Scientific Bulletin of the „Petru Maior" University of Tîrgu Mureş
Vol. 11 (XXVIII) no. 2, 2014
ISSN-L 1841-9267 (Print), ISSN 2285-438X (Online), ISSN 2286-3184 (CD-ROM)

INDUCTION OF INFLECTION RULES WITH CLASSIFICATION AND ASSOCIATIVE MEMORY FOR HUNGARIAN LANGUAGE

Zsolt TÓTH, László KOVÁCS
University of Miskolc
Miskolc, Hungary
tothzs@iit.uni-miskolc.hu
kovacs@iit.uni-miskolc.hu

Abstract

Inflection is a vital element to express semantic in synthetic languages. Proper induction is crucial for text generation and reporting systems. The induction of inflection rules is an open question in computational linguistics. The existing solutions use dictionary, transformation rules or statistical observations to inflect a stem. These methods have drawbacks either in precision and cost efficiency. This paper present a hybrid method which is based on classification and associative memory. The words which belong to non-frequent categories are stored in the associative memory thus the classification process can be performed faster. The transformations for the regular words are determined by the classifier. Precision, size and time cost of the algorithm are measured with different sized associative memory. The tests were performed on a training set of (stem, inflected form) pairs for the accusative case in Hungarian. The precision of the hybrid algorithm can exceed the 90 per cent based on the experimental results.

Key words: Natural Language Processing, Inflection, Rule Induction, Classification, Precision

1. Introduction

Natural Language Processing has a growing popularity for the last two decades. Text based documents such as emails, report, messages or even recorded speeches or videos give a significant part of the stored data. Text mining applications like information retrieval and document search systems have evolved in the last twenty years. In spite of the promising results and good performance of the existing systems they are often limited to the major languages. Because the solutions are related to a given language, they have to be ported to another natural language which is a costly task. The conversion requires deep knowledge about the language and its grammatical structures, moreover it is usually a time consuming task.

Although most of the researches are focused on English or other major language [1], there are solutions for languages with only a few million native speakers such as Hungarian. The "Szószablya" project [2] [3] provides a morphological analyzer for Hungarian. The KOPI is a plagiarism [4] [5] checker developed by the Hungarian Academy of Sciences. These projects are focused on the information extraction.

Text generation is another branch of natural language processing which aims to generate natural language texts such as reports or questions. For Hungarian, László Bednarik created a system to generate questions for exam from annotated text [6].

Hungarian is an agglutinative language so the inflection plays an important role in its grammar. Words are created by adding suffixes to the stem and the suffix slightly modifies the meaning of the stem. There are about 17 different cases in Hungarian [7] which makes the language complex. This paper focuses on the induction of inflection rules of the accusative case of the Hungarian.

There are dictionary based, rule based and statistical methods to learn inflection rules in the literature. The dictionary based methods have high precision but they are costly and they cannot deal with unknown words. The rule based methods have difficulties with the exceptions and they have a trade-off within precision and cost. Although it is easy to create statistical methods but they have low precision. This paper presents a novel method which uses a classifier enhanced with associative memory. The presented method can achieve approximately 90 per cent precision which is better than the pure classification based algorithms.

The paper is organized as follows. Section 2

presents a brief overview of the related works. The section presents a brief survey on the different morphological analyzers, stemmers and inflection algorithms. The proposed hybrid method is detailed in Section 3. The experimental results are summarized in Section 4. The corresponding training set contains approximately 54.000 (stem, inflected form) pairs of the accusative case in Hungarian. The performed measurements were focused on the precision, time cost and size of the classification structure. The conclusions are summarized in Section 5.

2. Related Works

Computational Linguistics aims at capturing the aspects of the natural languages by rule-based and statistical models. Computational linguistics provide solution for various tasks such as morphological analysis, stemming or inflection. The algorithms of computational linguistics are widely used in Natural Language Processing solutions. For example stemmers and morphological analyzers are used in information retrieval systems. Algorithms on inflection are used in text generation and machine translation applications, too.

The first systems were focusing on simpler morphological problems. For example the inflection of the past tense of English was analyzed in [8]. It is assumed that there are different modules in human mind which are responsible for the inflection. There is a rule based process to inflect the regular verbs which allow the inflection of even unknown words. But its drawback is the over regularization which transforms irregular verbs into incorrect form. For example children make grammatical errors when they say "comed" or "breaked" [9]. The learning of irregular verbs is similar to an associative memory. On the other hand there can be similarities found between irregular verbs which lead to the theory of rule-associative-memory.

