MorphoDeep: deep learning for Hungarian morphology
MorphoDeep: deep learning magyar morfológiára

Judit Ács*
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
judit@aut.bme.hu

Abstract

We present three types of experiments pertaining to Hungarian morphology. The first type is the classification of part-of-speech tags, the second type is the classification of frequent inflectional paradigms. Lastly, we use clusters of morphological tags as classes. All of our experiments apply feed forward neural networks and recurrent neural networks. Our data is the Hungarian Webcorpus including morphological analysis by a rule-based analyzer.

Ebben a munkában háromféle, magyar morfológiai jelenségeket vizsgáló kísérletek eredményeit mutatjuk be. Az első típus a szófaji címkék osztályozása, a második néhány gyakori ragozás vizsgálata, míg a harmadik a morfológiai elemzések előzetes klaszterezése utáni osztályozás. Mindegyik kísérlethez előrecsatolt, illetve rekurrens neurális hálókat alkalmazunk. A kísérletek adatait a Magyar Webcorpus morfológiailag elemzett változata szolgáltatja.

1 Introduction

Morphology is the study of word structure, how words are formed and how they are related to each other. Traditionally linguists define *morpheme* as the smallest unit that has semantic meaning on its own. Morphological analysis is the process of segmenting words into smaller units, called *morphemes* and analysing their syntactic and semantic roles.

Morphemes are either *bound* or *free*. Bound morphemes may only appear as parts of a larger word, while free or unbound morphemes stand on their own.

Bound morphemes are further categorized into *derivational* and *inflectional* morphemes. The former category consist of morphemes that change the grammatical category or the meaning of the word, while the latter category only affects plurality, tense, case and so on.

Morphological analysis plays a central role in many natural language processing tasks such as information retrieval, machine translation or spell checking. However, the large variation of morphological paradigms accross different languages makes it especially hard to create general solutions for morphological analysis. For example Chinese or Vietnamese are both *isolating languages* i.e. they almost exclusively use free morphemes, therefore analysis below word level is seldom hard or necessary. In contrast, Uralic languages such as Hungarian exhibit rich morphology, and the number of unique word forms is basically infinite. English – by far the most well resourced language – lies somewhere in the middle, therefore most general solutions do not focus as much on morphology as for example Hungarian NLP would necessitate.

*http://avalon.aut.bme.hu/ judit

Table 1: Frequency of the top 10 POS tags in 1 million words

POS	Frequency
NOUN	311259
PUNCT	162867
VERB	105032
ART	100766
ADJ	85076
ADV	67105
CONJ	55539
NUM	19446
POSTP	16480
PREV	12691

In this work, we shall only examine bound morphemes and mostly inflectional paradigms based on word forms in Hungarian.

2 Related work

Morphology is one of the last areas of natural language processing, where rule based systems still heavily outperform machine learning systems. Particularly, the unsupervised learning of morphology is very hard to perform. Perhaps the most significant attempt at unsupervised segmentation and basic analysis is *Morfessor* [3, 4]. Morfessor uses Hidden Markov Models to model simple morphology (prefix-stem-suffix).

[6] and [8] give an overview of the unsupervised learning of morphology before the era of deep learning.

Morphology has been receiving a lot more attention from the deep learning community and this year a shared task was organized by SIGMORPHON'16. [2] The task was morphological reinflection in a high number of languages including Hungarian. The top performers all used deep learning, the best one [9] outperforming others with a 10% margin in certain languages.

Other recent papers on the supervised learning of morphological paradigms include [5] and [1].

3 Classification problems

Our experiments can be grouped into three categories. All experiments use a single word and its analysis as a sample, no context is taken into account. Our training data is the Hungarian Webcorpus [7], which includes morphological analysis by HunMorph [10]. We consider the output of HunMorph – a rule-based analyzer – to be very reliable and treat the data as a high quality silver standard. Character n-grams were extracted as features.

