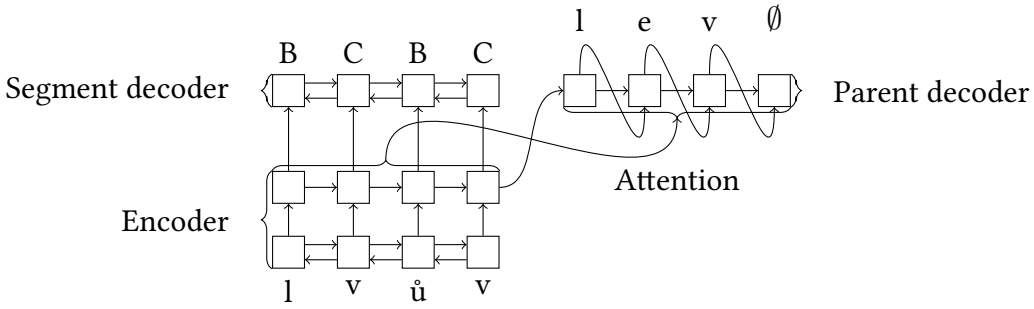


Figure 5.3: A diagram of our neural network model. It is composed of three interconnected parts: The encoder (bottom), the segment decoder (top left) and the parent dynamic decoder (top right). The segment decoder predicts Beginning of a morph or Continuation of the previous morph, while the parent decoder keeps predicting the parent word, until it generates a stop symbol.

evaluation, we only use one. The inputs of the segment prediction are taken directly from the outputs of the encoder.

The parent prediction is done using a dynamic decoder with optionally multiple layers of GRU RNN layers. In our evaluation, we only use one. The initial state of the first layer of the parent decoder is copied from the last layer of the encoder and the inputs of the decoder are input from the previous state summed with an attention over encoder outputs.

Each inter-layer transfer has optional dropout and each internal layer has residual connections.

When training, it is possible to disable a part of the network both for the forward pass and for backpropagation. We use this to disable the parent decoder when training a pure segmentation model.

6. Experiments and evaluation

6.1 Evaluation measures

The quality of segmentation can be measured on three levels: Segmentation points (e.g. the proportion of correctly recognized boundaries between morphs), morphs themselves (e.g. the proportion of correctly recognized segments) and words (e.g. the proportion of words that were fully correctly segmented).

The first two levels don't have a predefined answer size, we can guess more or fewer than the truth. Therefore, it makes sense to measure not only the accuracy of our predictor, but also its precision and recall.

Accuracy is the ratio of correct decisions to all test cases. In the case of segmentation points, the ratio of correctly predicted boundaries and non-boundaries to all possible inter-character boundaries. In the case of words, the ratio of correctly segmented words to all words. It makes little sense to measure accuracy on morphs, because the space of correctly non-recognized non-morphs is too large.

Precision is the ratio of correct predictions to all predictions. In the case of segmentation points, it is the ratio of correctly predicted boundaries to all predicted boundaries. In the case of morphs, it is the ratio of correctly predicted morphs to all predicted morphs. A high precision means that when the system makes a prediction, it is usually correct – but it doesn't say what happens when the system refrains from predicting.

Recall is the ratio of correct predictions to true test cases. In the case of segmentation points, it is the ratio of correctly predicted boundaries to all actual boundaries. In the case of morphs, it is the ratio of correctly predicted morphs to all actual morphs. A high recall means that when the system recognizes most of what there is to predict, but it doesn't say how many of the predictions are correct.

It is also possible to combine precision and recall into a single score by calculating the F1-measure (also called F1-score or simply F-score[1]), which is a harmonic mean of the precision and recall (van Rijsbergen, 1979; Chinchor, 1992).

6.1.1 Other surveyed measures

The measures defined above are the ones that are most commonly used in the literature, but there is a number of other measures.

Creutz; Lindén (2004), authors of the Hutmegs corpus of Finnish and English segmentations, observe that the segmentation point placement may be ambiguous. For example, the word *flies* may be segmented either as *fli·es* or as *flie·s*, depending on whether we consider the inserted *-e-* as part of the root or the ending. If we have no way of distinguishing homonyms, as is the case with both Hutmegs and the data by Slavíčková (1975), there is also another cause of ambiguities, which cannot be solved by introducing a more detailed theory to specify the unclear cases: words that have the same surface form, but different segmentation, e.g. *sladit* as in *slad·k·ý* ("sweet") → *slad·i·t* ("to sweeten") and *lad·i·t* ("to tune") → *s·lad·i·t* ("to match"). The Hutmegs corpus has such ambiguous segmentation points marked and their evaluation metric

[1] The "1" signifies the beta weighting factor, which can be set to other values in order to put higher weight on the recall or the precision. We will only use the equally-weighted formulation.

takes them into account, by considering a boundary correctly marked if any of the fuzzy possibilities are marked.

A similar, but more refined algorithm is given in Nouri et al. (2016). It compares the segmentation decisions globally and penalizes inconsistent handling of ambiguities across examples.

A measure with a different aim is the EMMA evaluation metric (Spiegler et al., 2010), which is designed for segmentation of words into morphemes, i.e. disambiguated segmentation. It is supposed to overcome the labelling isomorphism problem: Unsupervised systems don't have access to linguistic labels and so they usually produce "anonymous labels", they group items into unnamed clusters. In addition to that, they may produce more or fewer clusters than linguists would. EMMA finds a suitable 1:1 mapping and analyses the similarity of the labellings based on that. It is, however, unsuitable for our purposes, because we do not group morphs into morphemes.

Another is the Morpho Challenge metric, used in the eponymous competition (Kurimo et al., 2010), which addresses the same issue in a simpler way: They sample pairs of words containing the same morpheme. The evaluated system gets a point if its analysis of those words also contains an identical label. The final score is calculated as a precision and recall of those points vs. all samples.

6.2 Experiments

In this section, we evaluate all our test systems and report on the results. The systems are:

LCCS the longest contiguous common substring model, defined in Section 4.1

EMo the original, probability-based EM model with joint probability for substitutions as per Subsection 4.3.1

EMm the modified EM model with conditional probability for substitutions as per Section 4.4

EMmh the previous EM model with hand-crafted probability tables, as explained below in Subsection 6.2.1

NNs the neural network based model trained purely on segmentation data, introduced in Section 5.9

NNp the same neural network based model trained jointly on segmentation and parent-prediction data

FlatCat the Morfessor FlatCat model as per Grönroos et al. (2014), in both unsupervised and supervised setting

The neural network models need segmentation data for training and the FlatCat model can optionally use them as well. There are two possible sources of such data: The gold standard (Slavíčková, 1975) and the output of another model, e.g. the LCCS or EM models. We use the latter source, because the gold standard only contains verbs and therefore is not representative of the Czech language as a whole, and it is too small for training our neural network model.

The representativeness of our gold standard data is a large issue. Since the morphology and derivation patterns of Czech verbs differ from those of other categories (Dokulil et al., 1986, p. 387), when we trained the FlatCat model on a heldout portion of the gold standard data, it easily reached 77% morph F1-measure, 92% bounds F1-measure and 60% word accuracy (trained on a 10000-lemmas large excerpt from DeriNet with perplexity parameter set to 10 and word weight of 1.0), but completely failed to segment any nouns, adjectives and adverbs outside the most common adjectival suffix *-ý* and nominal *-ost*.

Because of this problem, apart from using the gold standard data to measure the precision, recall and F1-measure on morphs and boundaries, and accuracy on whole words, we also performed manual comparison of the output of the systems that perform the best on the gold standard data. It shows that apart from the overfitted system mentioned in the previous paragraph, all systems perform on non-verbs comparably with their performance on verbs.

6.2.1 Setting

The data used for evaluating the segmentation models (Slavíčková, 1975) contains 14 581 Czech verbs, split into 59 231 morphs. The parent prediction models were evaluated against a randomly sampled heldout portion of DeriNet 80 000 words large.

We evaluated the EM systems with a beamsearch beam width of 1000, which was experimentally determined to be more than enough to not influence the results.

The EMo model was pretrained for two iterations with MOP alternation probability smoothing parameter $\alpha = 10000$ and 100, respectively, and then trained for further 8 iterations with exponentially decaying $\alpha = (1, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.0000001)$. After these 10 iterations, the model converged and showed the best results. No smoothing was performed on affixes.

The EMm model was pretrained (see Subsection 4.5.3) for two iterations with MOP alternation $\alpha = (100000, 1000)$ and trained for further two iterations with $\alpha = (10, 0.1)$. Again, smoothing was turned off for affixes.

In either case, the smoothing parameter was found to have minimal influence on the result. One pretraining and two training iterations, or two pretraining and one training iteration, were found to be sufficient for the best results, but in both cases the training was let to continue for one more iteration to ensure this. With more iterations, the model started converging towards a worse solution.