Inflection is the inverse function of stemming which is well-studied in text mining. Stemming algorithms are based on various approaches such as dictionary, rules or statistics. Although dictionary based methods provides the highest accuracy, they cannot generalize nor handle unknown words. Moreover the building of the dictionary is costly, time-consuming and requires language experts. Rule based stemming methods has a trade-off between accuracy and cost. The Porter stemmer is one of the most popular rule based stemming algorithms [10], [11]. Statistical methods requires no language experts, but they can have a low accuracy. An unsupervised statistical stemming method is presented in [12] which transforms the induction task into an optimization task. Although it has promising results for fusional languages, it was not tested with agglutinative languages which have more suffixes and more complex inflection rules.

SMOR is a morphological analyzer [13] for German inflection rules based on Finite State Transducers. The rules were implemented in Stuttgart Finite State Transducer tools and SMOR uses a lexicon which only stores the properties of the stems. SMOR has rules for prefix, suffixes, derivation, composition and inflection. In the experiments the precision of the SMOR were above 95 per cent in general and the precision depends on the frequency of the word.

Finite State Transducers are widely used for morphological analysis and translation. Stochastic transducers are also used to learn morphology [14]. The AraComLex [15] is a morphological analyzer for Modern Standard Arabic. Finite State Transducers are used to perform the analysis. Finite State Transducers are used also in machine translation systems [16].

Bayesian approach was used to perform morphological analysis in [17]. It assumes that the spelling rules occur at the end of the word. The $P(c,t,f,y,r|w)$ model is used to define the stem for the word where w is the word, c is the class of the word, t is the stem, f is the suffix, y is the type of the spelling rule and r is the transformation. During the inference a standard Markov Chain Monte Carlo technique was used. Their experiments showed that the accuracy of the stem and suffix recognition depends on the context. The accuracy of stem recognition is about 65 per cent and the accuracy of suffix recognition is about 78 per cent. Although this method is no as precise as the above mentioned rule based algorithms, it does not require human experts and a priori knowledge about the grammatical rules of the language.

The endings of the words are considered as classes in [18] because the language learners often learn induction tables where a cell denotes an inflection class. Based on their endings the words are organized into candidate inflection classes. These classes can be organized into a lattice. The authors tested five different reduction algorithms from the point of view of precision and recall. The tests were evaluated with both English and Spanish languages.

3. Hybrid Method

The algorithms of Computational Linguistics usually have a common model which can be seen in Fig. 1. Morphological analyzers, stemmers and inflection systems usually have two core parts. There is 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 [11] is a language to describe stemming rules for Porter stemmer [10]. Rules of the SMOR [13] 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.

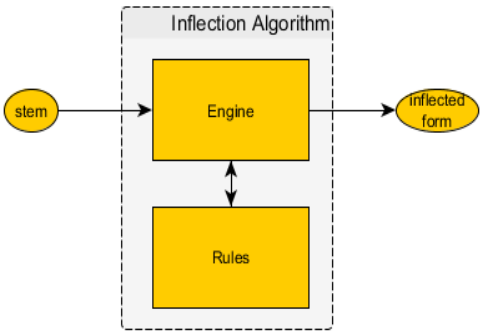

Fig. 1 Common Model of Inflection Algorithms

The proposed method is based on classification system. The input data structure of a classification system consists of a set of objects O, a set of class labels C, and a set of attributes A. The attributes of the objects are defined by the function $a: O \rightarrow P(A)$ where $P(A)$ is the power set of the attributes. The usually unknown function $c: O \rightarrow C$ maps the objects to class labels. Equation 1 shows the formal definition of the training set.

$$T \subseteq \{(a(o), c(o))|o \in O\} \quad (1)$$

The approximation of c, classification function $cl_T: O \rightarrow C$ depends on the data in the training set. The classification function is defined in Equation 2 where Pr stands for the probability.

$$cl_T(o) = argmax_C\{Pr(C|a(o), T)\} \quad (2)$$

The model of the presented inflection algorithm is shown in Fig 2. The implemented rule set consists of two parts: an associative memory and a classifier. The associative memory is used to store the transformation string of the irregular words and the classifier is used for the regular ones.

