UNIVERSITY OF MISKOLC

FACULTY OF MECHANICAL ENGINEERING AND INFORMATICS

Pattern Matching Based Automated Learning of Inflection Generation and Morphological Analysis

BOOKLET

AUTHOR:
Gábor SZABÓ
MSc in Information Engineering

ACADEMIC SUPERVISOR:
Prof. Dr. László KOVÁCS

Miskolc
2021

Contents

1. Introduction

The main goal of this research project was to create a novel pattern matching based morphology model that can learn the morphological features of the target language in an automated way without applying any artificial intelligence methods, and perform inflection generation and morphological analysis. I approached this problem on two levels: first, I created two single-affix transformation engine models that can learn the inflection rules of a single affix type from a word pair set, then I constructed a higher-level model that can perform inflection generation and morphological analysis for all the affix types of the target language.

The proposed models were evaluated against the Hungarian language, which is a morphologically complex agglutinative language containing many affix types, complex inflection rules and numerous exceptional cases. Even though such languages mean a huge challenge for morphology models, the proposed models were expected to reach high accuracy and generalization factor.

1.1 Problem Domain

Morphology is the lowest level of grammar that analyzes the internal structure of words. Understanding the morphology of a language helps in solving higher level grammatical problems such as the syntactic analysis of sentences or the automated processing of free texts.

The smallest units of words with associated meaning are called morphemes. The two main categories of morphemes are lemmas that are the grammatically correct root forms of the words, and the affixes that modify the base meaning of the words. The affixes that have the same grammatical role are associated with the same affix type, e.g. plural form, accusative case, past tense, etc. Affixes can appear at the beginning (prefixes) or at the end (suffixes) of the words, but sometimes in Hungarian they appear on both sides, in case of for example adjective superlative forms.

The main grammatical role of lemmas is determined by their part of speech (e.g. noun, verb, adjective, etc.). The part of speech of a word can be changed using derivative affix types.

There are several operations that a morphology model could perform:

- Inflection generation: adding a new affix type to a word form
- Morphological analysis: determining all the affix types found in an input word
- Segmentation: determining the affix boundaries in the input word form
- Stemming: cutting off the affixes from the input word form[1]
- Lemmatization: determining the lemma of the input word

The existing morphology models usually target a specific language category for which they can perform these operations optimally. The simpler models usually work well for morphologically simpler language categories such as analytic or isolating languages, in which there are just a few affixes with simpler inflection rules. On the other hand, agglutinative (Finnish, Hungarian, etc.) and fusional (Czech, Slovakian, etc.) languages are morphologically more complex, since they can have a high number of affixes in their words, and each added affix might cause complex transformations in the base word form.

1.2 Research of the Baseline Models

In the dissertation I introduce several existing morphology models, grouping them based on five dimensions. Those models that are used during the evaluation tests as baseline models are presented in detail.

Most of the modern morphology models apply some kind of artificial intelligence methods. The models presented by the SIGMORPHON[2] group fall into this category, since most of them use neural networks. Among these models I chose Helsinki 2016[3] [Öst16],

[1]The output of stemming is the stem, that is not always the same as the lemma.
[2]`https://sigmorphon.github.io`
[3]`https://github.com/robertostling/sigmorphon2016-system`

UF 2017[4] [ZLL17], UTNII 2017[5] [SA17], Hamburg 2018[6] [SKBK18], IITBHU 2018[7] [SKS18] and MSU 2018[8] [Sor18] as baseline models.

The proposed models are also evaluated against three popular segmentation models, including Morfessor 2.0[9] [VSGK13], MORSEL[10] [LCMY09, Lig10] and MorphoChain[11] [NBJ15]. Their similarity is that they can work both in supervised and semi-supervised mode, however, they can only determine the affix boundaries. Besides segmentation models I also chose two morphological analyzers as baseline models: Lemming [MCFS15] and Hunmorph-Ocamorph[12] [TKG+05, THR+06].

As opposed to AI based models, the pattern matching based models usually can only handle a single affix type by themselves. One of the most frequently used pattern matching based morphology model is the finite state transducer (FST) [DlH10] that essentially converts input strings to output strings. In literature we can find several FST categories, one possible implementation is provided by the Lucene library.[13] Another efficient single-affix morphology model is called the tree of aligned suffix rules (TASR) [SF07] that can learn the word-ending transformation rules of a single affix type from a training word pair set. Although both its training and search algorithms are relatively easy and efficient [9], its disadvantage is that it cannot model transformations inside or at the beginning of words.

[4]`https://github.com/valdersoul/conll2017/tree/master/dl`
[5]`https://github.com/hajimes/conll2017-system`
[6]`https://gitlab.com/nats/sigmorphon18`
[7]`https://github.com/abhishek0318/conll-sigmorphon-2018`
[8]`https://github.com/AlexeySorokin/Sigmorphon2018SharedTask`
[9]`https://github.com/aalto-speech/morfessor`
[10]`https://github.com/ConstantineLignos/MORSEL`
[11]`https://github.com/karthikncode/MorphoChain`
[12]`http://mokk.bme.hu/en/resources/hunmorph`
[13]`https://lucene.apache.org`

1.3 Research Goals

The main goal of the research was to create a novel pattern matching based morphology model that can learn the inflection rules of the target language in an automated way, and can perform inflection generation and morphological analysis efficiently, with high accuracy even for morphologically complex agglutinative languages. The architecture of such system can be seen in Figure 1.1.

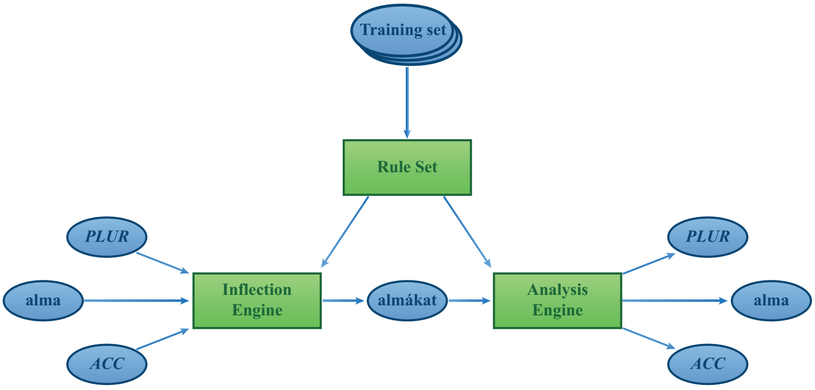


FIGURE 1.1: The architecture of the proposed morphology model

The main hypotheses of the research are the following:

- Pattern matching can be used to learn the morphological features of the Hungarian language.
- The accuracy of the proposed pattern matching models is comparable with that of models applying artificial intelligence.
- The proposed pattern matching models can be implemented in an efficient way so that they can be used to process large data sets and perform inflection generation and morphological analysis in acceptable finite time.

The goal during morphological analysis is to provide so much morphological information about the input word form as possible, including not only its lemma and affix types, but also its intermediate word forms and a calculated weight that specifies how confident the model is in the response. Another goal is to achieve the highest possible generalization capability and accuracy.

The proposed models apply language independent concepts that makes them applicable to languages other than Hungarian as well. However, during the evaluation of the models I use Hungarian training and evaluation data sets, since Hungarian is a morphologically complex agglutinative language, posing great challenge for morphology models of all sorts. The main characteristic of the proposed models is that they use simple string transformations to model inflection rules, and apply pattern matching techniques during search. This means that the set of language dependent components (such as the lemmas or affix types) is minimal. As we will see, these proposed models can achieve high accuracy and generalization capability, even without applying AI methods.