Another variant of the EM model, EMmh, was created by hand-crafting the affix tables. The affix tables were first pretrained on the LCCS algorithm as usual (see Subsection 4.5.3) and then they were printed out and edited by manually deleting affix combinations deemed illegal in Czech. We manually cropped affixes that included MOP alternations belonging to stems. The process took approximately one hour of editing. The resulting tables were then loaded back into the model and training continued as usual.

The neural network models were all trained with identical hyperparameter settings, that is batch size of 500, character embeddings of size 64, 2 encoder layers of width 128, 1 parent decoder layer of width 128, 1 segment decoder layer of width 128, dropout strength of 0.5 and learning rate of 0.001 with no decay. The networks were trained for 30 epochs, but some were stopped earlier, when convergence was achieved.

Table 6.1: Table with results measured on the gold-standard data.

Model	Morph			Bounds			Word
	prec.	rec.	F1	prec.	rec.	F1	acc.
LCCS	47.09%	61.28%	53.26%	81.92%	89.96%	85.75%	18.80%
EMo, i10	52.17%	47.91%	49.95%	87.34%	77.87%	82.33%	18.98%
EMm, i0	48.84%	61.94%	54.62%	83.05%	89.98%	86.38%	20.23%
EMm, i1	50.28%	62.48%	55.72%	83.87%	89.83%	86.75%	21.89%
EMm, i2	49.26%	62.14%	54.96%	83.32%	89.96%	86.51%	20.58%
EMm, i3	48.68%	61.87%	54.49%	83.01%	90.00%	86.36%	19.95%
EMmh, i1	64.82%	66.54%	65.67%	91.38%	87.42%	89.36%	35.45%
EMmh, i4	63.55%	66.89%	65.18%	90.83%	87.94%	89.36%	34.65%
NNs on LCCS	31.89%	20.07%	24.64%	82.73%	57.28%	67.69%	0.55%
NNp on LCCS	45.41%	27.69%	34.40%	89.30%	60.94%	72.44%	11.57%
NNs on EMmh	44.60%	48.16%	46.31%	78.91%	76.88%	77.88%	18.07%
NNp on EMmh	45.01%	39.50%	42.08%	79.67%	70.23%	74.65%	21.57%
FlatCat unsup	37.19%	21.57%	27.31%	97.98%	64.92%	78.10%	1.17%
FlatCat sup	66.20%	57.72%	61.67%	92.59%	81.15%	86.49%	31.10%

This setting produced the best results across the experiments. The widths of the layers and the embeddings are slightly larger than necessary and increasing them does not change the results. Similarly, adding another layer to any part of the network does not make the model improve. The model is not very sensitive to changes in learning rate, and the rate we used is optimal or very close to optimal for every run that we report on.

The Morfessor FlatCat model was tried in two scenarios – one without any supervised data, and another with 2 000 examples from the EMmh system (the same data the neural networks use, but smaller). The authors claim that a small amount of supervised data is sufficient (Grönroos et al., 2014) and our experiments confirm that the model does not get meaningfully better with more examples.

6.2.2 Evaluation

The results of evaluation on the gold-standard data are listed in Table 6.1 and sample outputs are to be found in Attachment A.2. They show that the unsupervised Morfessor FlatCat system performs the worst, followed by the neural networks. Our baseline, LCCS, performs better than any of them. The EM system in unsupervised setting can outperform the baseline, especially with the modified MOP alternations probability calculation, but it is in turn outperformed by the supervised FlatCat model. However, even this state of the art is outperformed by the guided EM model with hand-crafted probability tables.

EM model

The final scores of the model are listed in Table 6.1, and an alternative evaluation that does not count out-of-vocabulary words as errors is listed in Table 6.2. The first table is meant for comparison of the algorithm with its competitors, while the second

Table 6.2: Table with results when not measuring on out-of-vocabulary words.

Model	Morph			Bounds			Word
	prec.	rec.	F1	prec.	rec.	F1	acc.
LCCS	48.49%	61.43%	54.20%	82.79%	89.72%	86.12%	19.24%
EMmh, i1	67.76%	66.79%	67.27%	92.79%	87.11%	89.86%	36.23%
EMmh, i4	66.33%	67.15%	66.74%	92.20%	87.64%	89.86%	35.41%
EMm, i0	50.37%	62.11%	55.63%	83.98%	89.74%	86.76%	20.70%
EMm, i1	51.92%	62.65%	56.78%	84.85%	89.59%	87.15%	22.39%
EMm, i2	50.83%	62.30%	55.98%	84.26%	89.72%	86.90%	21.05%
EMm, i3	50.20%	62.04%	55.50%	83.94%	89.76%	86.75%	20.42%

table is of interest to DeriNet, because since the model was trained on the DeriNet dictionary, there are no out-of-vocabulary words in it.

Since the model has two distinct parts – the stemmer and the boundary transfer – we have also performed a detailed error analysis to find out, in which proportion do they contribute to the final error rate. The analysis was performed by manually inspecting the output of the model at various stages. We looked for recurring errors both in the final output (in the annotated verbs by Slavíčková (1975)), and in the stemmer output. The stemmer output was further divided into two parts – one where any MOP alternations were detected in the stem, and one where none were. This allowed us to distinguish error caused by over- and undergeneration of MOP alternations.

The part with MOP alternations was found to contain 77 stemming errors out of 5 537 cases, suggesting a very high precision. Out of these, 23 have an identical pattern of the *-t* ⟶ *-vka* suffix change incorrectly interpreted as *λ* ⟶ *-ka* with an MOP alternation *t* ⟶ *v*, e.g. *spojit* (“to connect”) ⟶ *spojiv·ka* (“conjunctiva”) or *kulhat* (“to limp”) ⟶ *kulhav·ka* (“foot-and-mouth disease”). The rest of the erros shows no discernible pattern, but we can group them by their causes as follows:

- Competition between infrequent MOP alternation patterns and frequent affix patterns, caused by irregularities and rare cases in the language, such as in *ok·o* (“eye”) ⟶ *oč·ko* (“little eye”), which is incorrectly segmented as the prefixation *oko* ⟶ *o·čko* with *čko* being the stem. This is because DeriNet has 167 examples of string suffix pair *-o* ⟶ *-ko*, but 18 639 examples of *λ* ⟶ *o-*. The correct pattern is thus composed of the medium-frequency alternation *k* ⟶ *č* and the rare affix combination, while the incorrect one has the very infrequent MOP alternation *o* ⟶ *č* and a common affix pair. Apparently, the latter overpowers the former in this case. A similar case is the derivation pair *estébé* (“StB, communist secret police”) ⟶ *estéb·ák* (“an officer of StB”), where the model incorrectly uses the final *é* in the parent as a suffix, because the correct MOP alternation – deletion of *é* – is too infrequent in the data.

- Irregularities and errors in DeriNet. We have identified 62 errors in DeriNet, some of which cause missegmentation. In addition to these, there are many cases where the choices made in DeriNet are not wrong, but a different assignment of derivational relations would help our models perform better. For example, in the tree around the word *scanovat* (“to scan”), there are many words with MOP alternations: *scanující* (“scanning”), as well as its alterna-

tive forms *scannující*, *skanující*, *skannující*, *skenující* and *skennující*. They are connected such that the alternatives are always siblings. For our models, it would be better if each alternative had its own subtree; we would rather see e.g. *naskennovaně* ("scanned$_{\text{Adv}}$") derived from *naskennovaný* ("scanned$_{\text{Adj}}$") instead of *nascanovaný* ("scanned$_{\text{Adj}}$"), as fewer required MOP alternations would make stemming easier.

The boundary transfer works well, we were unable to identify any wrong boundaries created by the transfer, i.e. boundaries that would be correct in the parent or child, but became wrong after being transferred across a correctly identified stem.

The main issue is therefore the fact, that many MOP alternations are unrecognized by the algorithm. Their rate is difficult to estimate, but from annotating a random sample of 1 000 words, which contained 21 words with alternations, we expect there to be about 16 000 MOP alternations in total, making for a roughly $\frac{1}{3}$ recall. Each unrecognized alternation while stemming penalizes the model both on recall and on precision, as the wrong stemming causes a wrong morph boundary to be inserted, typically somewhere into the stem, which is then propagated through a large portion of the derivational tree, affecting potentially hundreds of words.