3.1 Classifying part-of-speech tags

Part of speech is a category of words that display similar grammatical properties. The assignment of part of speech categories to words is called *part of speech tagging* or POS tagging. POS tagging is usually a crucial task in NLP and high quality tools are necessary. We investigate how well POS tags can be inferred from word forms using the output of HunMorph which includes POS tags. POS tagging can be considered as a baseline task of morphological analysis, since high quality morphological analysis includes and extends high quality POS tagging.

The number of different POS tags is very limited, only 10 categories take up at least 1% of the words (see Table 1).

Since punctuation (`PUNCT`) and article (`ART`) are *closed* categories – i.e. their members can be easily listed and the list is not very long –, they do not amount to an interesting classification problem and most of our experiements aim to distinguish between nouns and verbs. In Hungarian this is an almost trivial task for any competent speaker and we expect a learning algorithm to perform well too.

Table 2: Bigram features using the last 6 characters. The sample word is *nevetés*.

Feature (start index)	Value
-1	é$
-2	és
-3	té
-4	et
-5	ve
-6	ev

3.2 Inflectional paradigm classification

In Hungarian syntactic and semantic roles of words are most frequently marked with suffixes. Inferring inflectional paradigms from word forms without context is also an easy task for a human and should be easy for a learning algorithm too. We chose a few frequent inflectional paradigms for classification:

singular vs. plural both verbs and nouns can be singular and plural and we make no distinction between verbs and nouns in this case,

first person vs. second person whether a verb is first or second person (conjugation),

accusative vs. other noun cases among the dozens of noun cases, accusative is the most common,

past tense vs. present tense verb tenses regardless of the person,

conditional verbs vs. others verbs in conditional or other case,

most frequent noun cases 10 most frequent noun cases.

3.3 Classification of tag clusters

Full morphological analysis is represented via a complicated coding, called *KR coding*, which contains among others POS, case, tense and plurality. The number of unique tags in the first million words is about 1 700. The large number of different tags call for some kind of aggregation such as clustering.

Brown clustering is a hierarchical clustering method that uses a mutual information criteria for merging two clusters. As a hierarchical clustering method, Brown clustering eventually results in one big cluster unless a stopping criteria is specified. For our experiments we limit the number of clusters to 120. It is important to note that only the tags are used during clustering and word forms are ignored.

4 Feature extraction

The corpus contains a single word and its analysis per line and sentence boundaries are denoted with an empty line. Words and KR codes are tab separated. Since the corpus is already denoised and tokenized, the only preprocessing we use is lowercasing.

Our assumption is that the end of the word is more implicative of its grammatical category than its beginning, due to the agglutinative (rich in suffixes) nature of Hungarian. Character n-grams are used as features starting from the end of the word. The position of the n-grams are also preserved (i.e. this is not a bag-of-ngrams model). Table 2 shows an example of feature extraction.

Padding characters are added to the beginning of the words if they are shorter than the number of features extracted. The last character of the word is usually especially important, therefore $N - 1$ padding characters are added to the end of the word. When using recursive neural networks, we exclude position information.

5 Experimental setup

Each experiment involves 4 steps:

1. feature extraction: Webcorpus is read from the beginning until enough samples are collected,
2. training: train on 90% of the samples (split into 80% train and 20% validation set),
3. testing: test on 10% of the samples,
4. logging: save the experiment to a pandas DataFrame.

5.1 Architecture

The first architecture tested was a simple feed forward neural network. Since the length of the input is limited to 5-10 characters, we limit the depth of the network to two hidden layers (to be fair, three hidden layers were found not to improve the results while significantly increase training time), and the number of neurons-per-layer to 100. Grid search like optimization is performed in these boundaries.

Secondly recurrent neural networks were used with varying number of cells.

5.2 Technical background

For training and testing neural networks we use Keras with Tensorflow backend. FFNN experiments were run in a Jupyter notebook, while RNN experiments were implemented as standalone scripts. The reason for this is that the codebase is under heavy development and constant refactoring with the eventual goal of creating an experiment framework as a Python package.