2. Novel Scientific Results

2.1 The Analysis of Existing Hungarian Morphological Analyzers

Since the proposed models are evaluated using Hungarian data sets, choosing a morphological analyzer as a baseline model that can be used for Hungarian words was an important task. Therefore, as a first step, I created a novel method that can be used to compare and rank such models. This method was applied on the following four analyzers:

- Hunmorph-Ocamorph[1] [TKG+05, THR+06]
- Hunmorph-Foma[2]
- Humor [PT93, PK99]
- Hunspell[3]

This comparison method is based on several measured values. The output of the morphological analyzer A for the word $w \in W$ is $A\left(w\right)$, while the set of recognized words by the analyzer A is

$$W^A = \left\{ w \in W \mid \left|A\left(w\right)\right| > 0 \right\} \tag{2.1}$$

Besides that, we can also measure the recognition ratio of the analyzer A:

$$\nu^A = \frac{\left|W^A\right|}{\left|W\right|} \tag{2.2}$$

Based on the set of recognized words by the morphological analyzers A_i and A_j, we can calculate their recognition similarity:

$$S^R_{A_i,A_j} = \frac{\left|W^{A_i} \cap W^{A_j}\right|}{\left|W^{A_i} \cup W^{A_j}\right|} \tag{2.3}$$

[1]`http://mokk.bme.hu/en/resources/hunmorph`
[2]`https://github.com/r0ller/hunmorph-foma`
[3]`https://hunspell.github.io`

The similarity of their token systems can also be calculated, given there exists an m_{A_i,A_j} mapping between the two token systems,[4] and $T^{A_i} \in \mathbb{T}^{A_i}$ denotes the tokens of the morphological analyzer A_i. The similarity of the two token systems can be defined as

$$S^{T}_{A_i,A_j,m_{A_i,A_j}} = \frac{\left|\left\{T^{A_i} \mid \exists T^{A_j} : m_{A_i,A_j}\left(T^{A_i}\right) = T^{A_j}\right\}\right|}{\left|\mathbb{T}^{A_i}\right|} \tag{2.4}$$

The mapping similarity of two analyzers, denoted by $S^{M}_{A_i,A_j,m_{A_i,A_j}}$ is none other than the ratio of such words that result in equivalent morphological structures in case of the two analyzers:

$$\frac{\left|\left\{w \in W^{A_i} \cup W^{A_j} \mid \left|m_{A_i,A_j}\left(t\left(A_i\left(w\right)\right)\right) \cap t\left(A_j\left(w\right)\right)\right| > 0\right\}\right|}{\left|W^{A_i} \cup W^{A_j}\right|} \tag{2.5}$$

where $t\left(A_i\left(w\right)\right)$ denotes the tokens returned by the analyzer A_i for the word w.

I also convert the three similarity formulae to distance values. The distance value is the multiplicative inverse of the related similarity value, for example the recognition distance is the reciprocal of the recognition similarity. Finally, the cumulative distance is the sum of the three distance values:

$$D^{C}_{A_i,A_j,m_{A_i,A_j}} = D^{R}_{A_i,A_j} + D^{T}_{A_i,A_j,m_{A_i,A_j}} + D^{M}_{A_i,A_j,m_{A_i,A_j}} \tag{2.6}$$

These measurements were evaluated against a Hungarian data set that was assembled using an automated data generator algorithm. The details of this algorithm is part of the fifth thesis. The calculated distance values are also presented in 2D Eucledian space, so that we can visually see the distances of the examined morphological analyzers. The cumulative distances are visualized in Figure 2.1.

We can see in this figure that Hunspell and Humor are further away from Hunmorph-Ocamorph and Hunmorph-Foma. The latter two analyzers are similar to each other, but the detailed analysis of the results shows that the Foma based analyzer has incorrect responses that make it less usable than Hunmorph-Ocamorph.

[4]One possible token mapping among the four examined analyzers can be found in the appendix of the disseration.

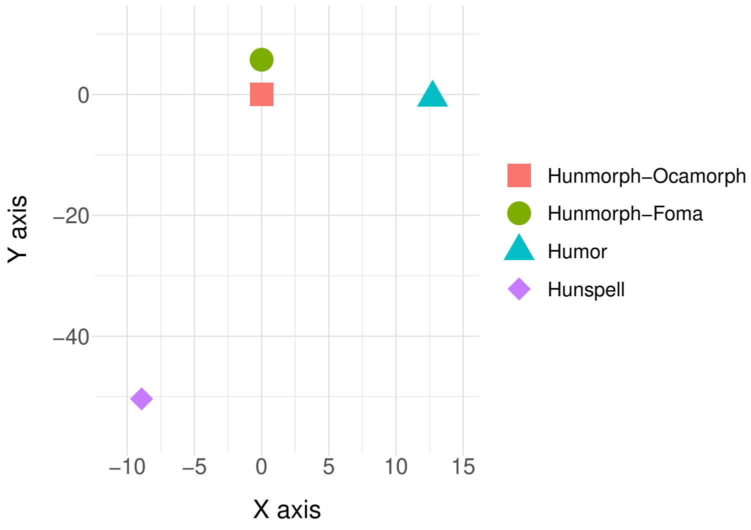


FIGURE 2.1: The cumulative distances of the four examined morphological analyzers

Thesis 1 ***[1]***

I have designed and implemented a new method to compare, analyze, evaluate and rank morphological analyzers. This method is based on novel formulae to calculate similarity and distance values among the different analyzers, including the recognition similarity, token similarity, mapping similarity; as well as the recognition distance, token distance, mapping distance and cumulative distance. I applied this analysis method on four popular morphological analyzers of the Hungarian language, namely Hunmorph-Ocamorph, Hunmorph-Foma, Humor and Hunspell. For the evaluation, I created a token mapping among these analyzers as well. Based on the performed evaluation, Hunmorph-Ocamorph proved to be the most usable model among the four analyzers.

2.2 Single-Affix Models

The first step towards learning the morphological features of the target language is to learn the inflection rules of a single affix type, based on a training word pair set. To achieve this goal, I proposed two models: a lattice based model and the ASTRA model.

2.2.1 Lattice Based Model

The concept behind the lattice based model is rooted in lattice theory [Bir40] and formal concept analysis (FCA) [GW12]. The generated rules are stored in a compact lattice structure.

The first problem to solve during the training phase of the lattice based model is to determine the changing parts of the training word pairs. To solve this problem, I use the classical Levenshtein distance [Lev66], applying a modified cost function that takes the phonetic attributes of each character into account. This way, the cost of a single step is equal to the number of changing phonetic attributes (instead of 0 or 1), therefore it is easier to choose the correct list of elementary transformation steps between two words.

After determining the list of elementary transformation steps for every word pair in the training set, we can construct the transformation rules, having the following structure:

$$R^L = \left(\alpha^L, \sigma^L, \omega^L, \eta_f^L, \eta_b^L, \Delta^L\right) \tag{2.7}$$

where

- α^L is the prefix of the rule (i.e. the characters before the transformation),
- σ^L is the core of the rule (i.e. the characters to be changed),
- ω^L is the postfix of the rule (i.e. the characters after the transformation),
- η_f^L is the front index of the rule context occurrence in the source word from its beginning,
- η_b^L is the back index of the rule context occurrence in the source word from its end, and
- $\Delta^L = \left\langle \delta_i^L \right\rangle$ is the list of elementary transformation steps.