The results of intermediate iterations, which show the progression of the training, do not point to an easily interpretable hierarchy of MOP alternation productivity or predictability. In general, from what we can see, the model correctly recognizes most alternations found deep inside the stems even after the first iteration, as well as many iterations lying near the boundary between the stem and the affixes. Some of those border alternations are only identified in later iterations. For example, the change *e* ⟶ *í* is found in the first iteration in cases of both *pleskav·ý* ("to patter") ⟶ *plískav·ice* ("heavy rain") and *běže·t* ("run") ⟶ *běží·cí* ("running"), but the change *z* ⟶ *s* is found in the first iteration when in *neorenezance* ("Neorenaissance$_{\text{Noun}}$") ⟶ *neorenesanční* ("Neorenaissance$_{\text{Adj}}$"), but in the second iteration in the case of *aktualizovat* ("to update") ⟶ *aktualisující* ("updating")

The only interpretation of which alternations are found in early iterations and which are only recognized later lies in the frequency of the change patterns. For example, *-g* ⟶ *-žka* (as in *filolog* ("philologist$_{\text{Masc}}$") ⟶ *filoložka* ("philologist$_{\text{Fem}}$")) is a frequent pattern, thus the change in it gets discovered later, because its probability as an affix with the change in it is high enough to overpower it initially.

Surprisingly, the algorithm also correctly segments some words with complex suppletive changes, such as *býv·t* ("to be") ⟶ *jsou·cí* ("existent"), although this is probably more due to chance than to the inherent qualities of the algorithm, as e.g. *jí·t* ("to go") ⟶ *chod·it* ("to be going") is segmented incorrectly as *jí·t* ⟶ *chodi·t*, with the *t* being the root.

Testing on inflected forms was not performed due to a lack of gold-standard data to evaluate against and because it has little utility for the main aim of this thesis, i.e. segmenting the DeriNet dictionary. Preliminary experiments have verified that the method works on inflectional relations the same it does on derivational ones, so an extension in this direction is possible and should be easy to do whenever a use case is found. Code that uses the longest-common-substring model to perform the segmentation is already in place and working, code to perform this task using the EM model with separate tables for inflection and derivation should be trivial to add.

Neural segmentation models

Our neural network architecture has two modes of training and prediction: segmentation only and joint segmentation and derivational parent prediction. Therefore, the dataset used for training and development evaluation has two parts: for each word, it lists its segmentation as well as its parent. The parent is taken directly from DeriNet, while the segmentation data has to be created by another machine learning model, since the gold-standard segmentation dataset used for evaluation is too small and too skewed for training.

We tried training the model on two different sources of segmentation data, as we found that depending on the exact source, the segmentation performance differs significantly on both the development and the evaluation dataset. One source is the EMmh model (with hand-crafted probability tables), the other one is the basic LCCS model. In both cases, we trained and evaluated the performance of both joint and segment-only prediction.

The development set was always created as an 80 000 word large heldout portion of the training data, leaving 605 591 words to train on. The final evaluation dataset was the same as for the other models, based on verbs from Slavíčková (1975).

Results of evaluation on the development dataset can be found in Figure 6.1 for the parent prediction and in Figure 6.2 for the segmentation prediction.

The segmentation evaluation results on the development set show an interesting result: when training on the lower-quality data created by LCCS, joint training with parent prediction gives better segmentations than only training to segment. In fact, the pure segmentation model did not appear to train at all for the first few epochs, and changing its learning rate or other hyperparameters did not help. But with the higher-quality data from EMmh, the pure segmentation model performs better on the development set than the joint one. Moreover, the joint model on the better dataset performs worse than the joint model on the worse dataset.

Of course, these data may be misleading, as the four runs are evaluated on two different datasets, which both contain errors. The evaluation on the actual final test set, which was performed last, shows a different pattern of results. There, the low-quality-data pure-segmentation still performs worse than the other models, but the two models trained on better data both perform better than the joint model trained on LCCS output, except for having lower precision on morph boundaries. Also, the difference between the two EMmh-trained models is markedly lower and the joint one even has a higher word accuracy.

A final verdict on the hypothesis that joint training with parent prediction may help the segmentation prediction is therefore inconclusive, but it probably does not help if the training data is good enough.

More detailed error analysis uncovers obvious differences between the results of the joint and segments-only models on the better data. The segments-only algorithm often splits suffixes into individual letters, while the joint algorithm clumps larger sections together. Therefore, they make complementary errors.

Morfessor

We used the SYN2010 corpus of the Czech language (Křen et al., 2010) for the token-frequency-based training and the DeriNet dictionary for the type-frequency-based training. The results in Table 6.3 show that the type-frequency-based model

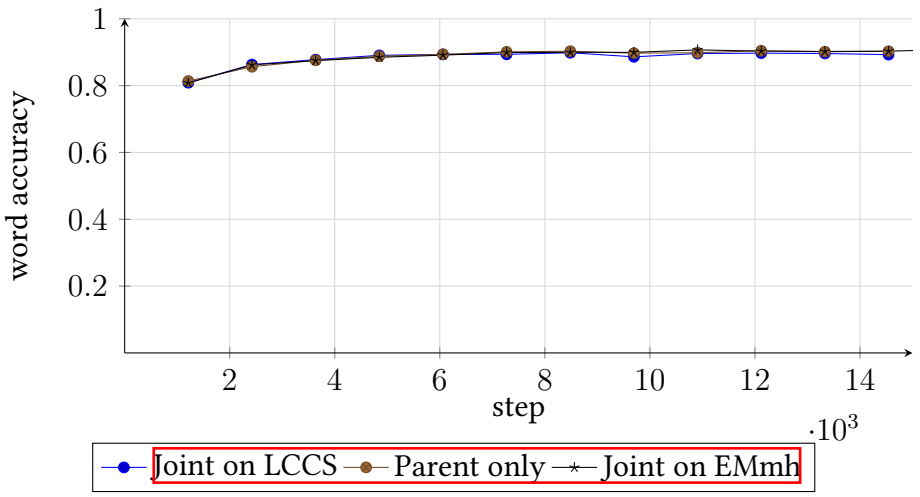


Figure 6.1: Word accuracy of parent prediction on the development set. Since the LCCS/EMmh distinction is only relevant for segmentation, parent only training produces identical results on both datasets.

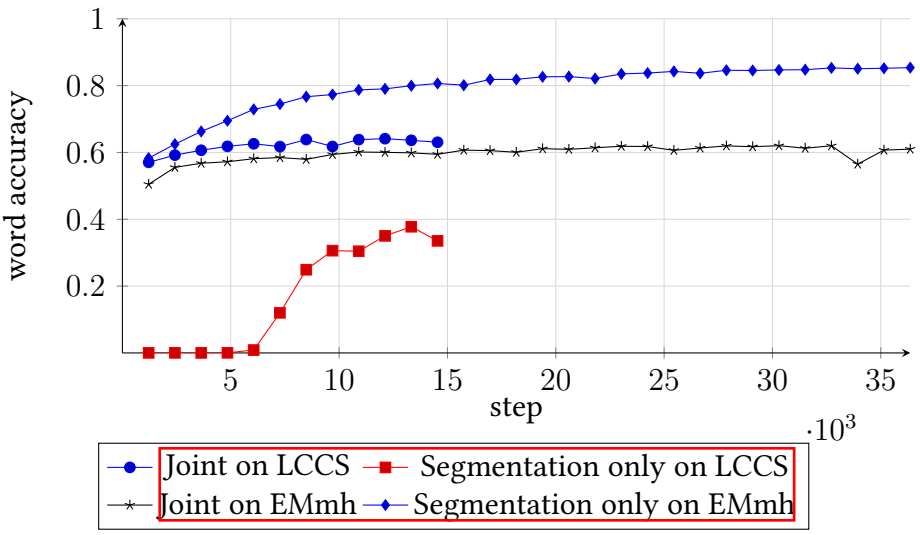


Figure 6.2: Word accuracy of segment prediction on the development set. The development set is a heldout portion of the same dataset the network was trained on, created by another machine learning model, so it contains errors with respect to the gold-standard data. This is why the word accuracy on development set is so much higher than the accuracy on the gold-standard. For the LCCS runs, only the first 12 iterations are shown, but the models did not improve with further training.

Table 6.3: Table with results of Morfessor FlatCat when trained on different data sources (SYN means the SYN2010 corpus and DN means lemmas extracted from DeriNet, the -u suffix annotates fully unsupervised training) and sizes. The corpus data were selected as samples of size 100, 1000, 10000 etc. sentences, which is why their word type counts do not match the counts of data from DeriNet. All models were trained with parameters `-w 1.0 -p 38`, which were found by a grid search to be optimal or near-optimal across a large scale of data sizes. The supervised segmentation data for finetuning FlatCat were obtained from the best EM model with handcrafted tables.

