UNIVERSITY OF MISKOLC
FACULTY OF MECHANICAL ENGINEERING AND COMPUTER SCIENCE AND ENGINEERING

Efficiency Analysis of Inflection Rules Geneartion in Hungarian

Summary of PhD Thesis

Written by
Zsolt Tóth
MSc in Engineering Information Technology

JÓZSEF HATVANY DOCTORAL SCHOOL FOR INFORMATION SCIENCE, ENGINEERING AND TECHNOLOGY

Scientific Advisor
Dr. habil. László Kovács

Miskolc
2014

Contents

1 Introduction

Natural languages are often classified by various aspects such as their history, word order or morphology. In historical linguistics the languages are organized into language families. There is an ancestor descendant relationship in a family, so a language family is often represented as a tree. The root of the tree is called proto-language such as Indo-European, Uralic, Caucasian, American, Austroasiatic, and Sino-Tibetan languages. These families contain other subgroups. For example the Indo-European languages can be classified into Anatolian, Tocharian, Germanic, Italic, Celtic, Armenian, Balto-Slavic, Hellenic, Indo-Iranian and Albanian subgroups. These subgroups can be also divided into subcategories. Most of the spoken languages in Europe is Indo-European such as English and German are Germanic, Spanish, France and Italian are Italic, and Slovakian and Polish are Balto-Slavic languages, but Hungarian is a Uralic language [Hel03].

In linguistic typology the languages are classified by their structural features such as word order. Subject, Object and verb are distinguished in typology and the languages are classified by the order of subject, object and verb. There are nine typological classes plus a category for free word order languages. The vast majority of the languages belong to Subject-Object-Verb or Subject-Verb-Object category [Mey10].

In morphology the languages are classified by how the words are formed from morphemes. Analytic or isolating languages and synthetic languages are distinguished in morphology. The morpheme per word ratio is low in isolating languages. The meaning of the words depends on their position and auxiliary words are used to express complex concepts. Chinese and English are isolating languages. In synthetic languages the morpheme per language ratio is high in contrast to the isolating languages. Words are formed by affixing the base morpheme. Synthetic languages are often subdivided into polysynthetic, fusional, and agglutinative languages. Words in polysynthetic languages are long and their meaning could be a whole sentence in other languages. Some Native American languages are polysynthetic. Fusional languages have some common aspects with analytic languages because generally the words have lost their inflections over the centuries. Many Indo-European languages belong to this group. Agglutinative languages, such as Japanese, Hungarian or Esperanto, form words by combining the base word with phonetically unchangeable affixes. The main difference between agglutinative and fusional languages is that there are more affixes in agglutinative languages and these affixes can be separated from the base word and each other.

Hungarian is an agglutinative language and belongs to the Uralic language

family. The word order is not strict and it is mostly used to emphasize the content. Inflection is used to modify the meaning of a word. An inflected form of a word is called a case. There are 17-28 different cases in Hungarian which depends on the analysis [Mor03]. A chain of inflection is defined in Hungarian because of two reasons. Firstly the affixes are put in the end of the word in most of the cases. Secondly an inflected word can be the base word of another inflection. For instance in the sentence "Peter has refused our calls" the "our calls" is the object and it is plural. This sentence in Hungarian is "Péter visszautasította a hívásainkat" where the objects is „hívásainkat". The affixes can be separated as "(hívás)(aink)(at)" where „hívás" is the stem, „aink" means „our" that the word is in plurarl and „at" denotes it is an object.

However there are serious efforts taken in the field of the text mining and natural language processing, but these are only a few works which focus on Hungarian. Moreover the most of these works focuses on Natural Language Processing tasks such as stemming [HKN+04, RG] or learning morphemes [Dud06]. On the other hand my researches were focused on not the stemming, but the inflection of Hungarian words. Inflection can be considered as a string transformation which is used in many natural languages and it also plays an important role in Hungarian too.

In the text books there are abstract inflection rules which are marked out for human understanding. To generate complex text in natural languages, these abstract inflection rules have to be converted into a more specific form which can be used by computer programs. This conversion is a difficult and costly task. There are a few researches on the generation of inflection rules, but these works are focused on other languages.

My researches were also focused on the induction of Formal Grammars which are often used to model the structures natural languages. The theory of formal grammars was laid in the mid 1950s by Noam Chomsky [Cho56] [Cho59]. Formal grammars are often classified by their production rules into these four categories: *Regular*, *Context–Free*, *Context–Sensitive* and *Recursive Enumerable Grammars*. However some algorithms was developed in the 1950s and 1960s the time–cost of these algorithms has exceeded the computational power of the time. Many induction algorithm was created, implemented and tested from 2000. My researches was focused on the time–cost of the Context–Free Grammar induction methods.

There are is lack of the comparison of Context–Free Grammar induction methods. A common environment was required to compare the different induction algorithms. However there are some data mining frameworks [HDW94, HFH+09, WFT+99, FHH+05], tools [Min], but only few standards [OAS, Apab] and educational softwares [Gru] focus on text mining and gram-

mar processing. Hence I design META Framework was designed to provide a common environment for grammar induction methods. Extensibility was one of the most important requirement of the framework, thus new grammar induction methods can be added to the system easily. META makes possible to create more accurate comparison of these methods. Some well–known Context–Free Grammar [SK99, Sak05, UJ07, OU09, NI00, NM02] and Probabilistic Context–Free Grammar induction method have been implemented, tested and compared in the META Framework.

1.1 Research Goals

The main goal of my researches was to analyze the different inflection rule generation methods. Basic classification based methods were evaluated and a possible limit of this approach have been shown. The analysis of the problem domain showed that there are non linear separable classes in the training set of 56,000 Hungarian word pairs. This property limits the efficiency and accuracy of the classifiers.