In order to store these rules in a lattice that can generalize well, we need to generate the intersections of the transformation rules, and determine their parent-child relationships. Intersection can be done by rule components: we intersect the prefix components from the end, the core in its entirety, and the postfix from the start. If any component pairs cannot be intersected, then the two rules are disjoint. Intersecting the indices and the transformation steps can only be done if they are identical for the two rules. The parent-child relationship is determined by the $\subseteq$ operator.

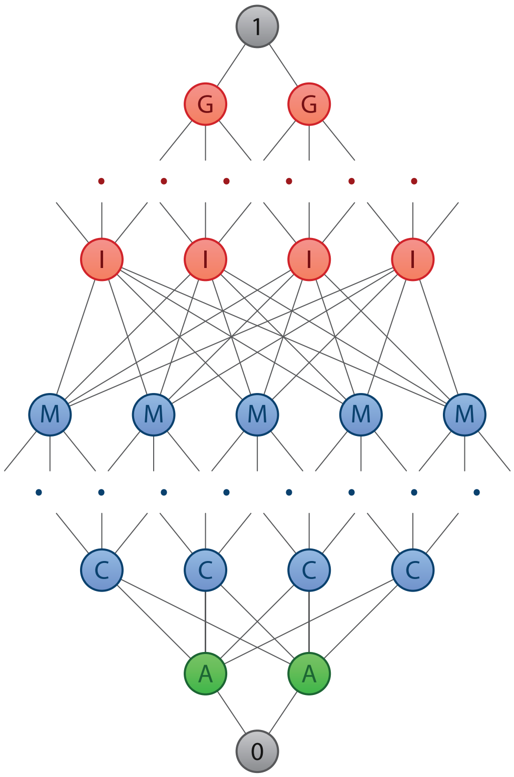


FIGURE 2.2: The structure of a sample lattice

The structure of a sample lattice can be seen in Figure 2.2. As we can see, the nodes can be categorized into several categories. Understanding the role of each node category can help in minimizing the lattice size.

If we insert every rule intersection in the lattice, we get a complete lattice that contains every information about the training data set. Since complete lattices can be very large, I also propose two more lattice builder algorithms. The so-called consistent lattice builder algorithm eliminates every inconsistent (*G* and *I*) node from the top of the lattice. This means that if we find a matching rule in the lattice for an input word, we can directly apply its transformation steps with a high confidence. The minimal lattice builder algorithm first builds a consistent lattice, then using the children of the root, it builds another lattice that has a very small size, contains the maximal consistent rules (*M*) and also the inconsistent rules that speed up the search process.

Analyzing these lattice builders, I proved that the consistency of a node can only change if it was originally consistent, and a new consistent descendant node is inserted into the lattice. This proposition helps in identifying the important cases during the lattice building process.

During inflection, the goal is to find the highest consistent rule whose context matches the input word, and then apply its transformation steps one by one. The lattice based model is asymmetric, which means that a given lattice can be applied either for inflection generation or morphological analysis, based on how it was built.

2.2.2 ASTRA

The proposed ASTRA model (*Atomic String Transformation Rule Assembler*) has a simplified rule model that omits position indices and is based on string transformations (changing substring and replacement string pairs) instead of elementary transformation steps. The atomic rule model has the following structure:

$$R^A = \left(\alpha^A, \sigma^A, \tau^A, \omega^A\right) \tag{2.8}$$

where

- α^A is the prefix of the rule (i.e. some characters before the transformation),
- σ^A is the changing substring (i.e. the characters that need to be replaced in the base word form),
- τ^A is the replacement string (i.e. the characters that need to replaced in the base word form), and
- ω^A is the postfix of the rule (i.e. some characters after the transformation).

During the training phase, we first extend the input word pairs with two special characters: \$ marks the word start, while $\#$ marks the word end. Then, we split each word pair to matching segments. A segment contains a related substring from each word in a word pair. If the two substrings are identical, then the segment is called invariant, otherwise it is called variant. The core of the atomic rules (the σ^A and τ^A components) are constructed from variant segments.

In case of the rule generated for a word pair using the minimal context, $\left|\alpha^A\right| = \left|\omega^A\right| = 0$. The subsequent rules are generated by extending the rule context with one character on the left and right side. This way we generate a number of atomic rules for each word pair, that are then grouped in so-called rule groups. A rule group contains atomic rules with the same context.

To speed up the search process during inflection generation or morphological analysis, ASTRA can also process the rules in parallel. Another way to speed up searching is to insert the rule groups in a prefix tree based on their contexts.

As we can see from the training process and rule structure, the ASTRA model is incremental, and its stored rules are symmetric. During inflection, we must find the best matching rules for the input word, and apply them one by one. In case of overlapping rules, only the one with the higher fitness value is used. The fitness function is defined as

$$f\left(R^A \mid w\right) = \frac{\left|\gamma\left(R^A\right)\right|}{|w|} \cdot \theta\left(\gamma\left(R^A\right), w\right) \tag{2.9}$$

where $\gamma\left(R^A\right)$ is the rule context and the simplest implementation of the θ function returns 1 if the $\gamma\left(R^A\right)$ rule context is contained by the input word w, and 0 otherwise.

It can be easily seen that if a word pair can be found in the training data set, then the model will be able to transform its word correctly.

During morphological analysis, the same rules can be used, only the rule groups and the contexts change. While the R^A atomic rule has a context of $\alpha^A + \sigma^A + \omega^A$ during inflection generation, this context changes to $\alpha^A + \tau^A + \omega^A$ during morphological analysis.

2.2.3 Experiments

To evaluate the proposed models, I compared them with a simple dictionary based system, the FST implementation of the Lucene[5] library and a TASR implementation[6] developed by me for the experiments. The proposed models included the complete, consistent and minimal lattice, as well as the sequential, parallel and prefix tree based ASTRA.

Regarding the training time, the three lattice builder algorithms and the TASR model show significant increase as we extend the training data set, while the prefix tree based and normal ASTRA model, the FST and the dictionary based system take less time to be trained.

The ASTRA model generated the most rules, reaching the largest model size, while the TASR model became the second. The complete and consistent lattice, and the FST model had smaller size, but the minimal lattice became the most compact.

The search time was way below 1 second for all the evaluated models. It is interesting to see that although the sequential ASTRA model had the slowest search time, the prefix tree based ASTRA was quicker to find the matching rules than the lattice based models.

The accuracy of the examined models can be seen in Figure 2.3. The set of training and test data sets were disjoint in all cases. In the figure we can see that the ASTRA and TASR models reached the highest accuracy values, but the ASTRA model was already more accurate using smaller training data sets. The FST model and the dictionary based system is not part of the diagram, since they cannot transform previously unseen words correctly.

The test results using 10 thousand training word pairs are summarized by Table 2.1.