Data	Type	Morph			Bounds			Word
src.	count	prec.	rec.	F1	prec.	rec.	F1	acc.
SYN	700	46.71%	36.52%	40.99%	86.90%	67.03%	75.68%	14.25%
SYN	5012	54.40%	45.25%	49.40%	88.69%	73.57%	80.42%	17.44%
SYN	24411	56.57%	53.46%	54.97%	88.39%	80.47%	84.25%	21.35%
SYN	96391	58.61%	48.22%	52.91%	92.41%	77.34%	84.20%	18.06%
SYN	368958	58.75%	45.09%	51.02%	94.39%	75.48%	83.88%	15.30%
SYN	1580543	57.66%	43.49%	49.58%	94.39%	74.43%	83.23%	14.38%
DN	100	47.32%	51.32%	49.24%	83.48%	80.46%	81.95%	16.54%
DN	1000	44.54%	32.83%	37.80%	88.64%	66.08%	75.72%	14.49%
DN	10000	66.20%	57.72%	61.67%	92.59%	81.15%	86.49%	31.10%
DN	100000	62.14%	47.01%	53.53%	96.57%	77.13%	85.76%	16.72%
DN	1011965	51.37%	30.86%	38.56%	98.19%	66.49%	79.29%	8.27%
DNu	100	0.10%	0.03%	0.04%	100.00%	39.51%	56.64%	0.10%
DNu	1000	0.10%	0.03%	0.04%	100.00%	39.51%	56.64%	0.10%
DNu	10000	33.96%	17.46%	23.06%	95.52%	58.26%	72.38%	0.43%
DNu	100000	37.19%	21.57%	27.31%	97.98%	64.92%	78.10%	1.17%
DNu	1011965	17.01%	7.77%	10.67%	99.03%	55.86%	71.43%	0.71%

generally performs comparably to the one trained on a corpus, even though it was trained on word forms instead of lemmas. A second interesting point is that the models trained on the whole dataset do not perform as well as models trained on just a small sample, despite our best effort at determining the best combination of hyperparameters for each training run. Obviously, the unsupervised training runs perform much worse than the semi-supervised ones.

6.2.3 Discussion

Limitations of the algorithms

The EM model is limited to segmenting along derivational relations. As such, it can not – even theoretically, under the best conditions – recognize morph boundaries created by analogy. An example of these is the analysis of cranberry morphemes mentioned in Subsection 2.1.4, such as the word *malina* (“raspberry”). Because a crucial component of the algorithm is the stem mapping mechanism, it also lacks the capability of reliably segmenting words created by (partial) suppletion, such as *jí·t* (“to go”) ⟶ *chod·it* (“to be going”), where the stems have no morphological relation with each other.

The neural segmentation model doesn't have these limitations, but in the state reported in this thesis, it is held back by the lack of reliable training data. Because the only source of gold standard data, Slavíčková (1975), is too small for training, we have trained our neural models on data produced by the EM model, which causes a stacking of errors of the two algorithms. When we compare the output of the neural system to the gold standard data, it performs worse than the EM model, but we believe that training on higher quality data would reverse this.

Neither model was explicitly trained to recognize segmentations of unmotivated words (words without derivational parent). The EM model is able to segment them with sufficient accuracy via the boundary tranfer algorithm, the NN model does not handle them in any special way and is able to segment them well without seeing any examples of them during training.

The fact that the EM model performs significantly better with hand-crafted tables than with automatically learned ones indicates that training the model by maximum likelihood may not be the best choice. Perhaps a variant of the gradient descent algorithm would perform better, although it would require at least some amount of gold-standard data for training. It would also allow us to introduce more features into the model. This is corroborated by the slow divergence of the EMm model after the first iteration. A better training procedure could perhaps prevent this performance decay.

7. Conclusion

It is a well known fact that derivations are closely related to morphological segmentation. However, as far as we know, this fact was never used in an supervised NLP system to infer the morphematic makeup of words from their derivational neighbors. We have devised several novel methods, ranging from simple to complex, that utilize derivational data in predicting morphological segmentation of Czech words. Our experiments confirm that this is a feasible way of segmenting words and that even a simple baseline method based on derivations is able to outperform state of the art (SOTA) unsupervised segmentation systems.

A more complex method then has the potential to outperform even supervised state of the art segmenters, although in our case we only managed to beat SOTA after sidestepping training issues by manually editing the internal variables of our model. But even without this manual intervention, our systems outperform SOTA in recall measured both on morphs and morph boundaries and get a respectable score in the other categories.

We hope that in the future, large high-quality derivational databases will become available for more languages and that we will be able to use our models to segment languages other than Czech.

7.1 Future work

The first steps taken in the nearest future will be adding the segmentation information into DeriNet. The data format of the current version does not allow storing morphemic composition of words, so the task has to wait until version 2.0 is ready. In this version, the annotation possibilities will be greatly enriched with not only morph segmentation, but also annotation of compounds.

Another rather obvious use of the data would be to predict new derivational relations in DeriNet. With the parent predictor of the neural network, this can be done directly, but the segmentation itself can also be used for this purpose, by looking for words with similar morphemic composition and proposing these as derivational relatives. The algorithms also give us an implicit way of scoring derivations according to their quality, using the probability tables or the loss of the neural network when predicting a given parent.

Another, more tenative use of the data, would be to label edges in DeriNet with the morph change occurring along them. This would likely require some level of morpheme annotation of the morph segmentation, for example by clustering the morphs according to their orthography and distribution.

We have also discovered many errors in DeriNet as a byproduct of working on the segmentation algorithm, and many of the corresponding fixes have not been merged yet. These do not have to wait until the next major release and some of them will be already incorporated in DeriNet 1.6, to be released in the nearest future.

After DeriNet 2.0 is released with annotated compounding, the algorithm could be extended to handle compounds. With the neural network, the extension could be straightforward, simply predicting two parents back-to-back, separated by a special symbol. With the EM-based model, the modifications would have to be more involved.

It is possible to use the tool in multilingual scenarios. We have attempted to evaluate on Finnish, for which we have both a derivational database and gold-standard segmentation data, but the two resources are not compatible with each other, as they differ both in their vocabularies and lemmatization style, resulting in an OOV rate too high for practical evaluation. Future work could include obtaining a good-enough pair of datasets for another language and evaluating on that.

We believe that the model should perform reasonably well at least on European languages, as the model makes only the following general assumptions:

- There is a small and fixed alphabet, from which words are constructed.
- Words can be identified as a separate construct (possibly not applicable to e.g. Mandarin Chinese) and are composed of morphs.
- Derivations exist (probably more-or-less universal) and the derivational relations form trees, or can be expressed as trees with little loss of information. This should not be a problem, but it may not apply to languages with very common compounding.
- Each word has a root to which affixes are attached from the inside out. Each additional affix changes the so-far-created stem just a little bit, if at all (if affixes change the stem heavily, e.g. by suppletion, the models might break). That means the affixes our model can handle come in three flavors: prefixes, suffixes and circumfixes (which are analyzed as a combination of a prefix and a suffix). No infixes (*mal·il·inký* (“tiny”)) or interfixes (*bíl·o·modrý* (“whiteblue”)) are allowed, they would be recognized as some combination of phonological changes and pre/suffixes.

Bibliography

Arcodia, Giorgio Francesco, 2013. *Lexical Derivation in Mandarin Chinese*. Ed. by Her, One-Soon; Chui, Kawai. Taiwan Journal of Linguistics. Chinese Linguistics, no. 03.

Aronoff, Mark, 1985. *Word Formation in Generative Grammar*. 3rd ed. Ed. by Keyser, Samuel Jay. Cambridge, Massachusetts, USA: MIT Press. Linguistic Inquiry Monographs, no. 1. ISBN 0-262-51017-0. ISSN 1049-7501.

Aronoff, Mark; Fudeman, Kirsten, 2011. *What is Morphology?* 2nd ed. West Sussex, United Kingdom: Wiley-Blackwell. ISBN 978-1-4051-9467-9.

Baayen, R. Harald; Piepenbrock, Richard; Gulikers, Leon, 1995. *The CELEX Lexical Database (CD-ROM)*. Philadelphia, Pennsylvania, USA: Linguistic Data Consortium.

Bahdanau, Dzmitry; Cho, Kyunghyun; Bengio, Yoshua, 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*. Vol. abs/1409.0473.

Barron, Andrew R., 1993. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*. Vol. 39, no. 3, pp. 930–945. ISSN 0018-9448.

Barták, Roman, 2013. *Automata and Grammars* [online] [visited on 2018-06-06]. Available from: https://ktiml.mff.cuni.cz/~bartak/automaty/.

Bauer, Laurie, 2014. In: *The Oxford Handbook of Derivational Morphology*. Ed. by Lieber, Rochelle; Štekauer, Pavol. Oxford, United Kingdom: Oxford University Press, chap. Concatenative Derivation, pp. 118–135. Oxford Handbooks in Linguistics. ISBN 978-0-19-964164-2.

Bergmanis, Toms; Goldwater, Sharon, 2017. From Segmentation to Analyses: a Probabilistic Model for Unsupervised Morphology Induction. In: *From Segmentation to Analyses: a Probabilistic Model for Unsupervised Morphology Induction. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 337–346.

Blust, Robert, 2014. In: *The Oxford Handbook of Derivational Morphology*. Ed. by Lieber, Rochelle; Štekauer, Pavol. Oxford, United Kingdom: Oxford University Press, chap. Austronesian, pp. 545–557. Oxford Handbooks in Linguistics. ISBN 978-0-19-964164-2.