Analysis and comparison of Context–Free Grammar induction methods was the other important goal of my researches. META Framework was designed and developed for this purpose. Some well–known induction algorithm have been implemented, tested and compared in META Framework. Thus META is capable to be a common grammar induction framework.

2 Scientific Results

This section sums up and presents briefly my scientific results and my thesis with the related publications.

2.1 Complexity Analysis of the Inflection Induction Problem

2.1.1 Phonetic Alphabet

In text mining there are huge efforts to create stemming algorithm [Por01, Dom07]. Stemming can be considered as an inverse function of the inflection because it transforms an inflected word into its base form. Although text mining is an actively investigated field of computer science, most of the papers focus on the English language [ONM01]. But there are efforts to create Hungarian specific text mining tools [HKN$^+$04, RG]. My work was focused on the learning of inflection rules of Hungarian.

The words and their inflected forms are given as strings. Vector space model is a well–known and widely used to represent objects in data mining and knowledge engineering. Moreover there are many formal methods to handle the represented objects. The mapping of the string into vector space model is often made by the conversion of the letters into real values.

There are various mapping functions and these methods are based on different approaches. Alphabets give and historical order of the letters of languages. The order of the words is given by these alphabets, but the numerical values for the letters are not defined exactly. Character tables, like ASCII, assign a number to each character. Although the numerical values are defined in this case, the distances do not represent the similarities of the letters.

The letters encode sounds so they are often characterized by their phonetic features. These features determine the sound and the letter too. In Hungarian one letter denotes only one sound and vice versa in most of the cases [ASA, Zsu]. These phonetic features also can be represented in a vector space. So one letter is encoded by a real vector. Although the phonetic features of the letters is given in textbooks, the numerical encoding of the features requires an expert. The phonetic features are the dimensions of the letters and the dimensional values are determined. The following dimensions are distinguished: *shape of the lips*, *pitch*, *position of the tongue*, *voice*, *way of production* and *place of production*. Thus the letters are represented by six dimensional real vectors. This representation allows to determine the distance of the words. Figure 1 shows the user interface of the MatLab

application which was developed to create phonetic alphabets.

Figure 1: MatLab Application For Hungarian Language

To create a phonetic features based alphabet, the letters from the six dimensional space have to be mapped into a one dimensional. Principal Component Analysis [Jol05] was used to perform the mapping. The points are put into a new coordinate system where each dimension denotes a feature. Then coordinate transformation is performed. In the result coordinate system, the deviation of the points is the greatest in the first coordinate. Principal Component Analysis requires the determination of the eigenvalues and eigenvectors which is a computational costly task. The eigenvalue denotes the cardinality of the dimension thus the dimension with the highest eigenvalue is chosen to create the alphabet and the other dimensions are omitted.

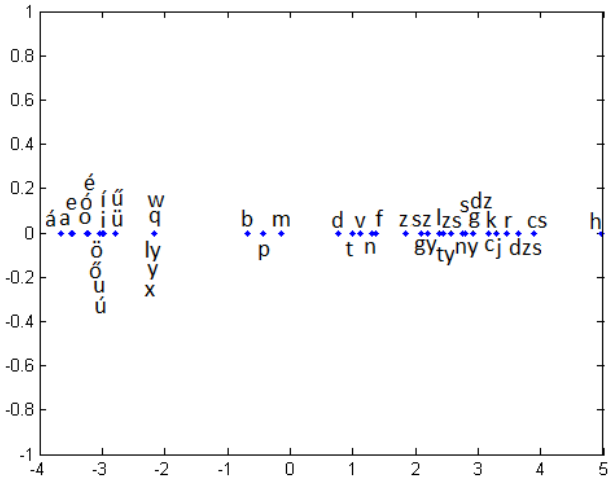


Figure 2: Phonetic features based Hungarian alphabet

Figure 2 shows the yielded alphabet. There are clusters of letters in the

figure and the members of these clusters have similar phonetic features and denote similar sounds. For example, there is a cluster of vowels and there are approximately three clusters of consonants. It also shows the sounds of `b`, `p` and `m` are similar while the sound of `h` significantly differs from the other letters. The letters have a real value in these alphabet which denotes its position. The distance of these letters can be calculated from their codes.

The above presented method was presented in [8] and a yielded phonetic alphabet was used in our further analysis [9].

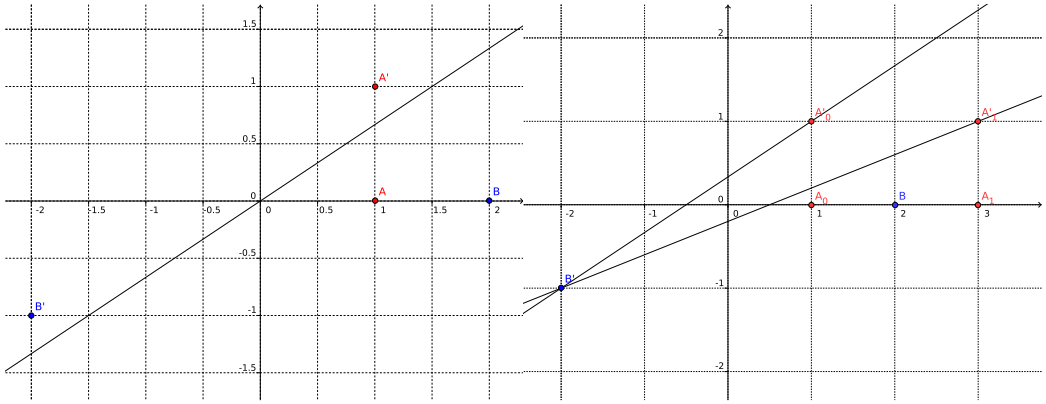
2.1.2 Evaluation of Classification based Learning of Inflection Rules

Inflection is a string transformation which convert a stem into its inflected form. The transformations are described by inflection rules. There is only one valid inflection rule for each word. Thus inflection can be considered as a classification task where the stem is the input and the inflection rule is the category.