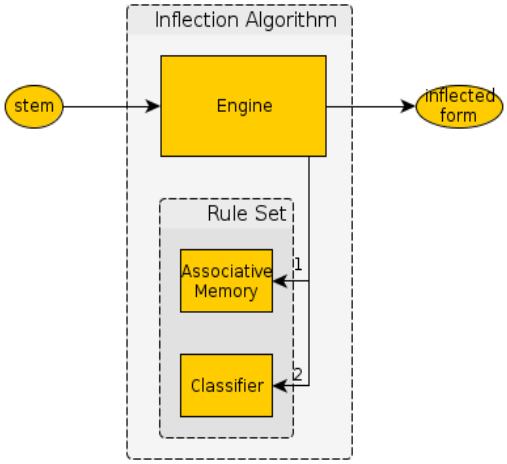

Fig. 2 Model of the presented Inflection Algorithm

The classification of the stems is a difficult problem due to the high number of the linear non separable cluster pairs [19]. The test method on linear separability is based on Simplex method [20]. The number of linear non separable cluster pairs can be reduced, if the irregular words are stored in an associative memory. Thus the classification problem can be simpler and higher precision can be achieved.

Table 1 shows how the number of the non-separable cluster pairs and the number of the clusters decreas if the small clusters are removed. The first column shows the minimum size of the clusters which remain in the training set. The second column shows the number of the non-separable cluster pairs. The third column shows the number of the clusters in the reduced training set. The high number of different class labels can increase the difficulty of the classification. Without the elimination of the small clusters there are 158 clusters and 96 non-separable cluster pairs in the training set. If the clusters which contains less than 100 objects are removed, then the number of the non-separable cluster pairs decreases approximately to its quart and the number of the clusters decreases about its tenth. After the reduction of small clusters, the elements of these clusters are stored in a separate list.

Table 1: Results of the reduction.

Min. Cluster Size	Number of non-separable cluster pairs	Number of Clusters
0	96	158
5	84	46
10	76	35
25	42	23
50	34	19
100	23	13

The classifier is used to learn the frequent inflection rules. The algorithm of this function depends on the chosen classification method. Classifiers can perform generalization. The generalization may easily fail on exceptions. Thus classifier systems usually have lower precision than associative memory. Moreover, classifiers have more difficult learning algorithm which requires a significant additional learning time cost. In some cases, the classification process also can have a significant time cost. The instance based classifiers, such as k-NN classifier [21], determines the k most similar object to the classified instance from an instance database. The distance calculation and the search can be costly thus the inflection algorithm can be slow.

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 [22]. The transformation strings are considered as categories of stems. Thus the inflection rule induction task is converted into a classification task where the stems are the investigated objects and the transformation strings are the categories.

The classification algorithms in the literature are based on various approaches such as Bayes theorem [23], decision trees [24], artificial neural networks [25] or support vector machines [26]. Our previous experiments [19] showed that approximately 70 per cent precision can be achieved with standard classification methods in the induction of inflection rules of the accusative case in Hungarian.

The representation form of the words affects the efficiency of the classification process, too. The words are usually converted into real vectors by mapping the letters into real values. This mapping can be based on code tables such as ASCII or traditional alphabet. These mappings have numerous drawbacks. For example the distance of the letters is constant and the phonetic features of the letters are not considered. A phonetic features based alphabet was presented in [27] for Hungarian. The phonetic alphabet based encoding was shown superior to the traditional alphabet and the ASCII code table based encodings.

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.

The learning phase has two main steps. In the first step the categories are ordered by their size. Then the associative memory is populated with the irregular words. If a word is put into the associative memory, then it is also removed from the training set. The population of the associative memory is based on a greedy approach i.e. if the size of the associative memory is bigger than the training set, then the training set is put into the associative memory. Then in the second step, the classifier is built based on the rest of the training set. The efficiency, the training cost and the precision depends on the chosen classification method.

The presented method is evaluated from the point of the view of precision, size and time cost. The precision is the ratio of the correctly inflected words and the total number of the words. The size of the rule set is measured with the length of the serialized object. Finally the time cost is measured as the required time for learning in milliseconds. The method was evaluated with alphabetical and phonetic alphabet based letter encodings and Naïve Bayes and Multi-Layer Perceptron classifiers and different sizes for the associative memory.