Bybee, Joan L., 2013. Usage-based theory and exemplar representation. In: *The Oxford Handbook of Construction Grammar*. Ed. by Hoffman, Thomas; Trousdale, Graeme. Oxford, United Kingdom: Oxford University Press, pp. 49–69.

Calzolari, Nicoletta (Conference Chair) et al. (eds.), 2016. *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia: European Language Resources Association. ISBN 978-2-9517408-9-1.

Can, Burcu; Manandhar, Suresh, 2018. Tree Structured Dirichlet Processes for Hierarchical Morphological Segmentation. *Computational Linguistics* [online]. Vol. 44, no. 2 [visited on 2018-06-06]. ISSN 0891-2017. Available from: https://www.mitpressjournals.org/doi/abs/10.1162/coli_a_00318. To be published.

Carstairs-McCarthy, Andrew, 2005. In: *Handbook of Word-Formation*. Ed. by Štekauer, Pavol; Lieber, Rochelle. Dordrecht, The Netherlands: Springer, chap. Basic Terminology, pp. 5–23. Studies in Natural Language and Linguistic Theory, no. 64. ISBN 978-1-4020-3597-5.

Chinchor, Nancy, 1992. MUC-4 Evaluation Metrics. In: *MUC-4 Evaluation Metrics. Proceedings of the Fourth Message Understanding Conference*, pp. 22–29.

Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua, 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In: *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734.

Chu, Yoeng-Jin; Liu, Tseng-Hong, 1965. On the Shortest Arborescence of a Directed Graph. *Science Sinica*. Vol. 14, pp. 1396–1400.

Creutz, Mathias, 2003. Unsupervised Segmentation of Words Using Prior Distributions of Morph Length and Frequency. In: *Unsupervised Segmentation of Words Using Prior Distributions of Morph Length and Frequency. Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Creutz, Mathias; Lagus, Krista, 2002. Unsupervised Discovery of Morphemes. In: *Unsupervised Discovery of Morphemes. Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*. Stroudsburg, Pennsylvania, USA: Association for Computational Linguistics, vol. 6, pp. 21–30.

Creutz, Mathias; Lagus, Krista, 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. In: *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical Report A81, Publications in Computer and Information Science*. Helsinki University of Technology.

Creutz, Mathias; Lindén, Krister, 2004. Morpheme Segmentation Gold Standards for Finnish and English. In: *Morpheme Segmentation Gold Standards for Finnish and English. Technical Report A77, Publications in Computer and Information Science*. Helsinki University of Technology.

Cybenko, George, 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*. Vol. 2, no. 4, pp. 303–314. ISSN 0932-4194.

Davis, Stuart; Tsujimura, Natsuko, 2014. In: *The Oxford Handbook of Derivational Morphology*. Ed. by Lieber, Rochelle; Štekauer, Pavol. Oxford, United Kingdom: Oxford University Press, chap. Non-concatenative Derivation, pp. 190–218. Oxford Handbooks in Linguistics. ISBN 978-0-19-964164-2.

Déjean, Hervé, 1998. Morphemes As Necessary Concept for Structures Discovery from Untagged Corpora. In: *Morphemes As Necessary Concept for Structures Discovery from Untagged Corpora. Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*. Sydney, Australia: Association for Computational Linguistics, pp. 295–298. NeMLaP3/CoNLL '98. ISBN 0-7258-0634-6.

Dempster, Arthur P.; Laird, Nan M.; Rubin, Donald B., 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*. Vol. 39, no. 1, pp. 1–38. ISSN 0035-9246.

Dobrovoljc, Kaja; Krek, Simon; Holozan, Peter; Erjavec, Tomaž; Romih, Miro, 2015. *Morphological lexicon Sloleks 1.2*. Available also from: http://hdl.handle.net/11356/1039. Slovenian language resource repository CLARIN.SI.

Dobrovský, Josef, 1791. Über den Ursprung und die Bildung der slawischen und insbesonder der böhmischen Sprache. In: Tomsa, František Jan. *Vollständiges Wörterbuch der böhmisch-, deutsch-, und lateinischen Sprache*.

Dokulil, Miloš; Horálek, Karel; Hůrková, Jiřina; Knappová, Miloslava; Petr, Jan, et al., 1986. *Mluvnice češtiny (1)*. 1st ed. Prague, Czech Republic: Academia.

Feldman, Laurie Beth, 1994. Beyond Orthography and Phonology: Differences between Inflections and Derivations. *Journal of Memory and Language*. Vol. 33, no. 4, pp. 442–470.

Furdík, Juraj, 1978. Slovotvorná motivovanosť slovnej zásoby v slovenčine. *Studia Academica Slovaca. 7. Prednášky XIV. letného seminára slovenského jazyka a kultúry*, pp. 103–115.

Geman, Stuart Alan; Geman, Donald Jay, 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. PAMI-6, no. 6, pp. 721–741. ISSN 0162-8828.

Goldsmith, John, 2001. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*. Vol. 27, no. 2, pp. 153–198. ISSN 0891-2017.

Goldwater, Sharon; Griffiths, Thomas L.; Johnson, Mark, 2005. Interpolating Between Types and Tokens by Estimating Power-law Generators. In: Weiss, Y.; Schölkopf, B.; Platt, J. C. (eds.). *Proceedings of the 18th International Conference on Neural Information Processing Systems*. Vancouver, British Columbia, Canada: MIT Press, pp. 459–466. NIPS'05.

Graves, Alex, 2013. Generating Sequences With Recurrent Neural Networks [online] [visited on 2018-06-06]. Available from: https://arxiv.org/abs/1308.0850.

Grčar, Miha; Krek, Simon; Dobrovoljc, Kaja, 2012. Obeliks: statistični oblikoskladenjski označevalnik in lematizator za slovenski jezik. In: Erjavec, T.; Žganec Gros, J. (eds.). *Proceedings of the 8th Language Technologies Conference*. Ljubljana, Slovenia: Institut Jožef Stefan, pp. 89–94.

Grönroos, Stig-Arne; Virpioja, Sami; Smit, Peter; Kurimo, Mikko, 2014. Morfessor FlatCat: An HMM-Based Method for Unsupervised and Semi-Supervised Learning of Morphology. In: *Morfessor FlatCat: An HMM-Based Method for Unsupervised and Semi-Supervised Learning of Morphology. Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 1177–1185.

Ten Hacken, Pius, 2014. In: *The Oxford Handbook of Derivational Morphology*. Ed. by Lieber, Rochelle; Štekauer, Pavol. Oxford, United Kingdom: Oxford University Press, chap. Delineating Derivation and Inflection, pp. 10–25. Oxford Handbooks in Linguistics. ISBN 978-0-19-964164-2.

Hafer, Margaret A.; Weiss, Stephen F., 1974. Word Segmentation by Letter Successor Varieties. *Information Storage and Retrieval*. Vol. 10, no. 11, pp. 371–385. ISSN 0020-0271.

Hajič, Jan, 2004. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Prague, Czech Republic: Karolinum. ISBN 80-246-0282-2.

Harris, Zellig Sabbettai, 1955. From Phoneme to Morpheme. *Language*. Vol. 31, no. 2, pp. 190–222.

Haspelmath, Martin; Sims, Andrea D., 2015. *O čem je morfologie*. 1st ed. Trans. by Klégr, Aleš; Vašků, Kateřina. Prague, Czech Republic: Karolinum. ISBN 978-80-246-2504-1.

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian, 2015. Deep Residual Learning for Image Recognition. *CoRR*.

Hinton, Geoffrey E.; Srivastava, Nitish; Krizhevsky, Alex; Sutskever, Ilya; Salakhutdinov, Ruslan, 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*.

Hladký, Josef, 1987. Word division and syllabification in English. *Brno studies in English*. Vol. 17, no. K9, pp. 123–130. ISSN 0231-5351.

Hochreiter, Sepp; Schmidhuber, Jürgen, 1997. Long Short-term Memory. Vol. 9, no. 8, pp. 1735–1780.

Hockett, Charles Francis, 1954. Two Models of Grammatical Description. *Word*. Vol. 10, no. 2-3, pp. 210–234.

Hopfield, John Joseph, 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*. Vol. 79, no. 8, pp. 2554–2558. ISSN 0027-8424.

Hornik, Kurt, 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks*. Vol. 4, no. 2, pp. 251–257. ISSN 0893-6080.

Hrušecký, Michal; Hlaváčová, Jaroslava, 2010. Automatické rozpoznávání předpon a přípon s pomocí nástroje Affisix. In: Pardubská, Dana (ed.). *Informačné technológie – Aplikácie a Teória, Zborník príspevkov prezentovaných na konferencii ITAT*. Seňa, Slovakia, pp. 63–67. ISBN 978-80-970179-3-4.