There are many different classification algorithms in the literature such as neural networks [Zha00], support vector machines [SV99] or Bayesian [JL95] classifiers. The efficiency of classification, among many factors, depends on the training set. Linear classifiers would be very accurate and efficient but they require that the clusters be linear separable. So there cannot be overlap or intersection between the clusters. There are numerous methods [Eli06] to test linear separability.

Testing of linear separability can be transformed into a linear programming task. The d dimensional points are mapped into a $d+1$ dimensional space. Each vector is augmented by 1. Then one of the clusters are mirrored to the origo. If there is at least one hyperplane $P(\mathbf{w}, t) = \{\mathbf{x} \mid \mathbf{w}^T\mathbf{x} + t = 0\}$ which goes across the origo, then the two clusters are linear separable. Figure 3 shows these transformations in both separable and inseparable cases. Multiple clusters can be separated conical hulls if they are piecewise linear separable [BM94]. The constrains are given by the points and the object function is defined by the hyperplane. This linear program has to be converted into standard minimization problem. Equation 1 shows the transformed constrains matrix and the final optimization problem is shown in Equation 2.

A training set of about 54.000 (stem, inflected form) pairs were used during the test [9]. The training set contains most of the Hungarian nouns thus it gives a good approximation of the .Hungarian language. Although inflection can be chained in Hungarian, the training set and my tests were focuses on only one case. There were about 160 different inflection rules distinguished in the test and about a half of the samples belonged to one specific class. Table 1 shows the 15 biggest categories and their size.



(a) Linear Separable Points (b) Non Linear Separable Points

Figure 3: Examples for linear separability

$$\begin{bmatrix} a_{1,1\_aux\_1} & -a_{1,1\_aux\_2} & a_{1,2\_aux\_1} & -a_{1,2\_aux\_2} & \dots & a_{1,d+1\_aux\_1} & -a_{1,d+1\_aux\_2} \\ a_{2,1\_aux\_1} & -a_{2,1\_aux\_2} & a_{2,2\_aux\_1} & -a_{2,2\_aux\_2} & \dots & a_{2,d+1\_aux\_1} & -a_{2,d+1\_aux\_2} \\ a_{3,1\_aux\_1} & -a_{3,1\_aux\_2} & a_{3,2\_aux\_1} & -a_{3,2\_aux\_2} & \dots & a_{3,d+1\_aux\_1} & -a_{3,d+1\_aux\_2} \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots \end{bmatrix} \tag{1}$$

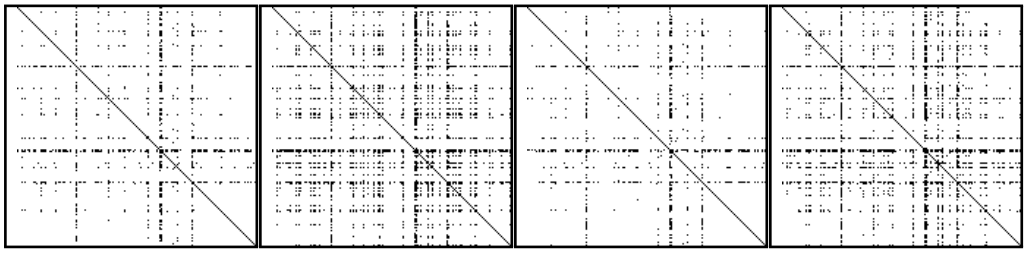
$$\begin{aligned} w_{1\_aux\_1} - w_{1\_aux\_2} + w_{2\_aux\_1} - w_{2\_aux\_2} + \dots + t_1 - t_2 \rightarrow min \\ [\mathbf{w}, t]^T_{trans} \, [\mathbf{A}_{trans}] \geq 1 \\ w_{i\_aux\_1}, w_{i\_aux\_2} \geq 0 \\ t_1, t_2 \geq 0 \end{aligned} \tag{2}$$

Java application has been developed the optimization task which used the Apahce's `commons-math` library [Apaa] to solve the optimization task. Because the optimization was computationally costly thus the size of the clusters was limited. Due to this limitation there can be false positive results in the tests. But there can be no false negative result thus the test can show only that, if two clusters are not linear separable. The test yields a symmetric boolean matrix which contains true if the two clusters are linear separable otherwise false. This matrix is hard to understand and handle so the results are visualized in Figure 4. In the visualization each pixel represent a cluster pair. If they are not linear separable then the point is black, otherwise it is white. So if there are only linear separable clusters then the image shows a white square with a black diagonal line.

Table 1: Number of transformations in categories

Transformation	# of word pairs	% of training set
*(-/t)	28239	52.31
*(-/o)(-/t)	7399	13.71
*(-/e)(-/t)	6731	12.47
*(a/á)(-/t)	5975	11.07
*(-/a)(-/t)	2041	3.78
*(e/é)(-/t)	969	1.80

The four different encodings were used during the tests. The shorter words were padded into the length of the longest word and both left and right padding were used to create the data set. Phonetic and traditional alphabet based encodings were used to map the letters into numerical values. The given results show there are non linear separable cluster pairs in each case which limit the efficiency of classification methods. There are more non linear separable pairs in the case of the right padding. This phenomenon can be explained with that the accusative inflection of Hungarian adds suffix to the word.