4. Experimental Results

The experimental measurements were implemented on a training set of 54,000 pairs (stem, inflected word) of the accusative case of Hungarian. The inflection algorithm was implemented as a module of the META framework in Java. Alphabetical and phonetic encodings were used in the tests. The Weka data mining and machine learning framework was used for classification. Naïve Bayes and Multilayer Perceptron classifiers were used to learn the inflection rules. In the learning phase, the 75, 90 and 100 percent of the training set was used to train the algorithm. But the entire training set was used during the testing. Thus the behavior of the algorithm with untrained input can be examined. The measurements were done with both fixed and relative associative memory sizes. The fix measurements were done with small associative memory sizes because it was assumed that there are only a few irregular words. The relative sizes were set to every 10 per cent of the size of the training set.

Precision

The precision is one of the most important parameters of the classifiers. It shows the ration of the correctly classified sample and the validation set. In the measurements there were 54.000 samples in the training set and this set was the validation set. The training set contains the stem of almost every Hungarian nouns. The training set was reduced in order to observe the behavior of the algorithm with untrained words.

Fig. 3 and Fig 4 show how the precision changes in the function of the size of the associative memory with alphabetical and phonetic alphabet based encodings with Naïve Bayes classifier. The x axis shows the size of the associative memory and the y axis shows the precision. If the size of the associative memory is zero then the algorithm uses only the classifier to determine the transformation strings. In this case the precision of the algorithm is equivalent with the precision of the classifier. The precision of the algorithm increases with the size of the associative memory. The precision reaches the top around 20.000 which is approximately the 40 per cent of the whole training set. The precision has a break down when the entire training set fits into the associative memory. It can be explained with that in this case the algorithm do not use the classifier so that it cannot generalize. In these case the precision is the ratio of the training set and the validating set.

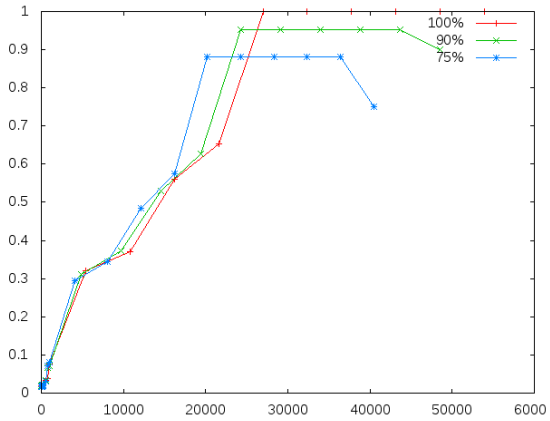

Fig. 3 Precision with Naive Bayes Classifier and Alphabetical Encoding

The precision increases more quickly with the reduced training sets in the cases of both encodings. Although the precision is peak at the same level with both alphabetical and phonetic encodings, the precision increases more quickly with phonetic encoding. These phenomena 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 incensement of the precision of the inflection algorithm.

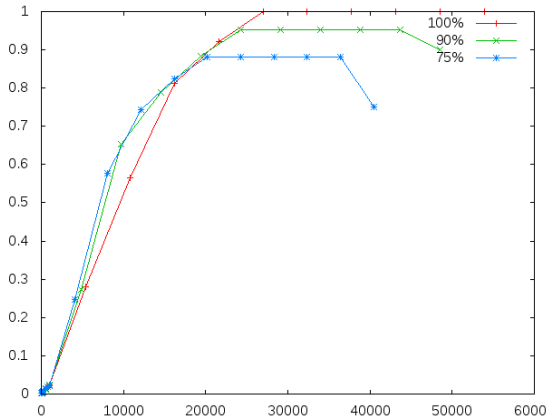

Fig. 4 Precision with Naive Bayes Classifier and Phonetic Encoding

Multilayer Perceptron shows better precision than Naïve Bayes classifier even without associative memory. In this case the precision increases steadily and tops around 20.000 similar to the Naïve Bayes classifier. Fig. 5 shows how the precision depends on the size of the associative memory in the case of the Multilayer Perceptron with alphabetical encoding. Fig. 6 shows how the precision changes with the size of the associative memory in the case of phonetic encoding. In this case, regarding the Multilayer Perceptron classifier there is no significant difference between the two encoding unlike with the Naïve Bayes classifier.