Karttunen, Lauri; Beesley, Kenneth R., 2001. A Short History of Two-Level Morphology. In: *A Short History of Two-Level Morphology. Twenty Years of Two-Level Morphology, an ESSLLI 2001 Special Event*. Helsinki, Finland.

Kneser, Reinhard; Ney, Hermann, 1995. Improved Backing-off for n-gram Language Modeling. In: *Improved Backing-off for n-gram Language Modeling. Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1, pp. 181–184. ISBN 0-7803-2431-5. ISSN 1520-6149.

Křen, Michal et al., 2010. *SYN2010: balanced corpus of written Czech*. Prague, Czech Republic: Institute of the Czech National Corpus, Charles University. Available also from: http://www.korpus.cz.

Kurfali, Murathan; Üstün, Ahmet; Can, Burcu, 2017. A Trie-Structured Bayesian Model for Unsupervised Morphological Segmentation.

Kurimo, Mikko; Virpioja, Sami; Turunen, Ville T. (eds.), 2010. *Proceedings of the Morpho Challenge 2010 workshop*. Technical Report TKK-ICS-R37. Espoo, Finland.

Kyjánek, Lukáš, 2018. *Morphological Resources of Derivational Word-Formation Relations*. Praha, Czech Republic: ÚFAL MFF UK. ISSN 1214-5521. Available also from: http://ufal.mff.cuni.cz/techrep/tr61.pdf. Technical report. To be published.

Lango, Mateusz; Ševčíková, Magda; Žabokrtský, Zdeněk, 2018. Semi-Automatic Construction of Word-Formation Networks (for Polish and Spanish). In: Calzolari, Nicoletta et al. (eds.). *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018*. Miyazaki, Japan: European Language Resources Association (ELRA).

Lehtonen, Minna; Monahan, Philip J.; Poeppel, David, 2011. Evidence for Early Morphological Decomposition: Combining Masked Priming with Magnetoencephalography. *Journal of Cognitive Neuroscience*. Vol. 23, no. 11, pp. 3366–3379.

Levenshtein, Vladimir Iosifovich, 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*. Vol. 10, pp. 707.

Lieber, Rochelle; Štekauer, Pavol (eds.), 2014. *The Oxford Handbook of Derivational Morphology*. Oxford, United Kingdom: Oxford University Press. Oxford Handbooks in Linguistics. ISBN 978-0-19-964164-2.

Macháček, Dominik; Vidra, Jonáš; Bojar, Ondřej, 2018. Morphological and Language-Agnostic Word Segmentation for NMT. In: *Morphological and Language-Agnostic Word Segmentation for NMT. Proceedings of the 21st International Conference on Text, Speech and Dialogue (TSD 2018)*. Brno, Czech Republic. Available also from: https://arxiv.org/abs/1806.05482. To be published.

Marchand, Hans, 1969. *The categories and types of present-day English word-formation: a synchronic-diachronic approach*. 2nd ed. Munich, Germany: C. H. Beck.

Martincová, Olga; Hlavsa, Zdeněk; Hrušková, Zdenka, 1993. *Pravidla českého pravopisu*. 3rd ed. Prague, Czech Republic: Pansofia. ISBN 80-901373-6-9.

McCulloch, Warren Sturgis; Pitts, Walter, 1943. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*. Vol. 5, no. 4, pp. 115–133. ISSN 1522-9602.

McDonald, Ryan et al., 2013. Universal Dependency Annotation for Multilingual Parsing. In: *Universal Dependency Annotation for Multilingual Parsing. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 92–97.

Mel'čuk, Igor, 2006. *Aspects of the Theory of Morphology*. Ed. by Beck, David. Berlin, Germany: Mouton de Gruyter. Trends in Linguistics. Studies and monographs, no. 146. ISBN 978-3-11-017711-4. ISSN 1861-4302.

Mikolov, Tomas; Yih, Wen-tau; Zweig, Geoffrey, 2013. Linguistic Regularities in Continuous Space Word Representations. In: *Linguistic Regularities in Continuous Space Word Representations. Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 746–751.

Narasimhan, Karthik; Barzilay, Regina; Jaakkola, Tommi, 2015. An Unsupervised Method for Uncovering Morphological Chains. *Transactions of the Association for Computational Linguistics*. Vol. 3, pp. 157–167.

Needleman, Saul B.; Wunsch, Christian D., 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*. Vol. 48, no. 3, pp. 443–453. ISSN 0022-2836.

Neuvel, Sylvain; Fulop, Sean A., 2002. Unsupervised Learning of Morphology Without Morphemes. In: *Unsupervised Learning of Morphology Without Morphemes. Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*.

Nouri, Javad; Yangarber, Roman, 2016. A Novel Evaluation Method for Morphological Segmentation. In: Calzolari, Nicoletta (Conference Chair) et al. (eds.). *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia: European Language Resources Association. ISBN 978-2-9517408-9-1.

Olsen, Susan, 2014. In: *The Oxford Handbook of Derivational Morphology*. Ed. by Lieber, Rochelle; Štekauer, Pavol. Oxford, United Kingdom: Oxford University Press, chap. Delineating Derivation and Compounding, pp. 26–49. Oxford Handbooks in Linguistics. ISBN 978-0-19-964164-2.

Panocová, Renáta, 2017. Internationalisms with the Suffix -ácia and their Adaptation in Slovak. In: *Internationalisms with the Suffix -ácia and their Adaptation in Slovak. Proceedings of the DeriMo workshop*. Milan, Italy.

Van Rijsbergen, C. J., 1979. *Information Retrieval*. 2nd ed. London, United Kingdom: Butterworth & Co. Available also from: http://www.dcs.gla.ac.uk/Keith/Preface.html.

Rissanen, Jorma J., 1996. Fisher Information and Stochastic Complexity. *IEEE Transactions on Information Theory*. Vol. 42, no. 1, pp. 40–47. ISSN 0018-9448.

Ruokolainen, Teemu; Kohonen, Oskar; Sirts, Kairit; Grönroos, Stig-Arne; Kurimo, Mikko; Virpioja, Sami, 2016. A Comparative Study of Minimally Supervised Morphological Segmentation. *Computational Linguistics*. Vol. 42, no. 1, pp. 91–120.

Sakoe, H.; Chiba, S., 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. Vol. 26, no. 1, pp. 43–49. ISSN 0096-3518.

Salvador, Stan; Chan, Philip, 2007. Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intelligent Data Analysis*. Vol. 11, no. 5, pp. 561–580. ISSN 1088-467X.

De Saussure, Ferdinand, 1961. *Course in General Linguistics*. 3rd ed. Ed. by Bally, Charles; Sechehaye, Albert. Trans. from the French by Baskin, Wade. New York, New York, USA: McGraw-Hill Book Company.

Sennrich, Rico; Haddow, Barry; Birch, Alexandra, 2016. Neural Machine Translation of Rare Words with Subword Units. In: *Neural Machine Translation of Rare Words with Subword Units. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. Berlin, Germany.

Ševčíková, Magda; Žabokrtský, Zdeněk; Vidra, Jonáš; Straka, Milan, 2016. The DeriNet lexical network: a language data resource for research into derivation in Czech. *Časopis pro moderní filologii*, pp. 62–76.

Shafaei, Elnaz; Frassinelli, Diego; Lapesa, Gabriella; Padó, Sebastian, 2017. DErivCELEX: Development and Evaluation of a German Derivational Morphology Lexicon based on CELEX. In: *DErivCELEX: Development and Evaluation of a German Derivational Morphology Lexicon based on CELEX. Proceedings of the DeriMo workshop*. Milan, Italy.

Shay, Erin, 2014. In: *The Oxford Handbook of Derivational Morphology*. Ed. by Lieber, Rochelle; Štekauer, Pavol. Oxford, United Kingdom: Oxford University Press, chap. Afroasiatic, pp. 573–590. Oxford Handbooks in Linguistics. ISBN 978-0-19-964164-2.

Šiška, Zbyněk, 1998. *Bázový morfematický slovník češtiny*. 1st ed. Olomouc, Czech Republic: Palacký University. ISBN 80-7067-885-2.

Slavíčková, Eleonora, 1975. *Retrograde morphemic dictionary of Czech language*. 1st ed. Prague, Czech Republic: Academia.

Slavíčková, Eleonora, 2018. *Retrograde Morphemic Dictionary of Czech*. Available also from: http://hdl.handle.net/11234/1-2838 LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Slavíčková, Eleonora; Hlaváčová, Jaroslava; Pognan, Patrice, 2017. *Retrograde Morphemic Dictionary of Czech - verbs*. Available also from: http://hdl.handle.net/11234/1-2546 LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Smith, Temple Ferris; Waterman, Michael Spencer, 1981. Identification of common molecular subsequences. *Journal of Molecular Biology*. Vol. 147, no. 1, pp. 195–197. ISSN 0022-2836.

Šnajder, Jan, 2014. DerivBase.hr: A High-Coverage Derivational Morphology Resource for Croatian. In: *DerivBase.hr: A High-Coverage Derivational Morphology Resource for Croatian. Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland, pp. 3371–3377.