(a) Alphabetical encoding – Left (b) Alphabetical encoding – Right (c) Phonetic encoding – Left (d) Phonetic encoding – Right

Figure 4: Linear Separability with Alphabetical and Phonetic encoding

Thesis 1.

I have measured the complexity of inflection rules induction by the ration of the linear separable clusters pairs in the vector space. I have introduced a methods to create phonetic features based alphabet. The created phonetic alphabet based encoding was shown superior to traditional alphabet based encoding.

Related Publications: [8], [9]

2.2 Induction of Inflection Rules with Classification and Associative Memory

The algorithms of Computational Linguistics usually have a common model which can be seen in Figure 5. Morphological analyzers, stemmers and inflection systems usually have two core parts. They contain an engine to perform the transformation on the input word and to produce the output word. The engine has no direct knowledge about the language. The morphological rules are stored in a separate rule set. The structure of the rule set depends on the inflection algorithm. For example Snowball [Por01] is a language to describe stemming rules for Porter stemmer [Por80]. Rules of the SMOR [SFH04] morphological analyzer are given by the Stuttgart Finite State Tools and the engine is realized as Finite State Transducer. Classification based inflection algorithms can use the category to encode the transformation.

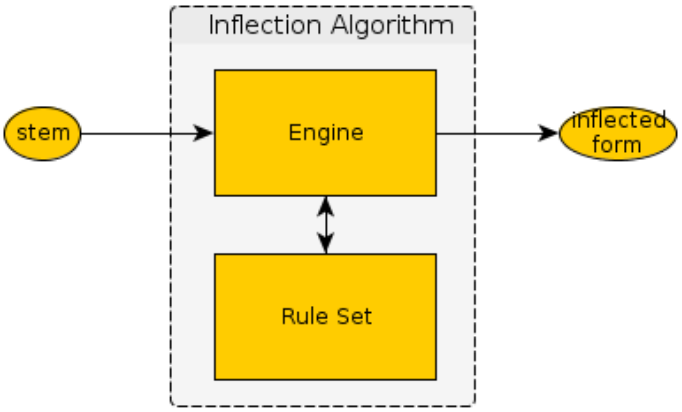


Figure 5: Common Model of Inflection Algorithms

The model of the presented inflection algorithm is showed in Figure 6. The implemented rule set consists of two parts: an associative memory and a classifier. The associative memory stores the transformation rules of irregular words. A word transformation class is considered as irregular, if there are only few words belonging to that class. Formally the $t = \{w\}$ category contains irregular words if $\mid t \mid \leq \epsilon$ where w is a word which belongs to class t and ϵ is a cardinality threshold. Although the associative memory can retrieve the exact transformation string for each learned stem it cannot be used to determine the transformation string of untrained words. The classifier is used to capture the frequent inflection rules. It can perform generalization thus untrained words can be processed. The generalization may easily fail on exception. Considering the classifier systems, they have lower precision than

associative memory and the precision usually depends on the training set too. Moreover they have more difficult learning algorithm which acquires a significant additional learning time cost. In some cases, the classification also can have a significant time cost for each word. The instance based classifiers such as k-NN classifier [CMBT] determines the k most similar object to the classified instance from an instance database. The distance calculation and the search also can be costly thus the inflection algorithm can be slow.

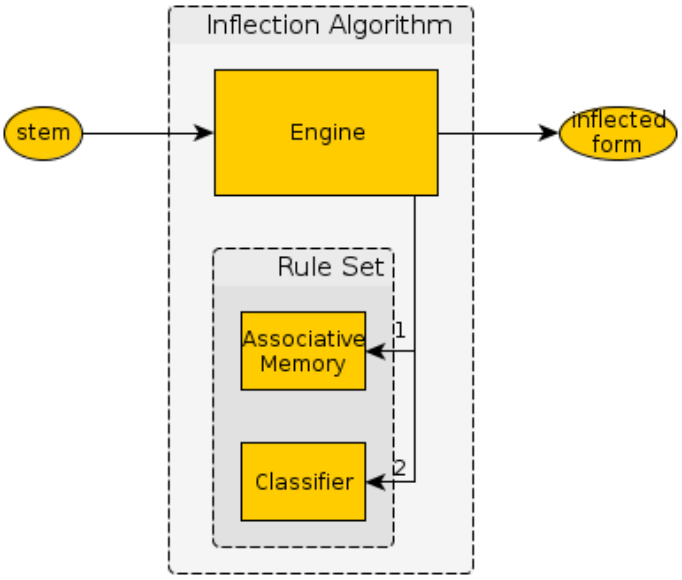


Figure 6: Model of the presented Inflection Algorithm

The rule set determines the behavior of the inflection algorithm so the precision of the algorithm depends on the rule set. During the learning process the rules set is defined as pairs of stem and inflected form. Transformation string can be determined for each word pairs with the Levenshtein distance algorithm [Nav01]. The transformation strings are considered as categories of stems. Naive Bayes [JL95] and Multilayer Preceptron [JMM96] classifiers were used to solve this classification problem.

The presented inflection algorithm uses both classifier and an associative memory to learn inflection rules. Regular words are classified by the classifier and the irregular words are stored in the associative memory. The size of the associative memory is a parameter of the method. The algorithm looks for the word in the associative memory. If the word is not found, then the transformation string is determined by the classifier.