[5] `https://lucene.apache.org`

[6] `https://github.com/szgabsz91/morpher`

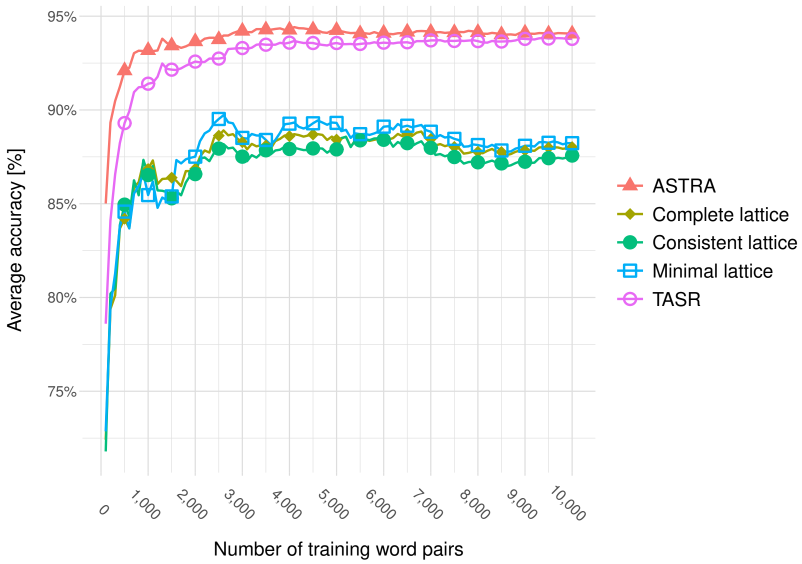


FIGURE 2.3: Accuracy of the transformation engine models

Thesis 2 ***[2] [3] [9] [10] [11] [12] [13] [14] [15] [16]***

I have proposed two novel single-affix transformation engine models that can learn inflection rules from a provided set of training word pairs. The first one is a lattice based model that has a more complex, position dependent rule structure, and stores its rules in a lattice. The second model called ASTRA describes inflection as a set of simple string transformations, omitting the position indices from its rule model. The atomic rules are stored in either a set or a prefix tree based data structure. Both models apply pattern matching during the rule search process. I performed the evaluation of the proposed models, showing that while the lattice based model can achieve minimal storage size, the ASTRA model has an outstanding accuracy for previously unseen words (about 94%), beating the examined baseline models including TASR, FST and a dictionary implementation.

TABLE 2.1: The test results using
10 thousand training word pairs

Model	Training	Size	Search	Accuracy
Sequential ASTRA	237 ms	65,894	15 ms	94.06%
Parallel ASTRA			2 ms	
Prefix tree based ASTRA	472 ms		32 μs	
Complete lattice	330 s	14,404	73 μs	88.03%
Consistent lattice	406 s	14,072	220 μs	87.57%
Minimal lattice	503 s	2,412	43 μs	88.23%
TASR	79 s	55,849	1.5 ms	93.79%
FST	125 ms	20,541	1.5 μs	0%
Dictionary	10 ms	9,978	232 ns	0%

2.3 Multi-Affix Morphology Model

To solve the multi-affix inflection generation and morphological analysis, I proposed the Morpher model, whose main components can be seen in Figure 2.4:

- Transformation engines: low-level transformation engine models that are capable of learning the transformations of a single affix type based on a set of training word pairs. (The ASTRA and the lattice based models can be used.)
- Probability store: the set of conditional probabilities among all the known affix types of the target language.
- Lemma store: the store of lemmas and their associated parts of speech.

To solve the multi-affix inflection generation and morphological analysis, the Morpher model coordinates the work of these components.

In case of agglutinative languages and concatenative morphology, it is important that a word might have several different morphological structures and inflected word forms generated by the same affix

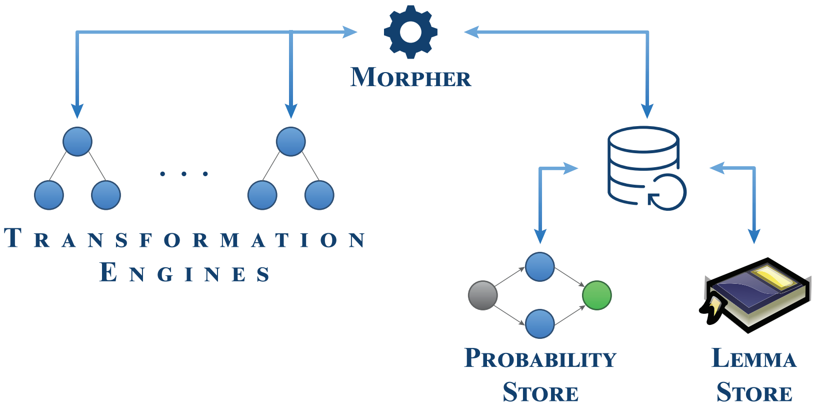


FIGURE 2.4: The main components of the Morpher model

type. Therefore the Morpher model has been designed so that it can return multiple responses for an input, and it also can provide a confidence value for each answer. Another important observation is that the words of the target language can be represented as dots in a vector space, where the basis is the set of lemmas, and every other word form can be reached from lemmas, applying a number of affix types.

The Morpher model can be trained using a $\mathfrak{T} = \left\{\left(w, \bar{w}, \bar{T}, \langle T_i \rangle\right)\right\}$ set, where $w \in W$ are the words, $\bar{w} \in \bar{W}$ are the lemmas, $\bar{T} \in \bar{\mathbb{T}}$ are the parts of speech and $T \in \mathbb{T}$ are the affix types, while $\langle\rangle$ denotes a list. During the training phase, the Morpher model executes the following steps:

1. It creates and trains a separate transformation engine (ASTRA) instance for each affix type with an appropriate word pair set deduced from the original training data.
2. It calculates the conditional probability of each affix type chain found in the training data.
3. It stores the lemmas and their parts of speech.

During inflection generation, the Morpher model does the following things:

1. It determines the possible orders of the input affix types based on the previously calculated conditional probabilities.

2. It has the appropriate transformation engine instances transform the input lemma one by one, until it reaches the final inflected word form.
3. It calculates an aggregated weight for every response, that serves as a confidence value.
4. It sorts the responses based on their aggregated weight in a descending order.

If $\mathcal{FC}^{E_T}$ denotes the conversion operator of the transformation engine model during inflection generation, then the inflection operator of the Morpher model can be defined as the following multi-valued function:

$$\mathcal{I} : \bar{W} \times \{T_i\}_{i=1}^{m} \to \left\langle \left(\bar{T}_{i_0},\ \left\langle S_{i_j}^{\mathcal{I}} \right\rangle_{j=1}^{m},\ \vartheta_i \right) \right\rangle_{i=1}^{n} \tag{2.10}$$

where

$$S_{i_j}^{\mathcal{I}} = \left(T_{i_j},\ \mathcal{FC}^{E_{T_{i_j}}} \left(w_{i_{j-1}} \right) \right) \tag{2.11}$$

and ϑ_i denotes the aggregated weight of the ith response.

According to this definition, the input of the inflection operator of the Morpher model is a lemma and a set of m affix types, while its output is n responses. These responses all contain a part of speech, $m-1$ intermediate word forms with their related affix type, and an mth final word form, as well as an aggregated weight.

During morphological analysis, the Morpher model executes the following steps:

1. It determines which affix types may appear in the input word.
2. It has the appropriate transformation engine instances transform the input word one by one backwords, until it reaches a valid lemma.
3. It calculates an aggregated weight for every response, that serves as a confidence value.
4. It sorts the responses based on their aggregated weight in a descending order.

If $\mathcal{BC}^{E_T}$ denotes the backwards conversion operator of the transformation engine model during morphological analysis, then the morphological analysis operator of the Morpher model can be defined as the following multi-valued function:

$$\mathcal{A} : W \to \left\langle \left(\left\langle S_{i_j}^{\mathcal{A}} \right\rangle_{j=m_i}^{1},\ \bar{T}_{i_0},\ \vartheta_i \right) \right\rangle_{i=1}^{n} \tag{2.12}$$

where

$$S_{i_j}^{\mathcal{A}} = \left(T_{i_j},\ \mathcal{BC}^{E_{T_{i_j}}}\left(w_{i_j}\right) \right) \tag{2.13}$$

and ϑ_i denotes the aggregated weight of the ith response.