Spencer, Andrew, 1994. Morphological theory and English. *Links & Letters*. No. 1, pp. 71–84. ISSN 1133-7397.

Spiegler, Sebastian; Monson, Christian, 2010. EMMA: A novel Evaluation Metric for Morphological Analysis. In: *EMMA: A novel Evaluation Metric for Morphological Analysis. Proceedings of COLING 2010, 23rd International Conference on Computational Linguistics*. Beijing, China, pp. 1029–1037.

Srivastava, Rupesh; Greff, Klaus; Schmidhuber, Jurgen, 2015. Highway Networks. In: *Highway Networks. International Conference on Machine Learning: Workshop on Deep Learning*. Lille, France.

Štekauer, Pavol; Lieber, Rochelle (eds.), 2005. *Handbook of Word-Formation*. Dordrecht, The Netherlands: Springer. Studies in Natural Language and Linguistic Theory, no. 64. ISBN 978-1-4020-3597-5.

Straka, Milan; Straková, Jana, 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In: *Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Vancouver, Canada: Association for Computational Linguistics, pp. 88–99.

Straková, Jana; Straka, Milan; Hajič, Jan, 2014. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In: *Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland, USA: Association for Computational Linguistics, pp. 13–18.

Stump, Gregory, 2005. In: *Handbook of Word-Formation*. Ed. by Štekauer, Pavol; Lieber, Rochelle. Dordrecht, The Netherlands: Springer, chap. Word-Formation and Inflectional Morphology, pp. 49–71. Studies in Natural Language and Linguistic Theory, no. 64. ISBN 978-1-4020-3597-5.

Sutskever, Ilya; Vinyals, Oriol; Le, Quoc V, 2014. Sequence to Sequence Learning with Neural Networks. In: Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; Weinberger, K. Q. (eds.). *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., pp. 3104–3112.

Trips, Carola, 2014. In: *The Oxford Handbook of Derivational Morphology*. Ed. by Lieber, Rochelle; Štekauer, Pavol. Oxford, United Kingdom: Oxford University Press, chap. Derivation and Historical Change, pp. 384–406. Oxford Handbooks in Linguistics. ISBN 978-0-19-964164-2.

Üstün, Ahmet; Can, Burcu, 2016. Unsupervised Morphological Segmentation Using Neural Word Embeddings. In: *Unsupervised Morphological Segmentation Using Neural Word Embeddings. Statistical Language and Speech Processing: 4th International Conference, SLSP 2016*. Pilsen, Czech Republic, pp. 43–53. ISBN 978-3-319-45924-0.

Uyar, Ahmet, 2009. Google stemming mechanisms. *Journal of Information Science*. Vol. 35, no. 5, pp. 499–514.

Vidra, Jonáš, 2015. *Extending the Lexical Network DeriNet*. Prague, Czech Republic. Bachelor's thesis. Charles University. Supervised by Zdeněk Žabokrtský.

Viterbi, Andrew J., 1967. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*. Vol. 13, no. 2, pp. 260–269. ISSN 0018-9448.

Werbos, Paul John, 1982. Applications of advances in nonlinear sensitivity analysis. In: Drenick, R. F.; Kozin, F. (eds.). *System Modeling and Optimization*. Berlin, Heidelberg: Springer, pp. 762–770. ISBN 978-3-540-39459-4.

Yao, Kaisheng; Peng, Baolin; Zweig, Geoffrey; Yu, Dong; Li, Xiaolong; Gao, Feng, 2014. Recurrent conditional random field for language understanding. In: *Recurrent conditional random field for language understanding. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4077–4081. ISSN 1520-6149.

Žabokrtský, Zdeněk; Ševčíková, Magda; Straka, Milan; Vidra, Jonáš; Limburská, Adéla, 2016. Merging Data Resources for Inflectional and Derivational Morphology in Czech. In: Calzolari, Nicoletta (Conference Chair) et al. (eds.). *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia: European Language Resources Association, pp. 1307–1314. ISBN 978-2-9517408-9-1.

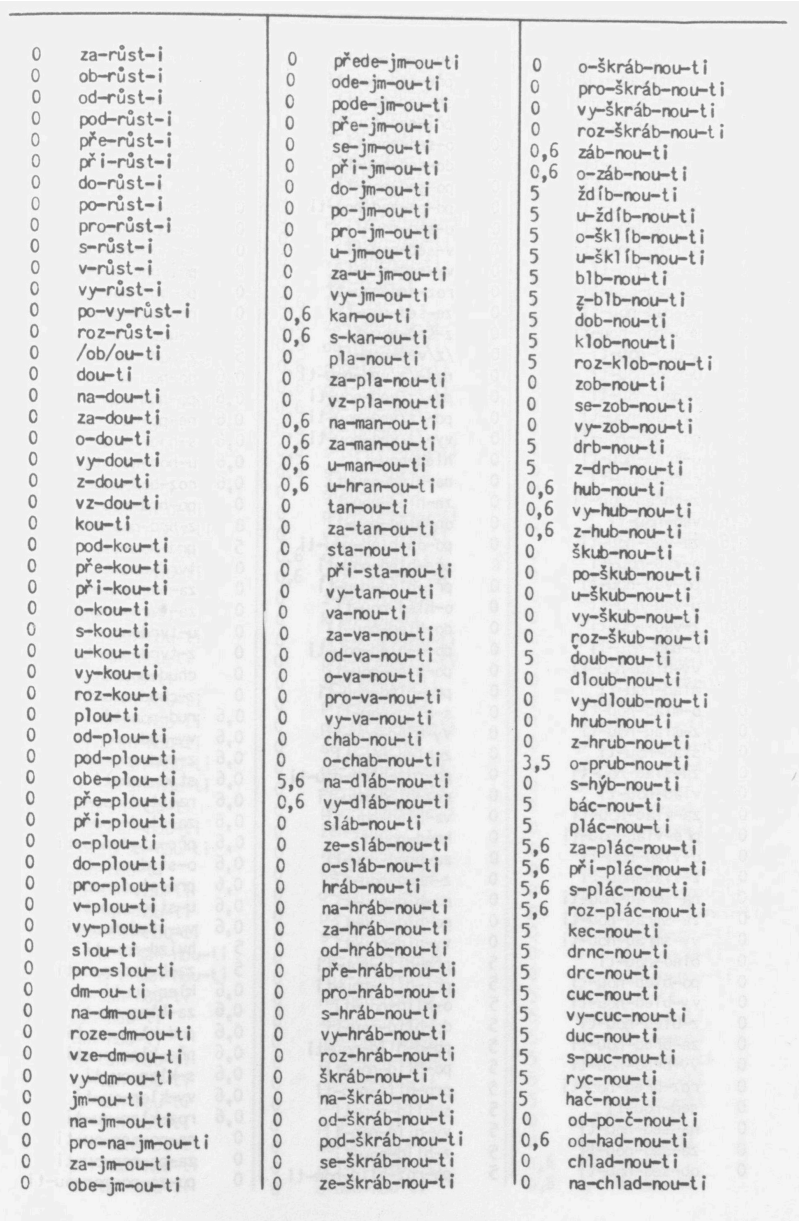
Zeller, Britta; Padó, Sebastian; Šnajder, Jan, 2014. Towards Semantic Validation of a Derivational Lexicon. In: *Towards Semantic Validation of a Derivational Lexicon. Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 1728–1739.

Zingler, Tim, 2017. Evidence against the morpheme: The history of English phonaesthemes. *Language Sciences*. Vol. 62, pp. 76–90. ISSN 0388-0001.

Zipf, George Kingsley, 1932. *Selective Studies and the Principle of Relative Frequency in Language*. Cambridge, Massachusetts, USA: Harvard University Press.