2.2.1 Precision

The precision increases more quickly with the reduced training sets in the cases of both encodings. Figure 7 shows the changes of the precision with the size of the associative memory and the size of the training set. The algorithm showed similar behavior with both Multilayer Perceptron and Naive Bayes classifiers. The precision can be increased with the usage of associative memory. This phenomenon can be explained with the learning algorithm of the classifier. Because the irregular cases are placed in the associative memory, the number of the categories is reduced. If a category, which is not linear separable from other clusters, is put into the associative memory, then the number of the linear non separable cluster pairs decreases. Hence the usage of the associative memory can reduce the number of the linear non separable cluster pairs. This reduction could yield the increase of the precision of the inflection algorithm.

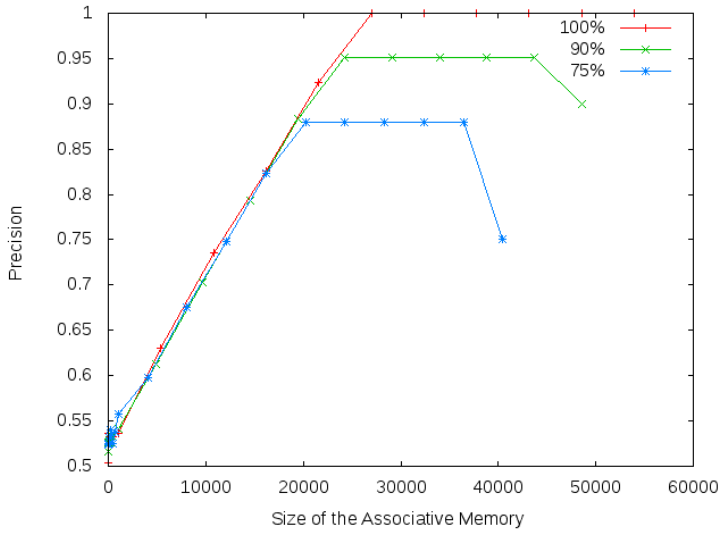


Figure 7: Precision with Different Size of Training Sets

2.2.2 Learning Cost

The time cost of the algorithm is the time which is require for training. Although this cost occurs only in the learning phase it makes the tuning of the algorithm slower. Moreover the learning cost can limit the applicability of the algorithm if it is too high. For example if the time cost would grow exponential with the number of samples then it could be applied with only small training sets.

The measurements showed that the learning cost of the algorithm significantly depends on the classifier. Figure 8 shows how the learning cost decreases with the increase the size of the associative memory. It also shows a short transient phase up to 3.000. Then it decreases steadily and there is a fall around 20.000.

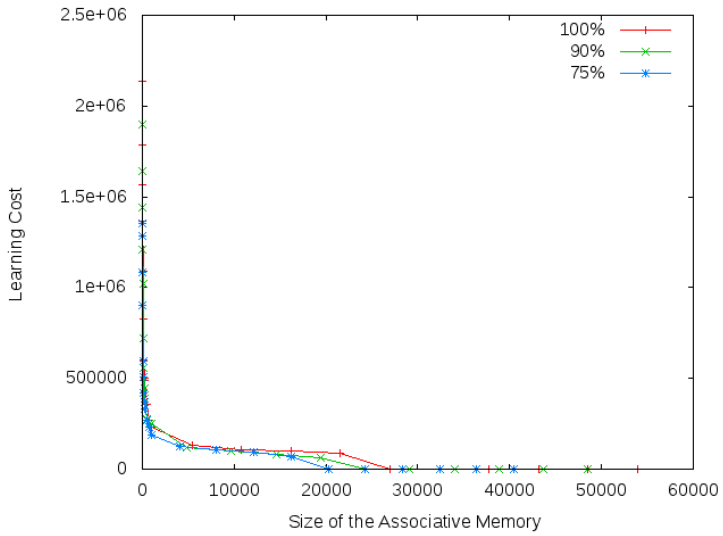


Figure 8: Learning cost with different size of training sets

2.2.3 Size

The size of the learning algorithm was measured as the size of the serialized classification structure in bytes. The serialization was possible with the Java API because the Classifier class of Weka implements the `Serializable` interface. Because the Naive Bayes and the Multilayer Perceptron classifiers had similar behavior with both encodings. Figure 9 shows how the size of the algorithm depends on the size of the associative memory. The measurements showed that the size of the classifier has no significant effect on the size of the inflecting algorithm.

The size of the algorithm decreases quickly until approximately 1000 then it starts to increase steadily. The fall can be explained with the simplification of the classification algorithm. Because the least frequent category is put into the associative memory first, many of the categories can be eliminated from the training set even with a small associative memory. This elimination yields a simplified classifier which requires less memory. Then the size of the classification structure increases linearly because the size of the associative

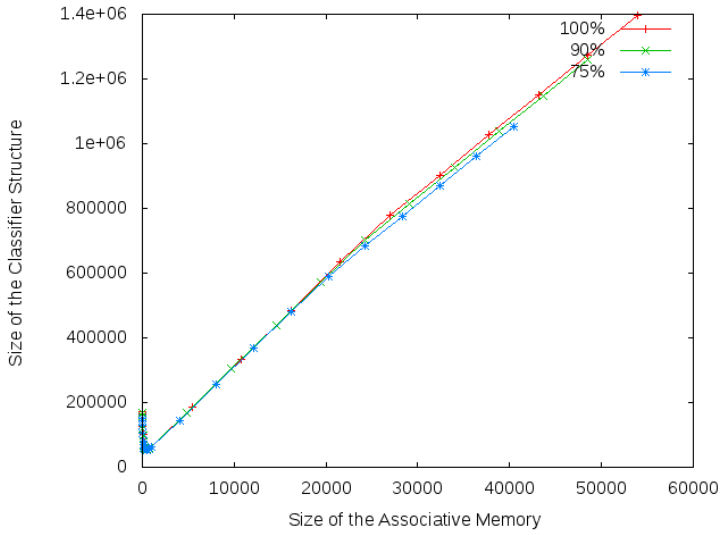


Figure 9: Size of the classification structure with different size of training sets

memory also grows linearly. So linear connection between the size of the algorithm and the size of the associative memory can be assumed.