According to this definition, the input of the morphological analysis operator of the Morpher model is an inflected word form, while its output is n responses containing a part of speech, $m_i - 1$ intermediate word forms with their related affix type, and the m_ith final word form (the lemma), as well as an aggregated weight.

It is important to note that morphological analysis is a more complex operation than inflection generation, since we do not know in advance which or even how many affix types there are in the input word form.

To evaluate the Morpher model, I compared it with several existing models found in literature: 6 SIGMORPHON models (Helsinki 2016,[7] UF 2017,[8] UTNII 2017,[9] Hamburg 2018,[10] IITBHU 2018[11] and MSU 2018[12]), 3 segmentation models (Morfessor 2.0,[13] MORSEL[14] and MorphoChain[15]), as well as 2 morphological analyzers (Lemming[16] and Hunmorph-Ocamorph[17]).

The training phase of the Morpher model using 100 thousand training records took less than 4 seconds, which is a good result among the examined models.

[7]https://github.com/robertostling/sigmorphon2016-system
[8]https://github.com/valdersoul/conll2017
[9]https://github.com/hajimes/conll2017-system
[10]https://gitlab.com/nats/sigmorphon18
[11]https://github.com/abhishek0318/conll-sigmorphon-2018
[12]https://github.com/AlexeySorokin/Sigmorphon2018SharedTask
[13]https://github.com/aalto-speech/morfessor
[14]https://github.com/ConstantineLignos/MORSEL
[15]https://github.com/karthikncode/MorphoChain
[16]http://cistern.cis.lmu.de/lemming
[17]http://mokk.bme.hu/en/resources/hunmorph

Regarding the size of the exported file containing the knowledge base, the Morpher model reached 5th place, with 8.5 megabytes. Although Morfessor 2.0 had the smallest exported file, it is only a segmentation model, therefore its knowledge base can omit some information about the training data set. There were also three models that produced much larger files than Morpher: Hunmorph-Ocamorph, Helsinki 2016 and UTNII 2017.

The inflection time of Morpher proved to be the fastest among the examined models, while its morphological analysis time tied for second with the segmentation time of MORSEL using 100 thousand training records. However, the analysis time of Morpher produced a steeper curve.

Figure 2.5 displays the average accuracy of the examined models. As we can see, Morpher reached the highest precision with more than 97%. The SIGMORPHON models were very close to it, except for the Hamburg 2018 model. Finally, the segmentation models reached much lower accuracy.

Since Morpher can return multiple responses for a single input, it is important to check how many responses it produces in average, and where is the expected (correct) answer in the response list based on the aggregated weight. During inflection generation, the average number of responses became 37, while during morphological analysis it became 5.4. Although these numbers can seem high, the expected response was in average at the index of 1.5 and 2.4, respectively, meaning that the correct response was at the beginning of the list, with high confidence.

To measure the generalization capability of the examined models, I tested them with generated artificial words as well. First, I took real syllables from meaningful Hungarian words and combined them to produce the base artificial word forms. Then, I transformed them empirically using the transformation rules of the Hungarian accusative case. Finally, I checked if the examined models can handle these artificial word pairs correctly. Only the Helsinki 2016 (41%), Morfessor 2.0 (80%) and Hunmorph-Ocamorph (89%) models could transform some of these words correctly, however, the Morpher model reached the highest accuracy with 95%.

In the final evaluation test, I used the Hungarian data sets provided by SIGMORPHON to train and evaluate the Morpher model.

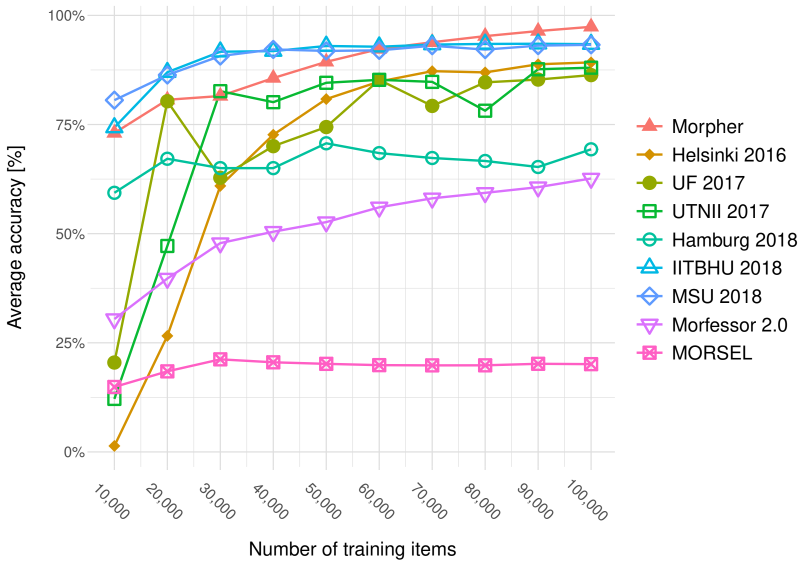


FIGURE 2.5: The average accuracy of the examined models

Table 2.2 contains the test results. As we can see, the model could reach outstanding accuracy using the training sets of high information, which dropped to around 50-60% using the medium and low information data sets.

Thesis 3 ***[3] [4]***

I have proposed a novel multi-affix morphology model called Morpher that can solve the inflection generation and morphological analysis problems, handling all the affix types of the target language. The main feature of the proposed Morpher model is that it builds a separate transformation engine instance for each affix type, and it takes the conditional probabilities of the affix type chains into account during inflection generation and morphological analysis. During the

TABLE 2.2: The accuracy of Morpher using the data sets provided by SIGMORPHON

Data set	Accuracy
SIGMORPHON 2016	98.03%
SIGMORPHON 2017 low	49.64%
SIGMORPHON 2017 medium	54.40%
SIGMORPHON 2017 high	95.14%
SIGMORPHON 2018 low	53.69%
SIGMORPHON 2018 medium	59.92%
SIGMORPHON 2018 high	95.43%

evaluation of Morpher I used the ASTRA model to train the transformation engines. The experiments confirmed the outstanding generalization capabilities and accuracy of Morpher, comparing it with state of the art models including 6 SIGMORPHON models, 3 unsupervised segmentation models and 2 analyzer models.

2.4 Complexity Analysis and Optimization of Morpher and ASTRA

The following components contribute to the space requirements of the Morpher and ASTRA models:

- the stored conditional probabilites whose number equals to the affix types found in the training data set
- the stored lemmas and their parts of speech whose number can be approximated with the size of the training data, as well as the number of parts of speech in the target language
- the trained transformation engines whose number equals the number of affix types in the target language

For every ASTRA instance, a training word pair set is generated, whose size can be approximated with the number of training records whose affix type list ends with the associated affix type. From these word pairs, atomic rules are constructed. For every word pair, the number of constructed atomic rules depends on the word length and the length of the transformed substring.

Based on the space complexity approximations, we can also approximate the time complexity of the inflection generation and morphological analysis operations. From these approximations it can be seen that during morphological analysis, the Morpher model needs to check much more possible cases, since it does not know how many or what kind of affix types can be found in the input word.