Every experiment is logged into a pandas DataFrame and saved to disk in a tab-separated file. FFNN and RNN experiments log into separate DataFrames and both files are versioned. Each line represents a single experiment with all its parameters and results. Due to the large number of parameters, the tables are not easy on the human eyes, and a human readable version with fewer lines is also extracted and saved. GitHub natively renders tsv as tables.

6 Parameter optimization and results

Feature extraction involves several parameters:

last_char the number of characters starting from the end of the word, to be used as features,

N the size of n-grams etracted,

include_smaller_ngrams include $1 \ldots N$ grams in the features,

use_padding use padding characters at the beginning and the end of words,

sample_per_class how many samples-per-class are to be extracted.

Setting `include_smaller_ngrams` and `use_padding` `True` were found to help during the first half of the experiments and for the sake of simplicity were kept that way later on. The number of training samples is only limited by the available GPU memory. 60 000 samples split into 90% train and 10% test can be easily trained on our system. We did experiment with more samples, but the gain in accuracy was not substantial. Although there are ways to avoid loading everything into memory, we have not experimented with them yet.

Model hyperparameters include the number of hidden layers, the number of neurons in the hidden layers, the number LSTM and GRU cells, the learning rate and the optimizer's parameters such as the loss function. Training parameters include the number of epochs and the batch size. The training data was split into 80% training and 20% validation and early stopping was performed using the latter. Each layer applied sigmoid activation function.

6.1 POS classification

Most of our experiments were run on noun vs. verb, so we shall go into detail regarding this case.

6.1.1 NOUN vs. VERB classification

The highest test accuracy we achieved using a feed forward network, was 96.73% using trigram features from the last 10 characters.

Table 3: Highest test accuracy for inflectional paradigms.

POS	FFNN	RNN
adverb, adjective	97.6	96.8
conjunction, noun	99.1	98.3
adjective, conj, noun, verb	85	87.43

Since recurrent neural networks are able to ”memorize” previous inputs, we can feed the samples as sequences of characters and expect the network to infer n-gram like features. We experiment with LSTM and GRU and the 95.61% using GRU with 125 cells and limiting the input sequence to the last 9 characters. Although the best configuration was achieved using GRU, it is not at all clear whether LSTM or GRU are better suited for this task. Figure 6.1.1 demonstrates the very small difference between the results and other comparisons are available in the repository.

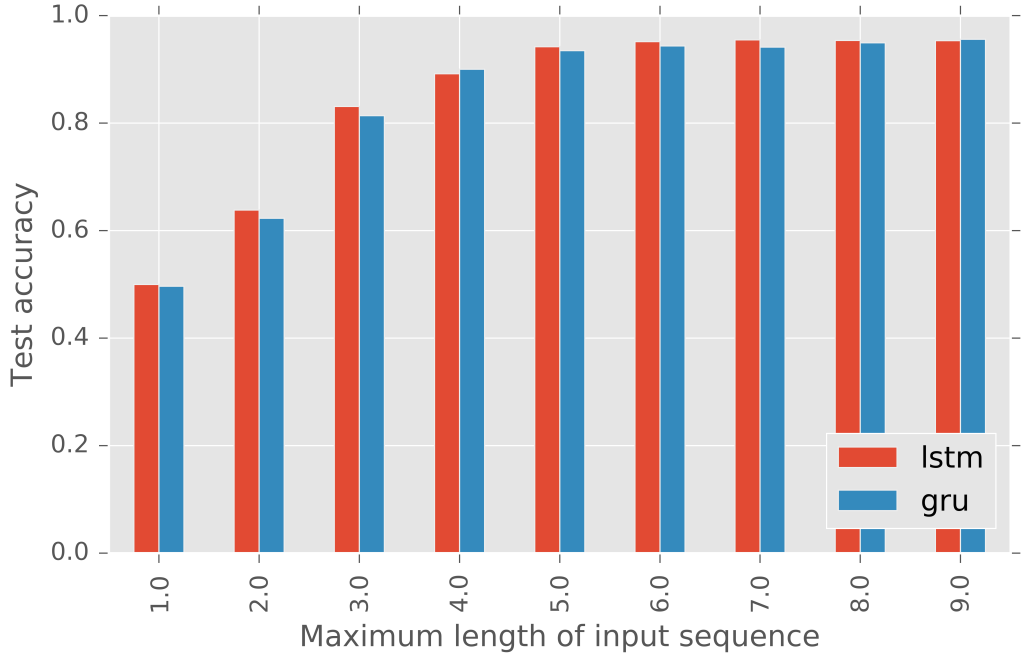

Figure 1: LSTM vs. GRU against varying sequence length

6.1.2 Other part-of-speech categories

A few experiments were run on other category combinations, see Table 3 for the results.