Thesis 2.
I have presented a classification based inflection method enhanced with associative memory. The algorithm has been shown superior to the standard classification based inflection algorithms from the point of view of precision.
Related Publications: [11], [10], [12]

2.3 META Framework

Theory of formal language is a widely researched are of computer science and mathematics. Formal languages can be given by enumeration or generative grammar. The mathematical basis of generative grammars was laid by Chomksy [Cho56, Cho59] in the late 1950s. He had defined four types of formal grammar which are "Regular Grammars", "Context–Free Grammars", "Context–Sensitive Grammars" and "Recursive Enumerable Grammars". It is still the most widely used classification of formal grammars. In spite of the early theoretical researches and results, these topic has gained popularity in the early 2000s.

A formal grammar is given as a tuple $\mathcal{G} = \langle \mathcal{T}, \mathcal{N}, \mathcal{P}, \mathcal{S} \rangle$. Where $\mathcal{T}$ denotes the set of terminal symbols $(a, b, c, \dots)$. Terminal symbols are valid words of the language and they cannot be replaced by other symbols. The set of non–terminal symbols are denoted by $\mathcal{N}$ and the capital letters $(A, B, C, \dots)$ denote the non–terminal symbols. These symbols are replaced in the derivation process. The next element of the tuple $\mathcal{P}$ stands for the set of production rules which defines the replacements in the grammar. Rules can be formalized as $\alpha \to \beta$ where $\alpha \in \{\mathcal{T} \cup \mathcal{N}\}^* A \{\mathcal{T} \cup \mathcal{N}\}^*$ and $\beta \in \{\mathcal{T} \cup \mathcal{N}\}^*$. Finally $\mathcal{S} \subseteq \mathcal{N}$ is a non–empty subset of non–terminal symbols which is called the set of sentence or start symbols. The derivation process can start with an element of $\mathcal{S}$. Each sentence of the language can be derived from one of the sentence symbols $\mathcal{L} = \{\omega \mid S \to^*_{\mathcal{G}} \omega, S \in \mathcal{S}\}$.

Generative grammars can be generated in manual or automatic way. The manual definition of a grammar is a costly task which requires an expert and it is time–consuming. The automatic generation of formal grammars is called grammar induction. My researches was focused on the induction of Context–Free Grammars which is an $\mathcal{NP}$–hard problem. It was an actively investigated area of computer science in the last decade thus there are many different methods in the literature [DFG11]. These grammar induction methods were implemented in various programming languages and they are based on different heuristic approaches and using various programming techniques to solve the problem. Due to these variety of these methods, their comparison is a hard task. META Framework was design to provide a common environment to implement, test and compare different grammar processing methods.

2.3.1 Architecture

META is a modular framework where each module has a single well–defined responsibility. It is designed to model formal grammars and it defines inter-

faces to extend it with grammar processing and induction methods. META is implemented in Java because Java is platform independent and popular programming language.

I designed the architecture of META framework and I broke it down into modules with a well–defined functionality [5]. Figure 10 shows the package structures of the META Framework. Three main modules can be distinguished in this figure. Core module contains basic, common functions which are used by other modules. Text mining modules which have interfaces for different text mining tasks such as filtering or text analyzing. This module contains only interfaces but no implementations. Grammar processing module provides basic classes to model formal grammars and interfaces for parsing and induction methods.

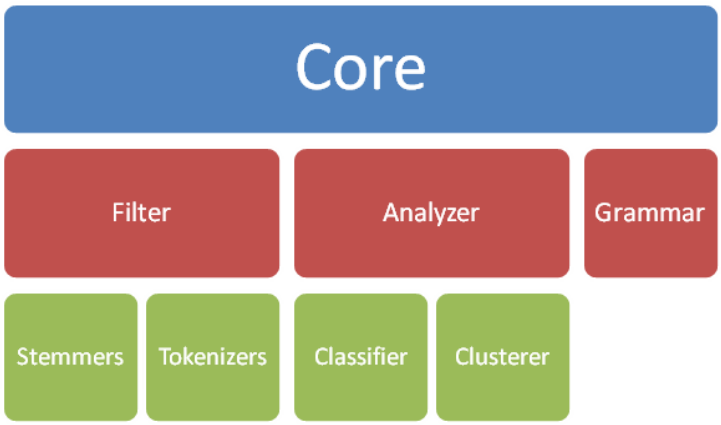


Figure 10: Modules of META Framework

2.3.2 Grammar Processing Module

The grammar processing module of META provides classes to model formal grammars and interfaces for induction and parsing methods. The packages of grammar module are showed in Figure 11. Based on the mathematical definition three packages has been created to organize the grammar processing related classes [6, 7]. These packages are the `grammar`, `grammar.symbols` and `grammar.probabilistic` [1] (see Figure 11). Terminal and non–terminal symbols are contained by the `grammar.symbols` package.