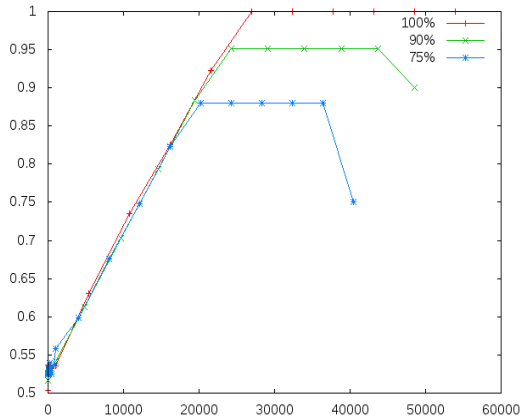

Fig. 5 Precision with Multilayer Perceptron Classifier and Alphabetical Encoding

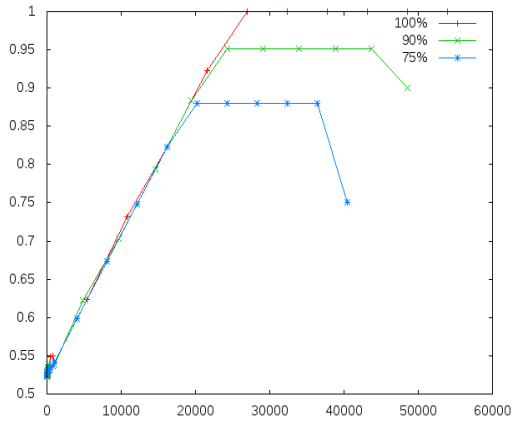

Fig. 6 Precision with Multilayer Perceptron Classifier and Phonetic Encoding

Size

The size of the algorithm is a vital property of the algorithm from the point of view of performance. Algorithms which use a lot of memory are often slow due to the frequent memory swaps. An algorithm with small memory cost could require less memory swap or even could fit into the memory which makes it faster.

The size of the learning algorithm was measured as the size of the serialized object in bytes. The serialization was possible with the Java API because the Classifier class of Weka implements the Serializable interface. Because the Naïve Bayes and the Multilayer Perceptron classifiers had similar behavior with both encodings, Fig. 7 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 x axis shows the size of the associative memory and the y axis shows the size of the serialized object of the algorithm in bytes.

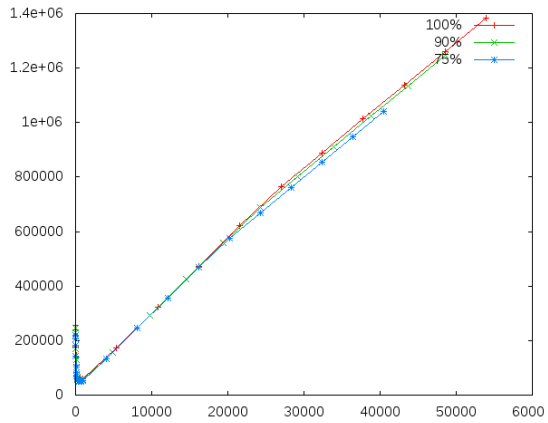

Fig. 7 Size of the inflection algorithm

The size of the algorithm decreases quickly until approximately 1000 then it starts to increase steadily. The size increases linearly because the size of the associative memory also increases linearly. So linear connection between the size of the algorithm and the size of the associative memory can be assumed. Fig. 8 shows how the size of the algorithm changes between 0 and 3000. It shows that the size drops until about 200 then it has a minimum between 500 and 1000. Finally it starts to increase. The simplification of the classification method can be the reason of this fall. 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.

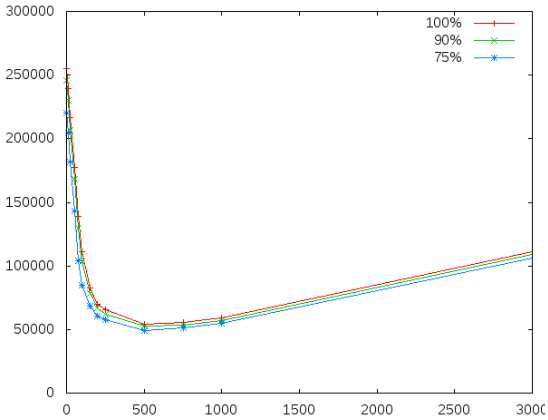

Fig. 8 Size of the inflection algorithm

Time 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 significanty depends on the classifier. Fig. 9 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 steadyly and there is a fall around 20.000.