6.2 Inflectional paradigms

We run the same experiments on a selected few paradigms that occur frequently in Hungarian. The results are listed in Table 4.

6.3 Classifying tag clusters

The significance of this experiment group is twofold: (i) the number of classes is considerably higher than previously, (ii) the clusters were created with no regard to word forms. If a word-based classifier is able to learn these clusters, the clustering does convey some kind of word information.

For these experiments we need additional preprocessing, namely we have to convert KR codes to cluster ids using an already existing mapping. The distribution of tags in a cluster is very far from

Table 4: Highest test accuracy for inflectional paradigms.

Paradigm	FFNN	RNN
singular/plural	99.7	99
first/second person	98.9	99.6
accusative/non-accusative	99.8	99.6
past tense/present tense	99.4	98.1
conditional verbs	99.9	100
10 noun cases	99.3	99.3

uniform, with a few tags dominating each class. To avoid learning single tags instead of clusters, we limit the number of times a tag (mapped to a cluster id) may appear in the training data.

By limiting the number of tags-per-cluster to 25 and requiring 100 samples, the classifier is able to achive close to 60% accuracy. When we do not limit the tags-per-cluster, accuracy reaches 70% on 120 classes.

7 Conclusion and future work

Our experiments show that all three tasks are relatively easy to solve with simple neural network architectures and a modest number of hidden layers and neurons. While FFNN are very simple, the obvious advantage of RNN is that raw character features can be fed to the network instead of fiddling with n-grams. One of the next steps would be to interpret the features learned by the network.

The eventual goal of this research project is the unsupervised discovery of morphological paradigms. Our study suggests that once these paradigms are labeled, it is very easy to distinguish them from each other. However, the biggest obstacle is the large number of paradigms and their uneven distribution, which makes the data very sparse.

Although we do have high quaility rule-based analyzers for Hungarian, this is not true for many other languages. A possible application of this line of research would be to extend it to other languages such as low-density Uralic languages with rich morphology.

For the machine learning background, only very simple approaches were compared since the task itself did not necessitate more sophisticated solutions. We do plan to apply other neural network architectures later on for automatic segmentation and rule discovery.

References

[1] Malin Ahlberg, Markus Forsberg, and Mans Hulden. Paradigm classification in supervised learning of morphology. In *Proc. of NAACL*, 2015.

[2] Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. The sigmorphon 2016 shared task—morphological reinflection. In *Proceedings of the 2016 Meeting of SIGMORPHON*, Berlin, Germany, August 2016. Association for Computational Linguistics.

[3] Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proc. 6th SIGPHON*, pages 21—30, 2002.

[4] Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3, 2007.

[5] Greg Durrett and John DeNero. Supervised learning of complete morphological paradigms. 2013.

[6] John A. Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, 2001.

[7] Péter Halácsy, András Kornai, László Németh, András Rung, István Szakadát, and Viktor Trón. Creating open language resources for Hungarian. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 203–210. ELRA, 2004.

[8] Harald Hammarström and Lars Borin. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350, 2011.

[9] Katharina Kann and Hinrich Schütze. Med: The lmu system for the sigmorphon 2016 shared task on morphological reinflection. *ACL 2016*, page 62, 2016.

[10] Viktor Trón, György Gyepesi, Péter Halácsky, András Kornai, László Németh, and Dániel Varga. Hunmorph: Open source word analysis. In *Proceedings of the ACL Workshop on Software*, pages 77–85. Association for Computational Linguistics, Ann Arbor, Michigan, 2005.