The `grammar` package contains classes to model formal grammars and interfaces for parser and induction methods. `Grammar` class is generic and its parameter is a `GrammaticalRule` type so the grammar object can work with probabilistic or non–probabilistic rules. Thus probabilistic and non–probabilistic grammar can be distinguished in the framework. I defined in

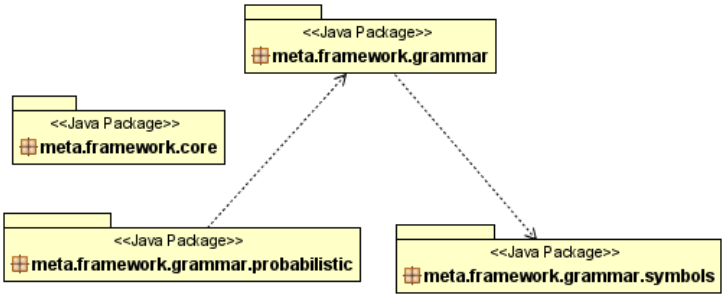


Figure 11: Packages of Grammar Processing Module

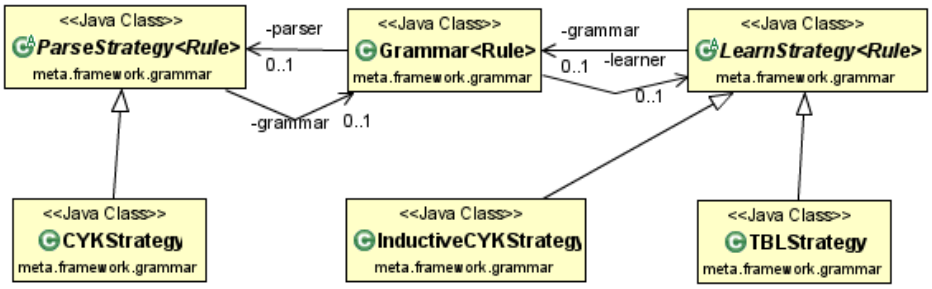


Figure 12: Grammar class with learning and parsing strategies

`LearnStrategy` and `ParseStrategy` abstract classes only interfaces for learning and parsing algorithms thus META can be easily extended by parser and induction methods. These interfaces are based on the Strategy design pattern [GHJV94] which is a common way to change the behavior of an object in run time. These abstract classes are also generics and their generic types have to match with the grammars generic type i.e. if a `Grammar` object works with probabilistic rules then it can use only probabilistic parser and learning strategies. The concrete algorithms are not generics and they defines their generic parameter during the inheritance.

Figure 12 shows the `Grammar` class and its relationship with `LearnStrategy` and `ParseStrategy` classes. I have implemented and tested the CYK parsing algorithm [2] as a `ParseStrategy` strategy and Inductive CYK [NI00, NM02] and TBL [SK99, Sak05] algorithms as `LearnStrategy` [4, 3]

Thesis 3.

I have designed and developed the META framework to provide an environment for implementation, testing and comparison of different natural language processing methods. My experiments showed that both inflection and formal grammar processing methods can be implemented in the META framework

Related Publications: [2], [3], [4], [5], [6], [7]

3 Further Tasks

My further goals are to evaluate the efficiency and accuracy of different linear and non–linear classifiers in the learning of inflection rules, and to develop a heuristic classifier which suits for the inflection. The evaluation is based on some well–known classification methods. For the testing the Weka data mining framework will be used.

The novel heuristic method is based on a conflict relationship between the categories which is based on the observation of that some inflection rules can be used instead of other rules however it results false forms. The conflict relationship is a partial order so there are chains of categories. It allows to create a concept lattice based classifier.

4 Summary

Thesis 1.

I have measured the complexity of inflection rules induction by the ration of the linear separable clusters pairs in the vector space. I have introduced a methods to create phonetic features based alphabet. The created phonetic alphabet based encoding was shown superior to traditional alphabet based encoding.

Related Publications: [8], [9]

Thesis 2.

I have presented a classification based inflection method enhanced with associative memory. The algorithm has been shown superior to the standard classification based inflection algorithms from the point of view of precision.

Related Publications: [11], [10], [12]

Thesis 3.

I have designed and developed the META framework to provide an environment for implementation, testing and comparison of different natural language processing methods. My experiments showed that both inflection and formal grammar processing methods can be implemented in the META framework

Related Publications: [2], [3], [4], [5], [6], [7]

References

[Apaa] Apache commons–math user guide. `http://commons.apache.org/proper/commons-math/userguide/index.html`. Accessed: 2014-09-05.

[Apab] Apahce. Uima. `http://uima.apache.org/index.html`.

[ASA] Raátz Judit Antalné Szabó Ágnes. *Magyar nyelv és kommunikáció Tankönyv 5-6. évfolyam számára.* Nemzeti Tankönyvkiadó.

[BM94] Kristin P Bennett and Olvi L Mangasarian. Multicategory discrimination via linear programming. *Optimization Methods and Software*, 3(1-3):27–39, 1994.

[Cho56] Noam Chomsky. Three models for the description of language. *Information Theory, IRE Transactions on*, 2(3):113–124, 1956.

[Cho59] Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959.

[CMBT] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. Gate: an architecture for development of robust hlt applications hamish cunningham, diana maynard, kalina bontcheva, valentin tablan department of computer science university of sheffield.

[DFG11] Arianna D'Ulizia, Fernando Ferri, and Patrizia Grifoni. A survey of grammatical inference methods for natural language learning. *Artif. Intell. Rev.*, pages 1–27, 2011.

[Dom07] Tikk Domokos. *Szövegbányászat.* Typotex Kft, 2007.

[Dud06] László Dudás. Morfémák megtanulása szövegből. In *MicroCAD 2006 International Scientific Conference*, pages 61–66, Miskolc, 2006. University of Miskolc.

[Eli06] David Elizondo. The linear separability problem: Some testing methods. *Neural Networks, IEEE Transactions on*, 17(2):330–344, 2006.