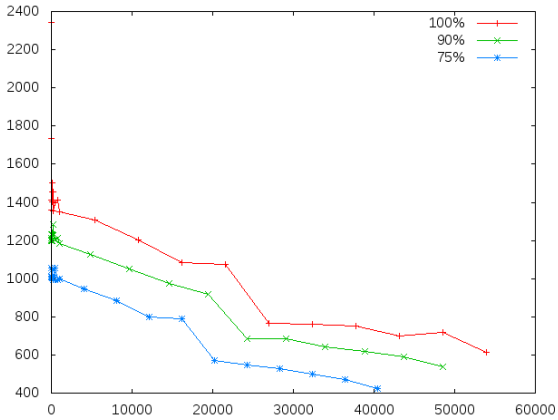

Fig. 9 Time Cost with Naive Bayes Classifier

Fig. 10 shows the time cost in the case of the Multilayer Perceptron classifier. Although the training cost of the neural network is much more higher than the Bayesian classifier the time cost function is similar. The time cost can be reduced with the application of a small associatvie memory. Then it decreases slowly and it also has a break down around 20.000.

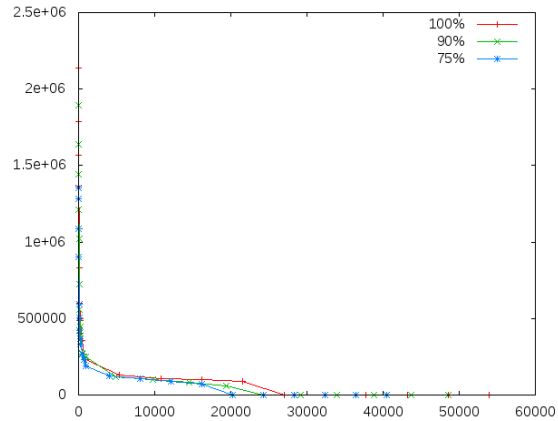

Fig. 10 Time Cost with Multilayer Perceptron Classifier

The measurements showed that the precision of the classification based inflection algorithms can be increased with the usage of associative memory. But above a certain size of associative memory the precision will not increase. Moreover if the associative memory is too big then the precision of the algorithm can decrease.

Measurements showed that the size of the algorithm depends on the size of the associative memory. Above a certain size of associative memory there is a linear connection between the algorithm and the associative memory. The learning cost depends on the classification method. However the different

classification methods had different learning cost the time cost decreased similar with both classifiers

5. Conclusion

This paper presented a classification method enhanced with associative memory for inflection algorithm. The method uses associative memory to learn the irregular words and the exceptions. The classifier is used to capture the regular transformation rules. The algorithm looks for the rules first in the associative memory. If it does not find the rule, then the classifier is used to determine the transformation rule. These two phases allows to achieve high precision and compact size.

The experimental measurements were focused on the precision, the size of the algorithm and the learning cost. Results showed that precision increases fast with the size of the associative memory. The maximum precision was achieved with an approximately 20.000 sized associative memory for a training set of 54.000 samples. Phonetic alphabet based encoding showed better results with Naïve Bayes classifier and the encoding had no significant effect on the Multilayer Perceptron classifier. Measurements on the size of the algorithm showed that the size grows linearly with the size of the associative memory and the classifier do not modify the size significantly. Experimental results show that the learning cost of the algorithm depends on classifier. The learning cost decreased similarly with both tested classifiers although the order of the cost function was different.

The presented inflection algorithm is capable to handle irregular words and to determine the transformation string for the regular ones with high precision. The algorithm could achieve approximately 90 per cent precision with incomplete training sets which is superior to the pure classification based methods. The size of the associative memory is a vital parameter of the method. The proper chose of this parameter requires tuning or examination of the training set.