The goal of the optimization techniques is to reduce the size of the rule base by eliminating rules that do not contribute much to the high accuracy, so that the time requirements of the inflection generation and morphological analysis are reduced in case of the same training data set size, while retaining high average accuracy. In the dissertation I proposed three optimization techniques that are based on new configuration parameters of the proposed models:

- Eliminating the redundant atomic rules: the p_{max} parameter determines how many atomic rules should be generated at most for each word pair during the training phase of the ASTRA model.
- Limiting the generalization factor: the p_{gen} parameter determines the minimum context length of the generated rules. The atomic rules with shorter context are eliminated.
- Indirect noise reduction: using the p_{supp} and p_{freq} parameters will drop those atomic rules whose *support* and *word frequency* values[18] are lower than the parameter values.

I analyzed these optimization parameters empirically, examining how their values change the previously measured metrics (e.g. accuracy, number of responses, index of expected response). The smallest rule base was achieved using the combination of p_{gen} and p_{max}, specifically $p_{gen} = p_{max} = 1$. This observation was not a surprise, as I proved formally that in this case, the rule base does not contain any redundant rules. However, although the rule base size is optimal using this configuration, the information content of the system reduces significantly, increasing its uncertainty.

[18]The *support* value of a rule is equal to the number of training words that are covered by the rule, while its *word frequency* value is equal to the number of occurrences of these words in the source free texts.

The winner configuration became $p_{supp} = 10$. Using this parameter value the rule base reduced to less than 2% of the original one, retaining high accuracy. The average inflection and morphological analysis times decreased significantly as well. Moreover, in case of inflection generation, the average number of responses dropped from 37 to 8.4, which is also an important result.

Using the $p_{supp} = 10$ configuration, I compared the original baseline, non-optimized Morpher model with the optimized one. During the evaluation, using 100 thousand training records, both the average inflection and analysis times decreased. Moreover, as Figure 2.6 shows, the analysis time curve became less steep. The original 2.4 seconds of average analysis time became a less more than 20 milliseconds using the optimized model.

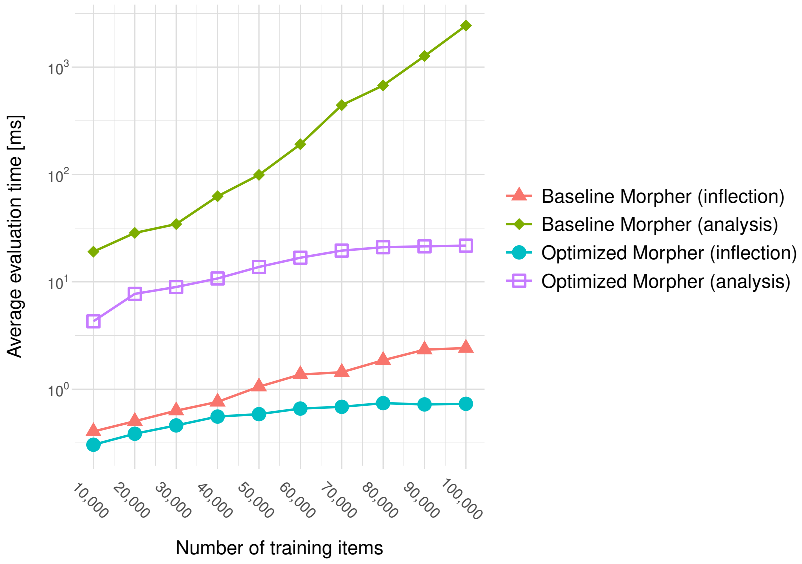


FIGURE 2.6: The average inflection and analysis times of the baseline and the optimized Morpher model

After comparing the baseline and optimized Morpher model, I increased the training data size up to 3 million. The original, non-optimized Morpher model could not handle such data sizes, but the optimized model could perform the training, inflection generation and morphological analysis in acceptable finite time. This is due to the time curves being less steep. However, from the test results we can see that after 1 million training items, further increasing the training data size does not have real benefit, as the accuracy stagnates.

Thesis 4 *[3] [4] [5]*

I have performed the space and time complexity analysis of the Morpher and ASTRA models. After analyzing the required space of the model components and the required time of the training phase, inflection generation and morphological analysis operations, I have proposed three optimization techniques that aim to reduce the rule base of the model and thus decrease the average inflection and analysis times. Evaluation shows that using the same amount of training data, the number of retained rules dropped from 578,497 to 10,291, the average inflection time decreased from 2.4 ms to 0.7 ms, and the average analysis time was reduced from 2.4 s to 21.75 ms, while the average accuracy remained 93.01%. Finally I evaluated the optimized Morpher model using up to 3 million training items, and both inflection generation and morphological analysis could be performed in acceptable, finite time, even though the unoptimized original model could not handle such big training data volumes at all.

2.5 The Reference Implementation of the Morpher Ecosystem

The proposed models including the Hungarian data sets were published on Github.

The training and test data sets were generated from the documents of the Hungarian Electronic Library.[19] From the 16,250 documents I extracted unique word candidates using a parallel algorithm, then

[19]`https://mek.oszk.hu`

used Hunmorph-Ocamorph to determine the morphological structures of these word candidates. Using this automated method, I managed to extract more than 4.4 million morphological structures for more than 2.5 million unique word forms.

The obtained records were applicable to train the Morpher model, however the training of the single-affix transformation engine models I needed to get training word pairs relating to a single affix type, thus this data set had to be processed further. To optimally deduce the appropriate word pairs, I did not process every single record pair, only those that had the same lemma. This way the time of data processing was reduced significantly. Moreover, those records that only contained a single affix type, could be processed on their own, since the base form and the inflected form could be used as a word pair.

The total number of word pairs that can be found in the Github repository[20] is 3,625,036.

The reference implementation of the proposed models including all the source code can be found in three Github projects:

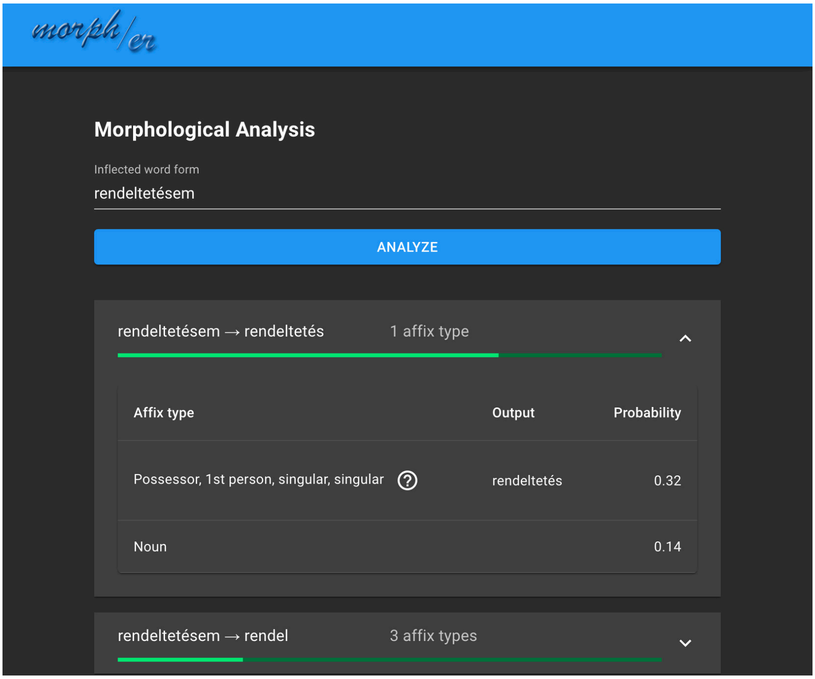
- Morpher framework:[21] a Java based, modular project that contains the reference implementation of the Morpher model, the lattice based model and the ASTRA model, as well as the other baseline single-affix transformation engine models.
- Morpher API:[22] a Spring Boot based REST API application that simplifies the usage of Morpher from software environments other than Java, like .NET or Node.js.
- Morpher client:[23] a React and React Native based client application that provides a graphical user interface for the inflection generation and morphological analysis operations of the Morpher framework, invoking the Morpher API.