[FHH+05] Eibe Frank, Mark Hall, Geoffrey Holmes, Richard Kirkby, Bernhard Pfahringer, Ian H Witten, and Len Trigg. Weka. In *Data Mining and Knowledge Discovery Handbook*, pages 1305–1314. Springer, 2005.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software.* Pearson Education, 1994.

[Gru] Zeph Grunschlag. Authored software. `http://www.cs.columbia.edu/~zeph/software.html`.

[HDW94] Geoffrey Holmes, Andrew Donkin, and Ian H Witten. Weka: A machine learning workbench. In *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pages 357–361. IEEE, 1994.

[Hel03] Eugene Helimski. Areal groupings (sprachbünde) within and across the borders of the uralic language family: A survey. *Nyelvtudományi közlemények*, 100:156–167, 2003.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[HKN+04] Péter Halácsy, András Kornai, László Németh, András Rung, István Szakadát, and Viktor Trón. Creating open language resources for hungarian. In *LREC*, 2004.

[JL95] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, 1995. Morgan Kaufmann.

[JMM96] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[Jol05] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[Mey10] Charles F Meyer. *Introducing English Linguistics International Student Edition*. Cambridge University Press, 2010.

[Min] Rapid Miner. Home page. `http://rapid-i.com`.

[Mor03] Edith Moravcsik. Inflectional morphology in the hungarian noun phrase: A typological assessment. *Noun phrase structure in the languages of Europe*, 20, 2003.

[Nav01] Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

[NI00] Katsuhiko Nakamura and Takashi Ishiwata. Synthesizing context free grammars from sample strings based on inductive cyk algorithm. 2000.

[NM02] Katsuhiko Nakamura and Masashi Matsumoto. Incremental learning of context free grammars. 2002.

[OAS] OASIS. Unstructured information management architecture. `http://docs.oasis-open.org/uima/v1.0/uima-v1.0.html`.

[ONM01] Kemal Ofazer, Sergei Nirenburg, and Marjorie McShane. Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computational Linguistics*, 27(1):59–85, 2001.

[OU09] Marcin Jaworski Olgierd Unold. Learning context-free grammar using improved tabular representation. *Applied Soft Computing*, 2009.

[Por80] Martin F Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[Por01] Martin Porter. Snowball: A language for stemming algorithms, 2001.

[RG] Farkas Richárd and Szarvas György. Statisztikai alapú tulajdonnév-felismerő magyar nyelvre.

[Sak05] Yasubumi Sakakibara. Learning context-free grammar using tabular representation. *Pattern Recognition*, 2005.

[SFH04] Helmut Schmid, Arne Fitschen, and Ulrich Heid. Smor: A german computational morphology covering derivation, composition and inflection. In *LREC*, 2004.

[SK99] Yasubumi Sakakibara and Mitsuhiro Kondo. Ga-based learning of context-free grammars using tabular representation. 1999.

[SV99] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[UJ07] Olgierd Unold and Marcin Jaworski. Improved tbl algorithm for learning context-free grammar. 2007.

[WFT+99] Ian H Witten, Eibe Frank, Leonard E Trigg, Mark A Hall, Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with java implementations. 1999.

[Zha00] Guoqiang Peter Zhang. Neural networks for classification: a survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 30(4):451–462, 2000.

[Zsu] Hajas Zsuzsa. *Magyar nyelv 9. osztály.* Pedellus Kiadó.

Author's Publications

[1] László Kovács, László Grigger and Zsolt Tóth, *Induction of PCFG trees*, microCAD International Scientific Conference, 2010

[2] Zsolt Tóth, László Kovács *Cost Analysis of Grammar Induction with CFG* , XXV. microCAD International Scientific Conference 2011.

[3] Zsolt Tóth, László Kovács *Előredukció alkalmazása a TBL algoritmus időköltségének csökkentésére* , Miskolci Egyetem, Doktoranduszok Fóruma 2011.

[4] Zsolt Tóth, László Kovács *Applying Prereduction to Reduce the Tim Cost of TBL Algorithm* , 12th IEEE International Symphosium on Computational Intelligence and Informatics, pp. 544-546, 2011.

[5] Zsolt Tóth, László Kovács *META¿ a novel grammar induction and text mining framework* , XXVI. microCAD, 2012.

[6] Zsolt Tóth, László Kovács *CFG Extension for META Framework* , 16th International Conference on Intelligent Engineering Systems, pp. 495-500 2012.

[7] Zsolt Tóth *Formális nyelvtani modul a META keretrendszer számára* , GÉP: A Gépipari Tudományos Egyesület folyóirata, 2012. volume 5, pp. 99-102

[8] Tóth Zsolt, Kovács László *Fonetikai tulajdonságok alapú abc készítése* , Multidiszciplináris tudományok, 2013. volume 3 pp. 317-326

[9] Zsolt Tóth, László Kovács *Testing Linear Separability in Classification of Inflection Rules* , SISY 2014 IEEE 12th International Symposium on Intelligent Systems and Informatics, pp XX. 2014

[10] Zsolt Tóth, László Kovács *Classification based Learninig of Inflection Rules Enhanced with Associative–Memory* , Scientific Bulletin of "Petru Maior", 2014, vol. 11, no. 2, pp. 9-16.

[11] László Kovács, Zsolt Tóth *Inference of Probabilistic Grammars in Different Rules Systems of Natural Languages* In Procedia Technology, volume 12, pp. 3 - 10, 2014. (The 7th International Conference Interdisciplinarity in Engineering, INTER-ENG 2013, 10-11 October 2013, Petru Maior University of Tirgu Mures, Romania)

[12] Zsolt Tóth, László Kovács *Lattices based Classification for Learning of Inflection rules in Hungarian* , to appear