A. Attachments

A.1 An excerpted page from Slavíčková (1975)



0 za-růst-i
0 ob-růst-i
0 od-růst-i
0 pod-růst-i
0 pře-růst-i
0 při-růst-i
0 do-růst-i
0 po-růst-i
0 pro-růst-i
0 s-růst-i
0 v-růst-i
0 vy-růst-i
0 po-vy-růst-i
0 roz-růst-i
0 /ob/ou-ti
0 dou-ti
0 na-dou-ti
0 za-dou-ti
0 o-dou-ti
0 vy-dou-ti
0 z-dou-ti
0 vz-dou-ti
0 kou-ti
0 pod-kou-ti
0 pře-kou-ti
0 při-kou-ti
0 o-kou-ti
0 s-kou-ti
0 u-kou-ti
0 vy-kou-ti
0 roz-kou-ti
0 plou-ti
0 od-plou-ti
0 pod-plou-ti
0 obe-plou-ti
0 pře-plou-ti
0 při-plou-ti
0 o-plou-ti
0 do-plou-ti
0 pro-plou-ti
0 v-plou-ti
0 vy-plou-ti
0 slou-ti
0 pro-slou-ti
0 dm-ou-ti
0 na-dm-ou-ti
0 roze-dm-ou-ti
0 vze-dm-ou-ti
0 vy-dm-ou-ti
0 jm-ou-ti
0 na-jm-ou-ti
0 pro-na-jm-ou-ti
0 za-jm-ou-ti
0 obe-jm-ou-ti

0 přede-jm-ou-ti
0 ode-jm-ou-ti
0 pode-jm-ou-ti
0 pře-jm-ou-ti
0 se-jm-ou-ti
0 při-jm-ou-ti
0 do-jm-ou-ti
0 po-jm-ou-ti
0 pro-jm-ou-ti
0 u-jm-ou-ti
0 za-u-jm-ou-ti
0 vy-jm-ou-ti
0,6 kan-ou-ti
0,6 s-kan-ou-ti
0 pla-nou-ti
0 za-pla-nou-ti
0 vz-pla-nou-ti
0,6 na-man-ou-ti
0,6 za-man-ou-ti
0,6 u-man-ou-ti
0,6 u-hran-ou-ti
0 tan-ou-ti
0 za-tan-ou-ti
0 sta-nou-ti
0 při-sta-nou-ti
0 vy-tan-ou-ti
0 va-nou-ti
0 za-va-nou-ti
0 od-va-nou-ti
0 o-va-nou-ti
0 pro-va-nou-ti
0 vy-va-nou-ti
0 chab-nou-ti
0 o-chab-nou-ti
5,6 na-dláb-nou-ti
0,6 vy-dláb-nou-ti
0 sláb-nou-ti
0 ze-sláb-nou-ti
0 o-sláb-nou-ti
0 hráb-nou-ti
0 na-hráb-nou-ti
0 za-hráb-nou-ti
0 od-hráb-nou-ti
0 pře-hráb-nou-ti
0 pro-hráb-nou-ti
0 s-hráb-nou-ti
0 vy-hráb-nou-ti
0 roz-hráb-nou-ti
0 škráb-nou-ti
0 na-škráb-nou-ti
0 od-škráb-nou-ti
0 pod-škráb-nou-ti
0 se-škráb-nou-ti
0 ze-škráb-nou-ti

0 o-škráb-nou-ti
0 pro-škráb-nou-ti
0 vy-škráb-nou-ti
0 roz-škráb-nou-ti
0,6 záb-nou-ti
0,6 o-záb-nou-ti
5 ždíb-nou-ti
5 u-ždíb-nou-ti
5 o-šklíb-nou-ti
5 u-šklíb-nou-ti
5 blb-nou-ti
5 z-blb-nou-ti
5 dob-nou-ti
5 klob-nou-ti
5 roz-klob-nou-ti
0 zob-nou-ti
0 se-zob-nou-ti
0 vy-zob-nou-ti
5 drb-nou-ti
5 z-drb-nou-ti
0,6 hub-nou-ti
0,6 vy-hub-nou-ti
0,6 z-hub-nou-ti
0 škub-nou-ti
0 po-škub-nou-ti
0 u-škub-nou-ti
0 vy-škub-nou-ti
0 roz-škub-nou-ti
5 doub-nou-ti
0 dloub-nou-ti
0 vy-dloub-nou-ti
0 hrub-nou-ti
0 z-hrub-nou-ti
3,5 o-prub-nou-ti
0 s-hýb-nou-ti
5 bác-nou-ti
5 plác-nou-ti
5,6 za-plác-nou-ti
5,6 při-plác-nou-ti
5,6 s-plác-nou-ti
5,6 roz-plác-nou-ti
5 kec-nou-ti
5 drnc-nou-ti
5 drc-nou-ti
5 cuc-nou-ti
5 vy-cuc-nou-ti
5 duc-nou-ti
5 s-puc-nou-ti
5 ryc-nou-ti
5 hač-nou-ti
0 od-po-č-nou-ti
0,6 od-had-nou-ti
0 chlad-nou-ti
0 na-chlad-nou-ti

Figure A.1: Scanned page 281 from the Retrograde morphemic dictionary of Czech language (Slavíčková, 1975), showing the organization of the dictionary. The numbers in front of each segmented word indicate some of its properties and etymological origin, e.g. 0 means autosemantic Slavic words, 3 means loanwords, 5 indicates markedness and 6 indicates homonymy of the word or its root.

A.2 Sample outputs from selected models

Table A.1: Table with sample outputs of various systems. The words were chosen manually to showcase the differences.

LCCS	EMm i1	EMmh i4
ambici·ózn·ě	ambiciózn·ě	ambiciózn·ě
and·ěl	anděl	anděl
autoris·ov·áv·a·t·eln·ě	autoris·ov·áv·a·t·eln·ě	autoris·ová·va·t·eln·ě
br·u·s·i·č	br·us·i·č	brus·i·č
dat·lův	dat·l·ů·v	dat·l·ů·v
dokt·ůr·ek	doktů·r·ek	dokt·ůre·k
drů·žič·čin	drůžič·čin	drůžič·čin
Hav·lův	Havl·ů·v	Havl·ů·v
hrn·íč·ek	hrn·í·č·ek	hrn·íče·k
ioni·z·ov·a·t	ioniz·ov·a·t	ioniz·ova·t
k·ů·ž·e	kůž·e	kůž·e
l·et·n·í	let·n·í	let·n·í
lit·ev·sk·ý	lit·ev·sk·ý	litev·sk·ý
mali·l·in·k·at·ý	mal·i·link·a·t·ý	mal·i·link·a·t·ý
m·y·s·l·i·v·e·c	m·ysl·i·v·e·c	mysl·i·v·e·c
na·ch·yl·ov·a·t	na·chyl·ov·a·t	na·chyl·ova·t
na·kl·adatel	na·kl·a·d·a·t·el	na·kla·d·a·t·el
Narni·e	Narnie	Narni·e
no·h·a	noh·a	noh·a
od·ch·ov·a·t	od·ch·ov·a·t	od·chov·a·t
odol·n·ý	odol·n·ý	odol·n·ý
odzn·áček	odzná·ček	odzná·ček
pla·zm·a	plazm·a	plazm·a
po·vypr·áv·ě·t	po·vypráv·ě·t	po·vyprávě·t
p·r·aní	p·r·a·ní	pra·ní
pro·pleša·t·iteln·ý	pro·pleša·t·i·t·eln·ý	pro·plešat·i·t·eln·ý
prost·or	prostor	prost·or
překl·adatel	překlad·a·t·el	překlad·a·t·el
st·a·řec	st·ař·e·c	stař·e·c
uz·urp·ov·a·t	uzurp·ov·a·t	uzurp·ova·t
v·olsk·ý	v·ol·sk·ý	vol·sk·ý
vy·pr·av·ov·a·t	vy·pr·av·ov·a·t	vy·prav·ova·t
za·rd·ěn·ý	za·rdě·n·ý	za·rd·ě·n·ý

Table A.2: Table with sample outputs of various systems, continued.

FlatCat DN 10000	NNs on EMmh	NNp on EMmh
ambiciózně	ambic·i·ó·z·n·ě	ambiciózně
a·n·děl	anděl	anděl
autoris·ovávatelně	autorisov·ávat·eln·ě	autorisov·ávat·eln·ě
brusič	brus·ič	brus·ič
datl·ův	datl·ův	datl·ův
do·k·t·ůre·k	doktůr·ek	do·ktůr·ek
drůžič·čin	drůžič·čin	drůžič·čin
Havl·ův	Havl·ův	Havl·ův
hr·ní·ček	hrn·íč·e·k	hrn·íček
ionizova·t	ionizovat	ionizovat
kůže	kůže	kůže
let·ní	letn·í	let·n·í
lit·e·v·ský	litev·sk·ý	lit·ev·ský
mali·link·a·t·ý	malilinkat·ý	malilinkat·ý
myslivec	mysli·v·ec	mysliv·ec
na·chyl·ova·t	na·chyl·ovat	na·chylovat
na·klada·tel	na·kladat·el	na·kladat·el
Narnie	Narn·ie	Narn·ie
od·chov·a·t	od·chovat	od·chovat
o·doln·ý	odol·ný	od·ol·ný
od·zná·ček	odzná·ček	odzná·ček
plazm·a	plazm·a	plazma
po·vyprá·vět	po·vyprávět	po·vy·pr·ávět
pra·ní	praní	praní
proplešatitelný	pro·plešat·it·elný	pro·plešatit·elný
pro·stor	prostor	pro·st·or
pře·klada·tel	překladat·el	pře·kladat·el
st·ařec	s·t·aře·c	stařec
u·zurp·ova·t	uzurpovat	u·zurpovat
vol·ský	vol·sk·ý	vol·sk·ý
vy·prav·ova·t	vy·prav·ovat	vy·prav·ovat
zardě·n·ý	za·rdě·ný	za·rdě·ný