The Morpher framework has been published on Maven Central[24] and jcenter[25]. The Docker images have also been published on Docker

[20]`https://github.com/szgabsz91/morpher-data`
[21]`https://github.com/szgabsz91/morpher`
[22]`https://github.com/szgabsz91/morpher-api`
[23]`https://github.com/szgabsz91/morpher-client`
[24]`https://search.maven.org/search?q=morpher`
[25]`https://bintray.com/search?query=morpher`

Hub, including the Docker image of the Morpher API[26] and the Morpher web client.[27]

The morphological analysis page of the Morpher web client can be seen in Figure 2.7.



Morphological Analysis

Inflected word form: rendeltetésem

rendeltetésem → rendeltetés — 1 affix type

Affix type	Output	Probability
Possessor, 1st person, singular, singular	rendeltetés	0.32
Noun		0.14

rendeltetésem → rendel — 3 affix types

FIGURE 2.7: The morphological analysis page of the Morpher web client

Thesis 5 ***[1] [3] [4] [5] [8]***

I have developed the reference implementation of the Morpher ecosystem, that consists of several layers such as the Morpher multi-affix

[26]`https://hub.docker.com/r/szgabsz91/morpher-api`
[27]`https://hub.docker.com/r/szgabsz91/morpher-client`

morphology model and the single-affix transformation engine models including the lattice based model and ASTRA. I have also developed a Spring Boot based Morpher REST API, so that the Morpher framework can be consumed more easily from different software environments. To provide web and mobile user interfaces for inflection generation and morphological analysis, I have developed the Morpher client application using React and React Native. For training and evaluation purposes, I have also implemented an automated method to generate large training and evaluation data sets for the Hungarian language, resulting in more than 4.4 million morphological structures, covering more than 2.5 million unique word forms. The source code of these projects can be found on Github, while the binaries are published to jcenter and Maven Central, and the Docker images are published to Docker Hub.

3. Összefoglalás

A kutatás célja egy olyan mintaillesztésen alapuló morfológiai modell kidolgozása volt, amely képes morfológiailag komplex, agglutináló nyelvek (pl. a magyar nyelv) esetén automatizáltan megtanulni a célnyelv toldalékolási szabályait, és hatékonyan megoldani a ragozás és morfológiai elemzés problémáját.

A bemutatott morfológiai modellek két fő csoportba sorolhatók:

- az alacsonyabb szintű, egytoldalékos transzformációs modellek (a hálóalapú és az ASTRA modell) a célnyelv egy adott toldaléktípusának szabályait képesek betanulni egy megfelelő szópárhalmaz alapján, míg
- a magasabb szintű, többtoldalékos Morpher modell a célnyelv összes toldaléktípusát képes kezelni.

A modelleket a kiértékelés közben számos létező alapmodellel vetettem össze, és a bemutatott modellek mind pontosságban, mind az általánosítóképesség terén kiemelkedőnek bizonyultak.

Az új tudományos eredmények a következő öt tézisben foglalhatók össze.

1. tézis *[1]*

Létrehoztam egy új módszert morfológiai elemzők összehasonlítására, elemzésére és rangsorolására. A módszer alapját olyan formulák képezik, melyekkel kiszámolhatjuk az elemzők hasonlóságát és távolságát, figyelembe véve azok szófelismerési képességeit, tokenrendszereit és leképzési tulajdonságait. Az elkészült módszert négy népszerű, magyar nyelvvel is kompatibilis morfológiai elemzőn alkalmaztam, a Hunmorph-Ocamorph, Hunmorph-Foma, Humor és Hunspell eszközökön. A mérések kivitelezéséhez elkészítettem egy token leképzést is a négy elemző között. Az elvégzett mérések alapján a Hunmorph-Ocamorph modell bizonyult a legjobban használható morfológiai elemzőnek.

2. tézis *[2] [3] [9] [10] [11] [12] [13] [14] [15] [16]*

Létrehoztam két új transzformációs modellt, melyek képesek megtanulni a szükséges toldalékolási szabályokat egytoldalékos esetben, az adott toldaléktípusra jellemző szópárok alapján. Az első egy hálóalapú modell, amely egy komplexebb, pozíciófüggő szabálystruktúrával rendelkezik, és ezeket a szabályokat egy hálóban tárolja. A második modell (ASTRA) a toldalékolási szabályokat egyszerű string transzformációkként írja le, elhagyva a pozícióindexeket a szabályleíróból. Ezeket az atomi szabályokat egy halmaz vagy prefixfa alapú adatstruktúrában tárolja. Mindkét modell a mintaillesztés elvét alkalmazza az illeszkedő szabályok keresésekor. A modellek kiértékelése azt mutatja, hogy míg a hálóalapú modell képes minimális tárigényt elérni, addig az ASTRA modell pontossága kiemelkedő a betanított modell által még nem látott, ismeretlen szavak esetén (kb. 94%), megelőzve a TASR, FST és szótár alapú modelleket.

3. tézis *[3] [4]*

Létrehoztam egy új morfológiai modellt, amely többtoldalékos esetben is képes a toldalékolás és morfológiai elemzés megtanulására és elvégzésére, lefedve a célnyelv összes toldaléktípusát. A Morpher modell alapja, hogy minden egyes toldaléktípushoz külön ASTRA példányt hoz létre a transzformációs szabályok betanulásához, toldalékolás és morfológiai elemzés közben pedig figyelembe veszi a toldaléktípus láncok feltételes valószínűségeit. A modell kiértékelése megerősíti annak kiemelkedő általánosítóképességét és pontosságát számos korszerű modellel szemben, beleértve 6 SIGMORPHON modellt, 3 szegmentációs modellt és 2 morfológiai elemzőt.

4. tézis *[3] [4] [5]*

Elvégeztem a Morpher és ASTRA modellek tár- és időkomplexitásának elemzését. A különböző komponensek tárigényének, valamint a betanítás, a toldalékolás és a morfológiai elemzés időigényének vizsgálata után három optimalizációs technikát vezettem be, melyek célja, hogy a transzformációs szabályok eliminálásával csökkentsék az átlagos toldalékolási és elemzési időt. A kiértékelés megmutatta, hogy az optimalizált modell azonos méretű tanítóhalmaz mellett az eddigi 578 497 szabály helyett csak 10 291 szabályt generált, így az átlagos toldalékolási idő 2.4 milliszekundumról 0.7 milliszekundumra csökkent, az átlagos elemzési idő 2.4 másodpercről 21.75 milliszekundumra esett vissza, míg az átlagos pontosság 93,01% maradt. A bevezett optimalizációnak köszönhetően a Morpher modell képessé vált elfogadható, véges időn belül elvégezni a toldalékolást és morfológiai elemzést akár 3 milliós tanítóminta betanítása után is, amire az eredeti modell még nem volt képes.

5. tézis *[1] [3] [4] [5] [8]*

Elkészítettem a Morpher rendszer referenciaimplementációját, amely magában foglalja a Morpher többtoldalékos morfológiai modellt és az egytoldalékos transzformációs modelleket, köztük a hálóalapú modellt és az ASTRA modellt. A Morpher keretrendszer különböző szoftverkörnyezetekből történő kényelmesebb felhasználásához egy Spring Boot alapú Morpher REST API-t is fejlesztettem. A böngészőben, Android és iOS eszközökön történő felhasználás érdekében pedig elkészítettem a Morpher kliens alkalmazást, React és React Native alapokon. A modellek kiértékeléséhez egy automata betanítási és tesztadat generáló módszert dolgoztam ki, amely több mint 4,4 millió morfológiai felbontást eredményezett, lefedve több mint 2,5 millió egyedi szóalakot. Ezen projektek forráskódja megtalálható a Githubon, míg a binárisokat a jcenter és Maven Central repositorykba, a Docker image-eket pedig a Docker Hubra publikáltam.

Author's Publications

[1] Gábor Szabó and László Kovács. Benchmarking morphological analyzers for the Hungarian language. In *Annales Mathematicae et Informaticae*, volume 49, pages 141–166. Eszterházy Károly University Institute of Mathematics and Informatics, 2018. **Q3**. SJR: 0.157.

[2] G. Szabó and L. Kovács. Lattice based morphological rule induction. *Acta Universitatis Apulensis*, (53): 93–110, 2018.

[3] László Kovács and Gábor Szabó. String transformation based morphology learning. *Informatica*, 43 (4): 467–476, December 2019. **Q4**. SJR: 0.178.

[4] Gábor Szabó and László Kovács. Automated learning of hungarian morphology for inflection generation and morphological analysis. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 8 (4): 746–756, December 2020. **Q4**. SJR: 0.168.

[5] Gábor Szabó and László Kovács. Optimization of the Morpher morphology engine using knowledge base reduction techniques. *Computing and Informatics*, 38 (4): 963–985, 2019. **Q3**. SJR: 0.217. Impact Factor: 0.524.

[6] László Kovács and Gábor Szabó. Utilizing Apache Hadoop in clique detection methods. *Production Systems and Information Engineering*, 7: 43–53, 2015.

[7] László Kovács and Gábor Szabó. Conceptualization with incremental Bron-Kerbosch algorithm in big data architecture. *Acta Polytechnica Hungarica*, 13 (2), 2016. **Q2**. SJR: 0.298. Impact Factor: 1.219. Independent citations: 7.

[8] Gábor Szabó. Efficient method for training set generation in morphological analysis. In *Doktoranduszok Fóruma - Gépészmérnöki és Informatikai Kar szekciókiadványa*, November 2014.

[9] G. Szabó and L. Kovács. Efficiency analysis of inflection rule induction. In *Proceedings of the 2015 16th International Carpathian Control Conference (ICCC)*, pages 521–525, May 2015.

[10] Gábor Szabó. Grammatical rule generation strategies for concept lattice based inflection systems. In *Doktoranduszok Fóruma - Gépészmérnöki és Informatikai Kar szekciókiadványa*, pages 5–10, November 2015.

[11] László Kovács and Gábor Szabó. String transformation approach for morpheme rule induction. *Procedia Technology*, 22: 854–861, 2016.

[12] Gábor Szabó. Edit distance based grammatical rule generation using an improved cost function. In *The Publications of the MultiScience - XXX. microCAD International Multidisciplinary Scientific Conference*, pages 1–8, University of Miskolc, Hungary, April 2016.

[13] Gábor Szabó. Lattice-based ruleset representation for morpheme analysis. In *Doktoranduszok Fóruma - Gépészmérnöki és Informatikai Kar szekciókiadványa*, November 2016.

[14] László Kovács and Szabó Gábor. Generalization of string transformation rules using optimized concept lattice construction method. *Procedia Engineering*, 181: 604–611, 2017. SJR: 0.286. Independent citations: 1.

[15] Gábor Szabó. Morphological models for natural languages. In *XX. Tavaszi Szél Konferencia 2017*, March 2017.

[16] Gábor Szabó. Computational models for morphology. In *The Publications of the MultiScience - XXXI. microCAD International Multidisciplinary Scientific Conference*, pages 1–8, University of Miskolc, Hungary, April 2017.

References

[Bir40] Garrett Birkhoff. *Lattice theory*, volume 25. American Mathematical Soc., 1940.

[DlH10] Colin De la Higuera. *Grammatical inference: Learning automata and grammars*. Cambridge University Press, 2010.

[GW12] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: Mathematical foundations*. Springer Science & Business Media, 2012.

[LCMY09] Constantine Lignos, Erwin Chan, Mitchell P Marcus, and Charles Yang. A rule-based unsupervised morphology learning framework. In *CLEF (Working Notes)*, 2009.

[Lev66] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, February 1966.

[Lig10] Constantine Lignos. Learning from unseen data. In Mikko Kurimo, Sami Virpioja, and Ville T. Turunen, editors, *Proceedings of the Morpho Challenge 2010 Workshop*, pages 35–38, Helsinki, Finland, September 2–3 2010. Aalto University School of Science and Technology.

[MCFS15] Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274, 2015.

[NBJ15] Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, 3:157–167, 2015.

[Öst16] Robert Östling. Morphological reinflection with convolutional neural networks. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 23–26, Berlin, Germany, August 2016. Association for Computational Linguistics.

[PK99] Gábor Prószéky and Balázs Kis. A unification-based approach to morpho-syntactic parsing of agglutinative and other (highly) inflectional languages. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 261–268. Association for Computational Linguistics, 1999.

[PT93] Gábor Prószéky and László Tihanyi. Humor: High-speed unification morphology and its applications for agglutinative languages. *La tribune des industries de la langue*, 10:28–29, 1993.

[SA17] Hajime Senuma and Akiko Aizawa. Seq2seq for morphological reinflection: When deep learning fails. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 100–109, Vancouver, August 2017. Association for Computational Linguistics.

[SF07] Ksenia Shalonova and Peter Flach. Morphology learning using tree of aligned suffix rules. In *ICML Workshop: Challenges and Applications of Grammar Induction*, 2007.

[SKBK18] Fynn Schröder, Marcel Kamlot, Gregor Billing, and Arne Köhn. Finding the way from ä to a: Sub-character morphological inflection for the SIGMORPHON 2018 shared task. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 76–85, Brussels, October 2018. Association for Computational Linguistics.

[SKS18] Abhishek Sharma, Ganesh Katrapati, and Dipti Misra Sharma. IIT(BHU)–IIITH at CoNLL–SIGMORPHON 2018 shared task on universal morphological reinflection. In

Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection, pages 105–111, Brussels, October 2018. Association for Computational Linguistics.

[Sor18] Alexey Sorokin. What can we gain from language models for morphological inflection? In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 99–104, Brussels, October 2018. Association for Computational Linguistics.

[THR$^+$06] Viktor Trón, Péter Halácsy, Péter Rebrus, András Rung, Péter Vajda, and Eszter Simon. Morphdb.hu: Hungarian lexical database and morphological grammar. In *LREC*, pages 1670–1673. Citeseer, 2006.

[TKG$^+$05] Viktor Trón, András Kornai, György Gyepesi, László Németh, Péter Halácsy, and Dániel Varga. Hunmorph: Open source word analysis. In *Proceedings of the Workshop on Software*, pages 77–85. Association for Computational Linguistics, 2005.

[VSGK13] Sami Virpioja, Peter Smit, Stig-Arne Grönroos, and Mikko Kurimo. Morfessor 2.0: Python implementation and extensions for Morfessor Baseline. Technical report, 2013.

[ZLL17] Qile Zhu, Yanjun Li, and Xiaolin Li. Character sequence-to-sequence model with global attention for universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 85–89, Vancouver, August 2017. Association for Computational Linguistics.