References

[1] Maria Chiara Caschera, Arianna D'Ulizia, Fernando Ferri, Patrizia Grifoni (2014) An Italian Multimodal Corpus: The Building Process, in *On the move to Meaningful Internet Systems: OTM 2014 Workshop*, pp.: 557-566

[2] Richárd Farkas, György Szarvas (2004), Statisztikai alapú tulajdonnév-felismerő magyar nyelvre, [Statistical Named Entity Recognition for Hungarian], in *II. Magyar Számítógépes Nyelvészeti Konferencia [2nd Conference on Hungarian Computational Linguistics]*, pp. 136-140

[3] Péter Halácsy, András Kornai, László Németh, András Rung, István Szakadát, Viktor Trón (2004), Createdin Open Language Resources of Hungarian, *in LREC*

[4] Máté Pataki (2006) Plagiarism search within one document, in *Proceedings of the Automation and Applied Computer Science Workshop*, pp. 187-194.

[5] Máté Pataki (2006), Distributed similarity and plagiarism search, in *Proceedings of the Automation and Applied Computer Science Workshop*, pp. 121-130.

[6] László Bednarik (2012), *Automatizált kérdésgenerálás annotált szövegből [Automatic Question Generation from Annotated Text]*, PhD Thesis

[7] Edith Moravcsik (2003), Inflectional morphology in the Hungarian noun phrase, in *Noun phrase structure in the languages of Europe*, vol. 20.

[8] Steven Pinker (1991), Rules of Languages *Science* vol. 253, no. 5019, pp. 530-535.

[9] Gary F. Marcus (1996) Why do children say "breaked"?, *Current Directions of Psychological Science*, vol. 5, pp. 81-85.

[10] Martin F. Porter (1980), An algorithm for suffix stripping, *Program: electronic library and information system*, vol. 14 no. 3 pp. 130-137.

[11] Martin F. Porter (2001), Snowball: A language for stemming algorithms [Online]. Available: http://snowball.tartarus.org/texts/introduction.html

[12] Alexander Gelbukh, Mikhail Alexandrov, Sang Yong Han (2004), Detecting inflection patterns in natural language by minimization of morphological model, *Progress in Pattern Recognition , Image Analysis and Applications*, Springer pp. 432-438.

[13] Helmut Schmid, Arne Fitschen, Ulrich Heid (2004), SMOR: A German Computational Morphology Covering Derivation, Composition and Inflection, *Proc. LREC*, pp. 1263-1266.

[14] Alexander Clark (2002), Memory-based learning of morphology with stochastic transducers, *Proceedings of 40th Annual Meeting of Association of Computational Linguistics*, pp. 513-520.

[15] Mohammed Attia, Pavel Pecina, Antonio Toral, Lamia Tousi, Josef van Genabith (2011) in *Proceedings of the 9th International Workshop on Finite State Methods and Natural Language Processing*, pp. 125-133.

[16] Johnathan North Washington, Mirlan Ipasov, Francis M. Tyres (2012) A finite-state morphological transducer for Kyrgyz in *Proceedings of the 8th international conference on Language Resources and Evaluation (LREC'12)*, pp. 934-940

[17] Jason Naradowsky, Sharon Goldwater (2009), Improving Morphology Induction by Learning Spelling Rules, *IJCAI*, pp. 1531-1536.

[18] Christian Monson, Alon Lavie, Jamie Carbonell, Lori Levin (2004), Unsupervised Induction of Natural Language Morphology Inflection Classes, *Proceeding of 7th Meeting of the ACL Special Interest Group in Computational Phonology*, pp. 52-61.

[19] Zsolt Tóth, László Kovács (2014), Testing linear separability in classification of inflection rules, *Proceedings of 12th International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 27-32.

[20] David Elizondo (2006) *The linear separability problem: Some testing methods*, in IEEE Transactions on Neural Networks vol. 17. No. 2. pp. 330-344.

[21] Pádraig Cunningham, Sarah Jane Delany (2007), k-Nearest Neighbour Classifiers, in *Multiple Classifier System* pp. 1-17.

[22] Gonzalo, Navaro (2001), A guided tour to approximate string matching, *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31-88.

[23] George H. John, Pat Langley (1995) Estimating Continuous Distributions in Bayesian Classifiers, in *Proceedings of 11th Conference on Uncertainty and Artificial Intelligence*, pp. 338-345.

[24] A. Sirvastava, E. Han, V. Kumar, V. Singh (2002), Parallel Formulations of Decision-tree Classification Algorithms

[25] Anil K. Jain, Jianchang Mao, K. Modin Mohiuddin (1996) Artificial Neural Networks: A tutorial in *Computer* vol. 29 no. 3 pp. 10-18.

[26] Johan Suykens, Tony Van Gestel, Jos de Brabanter, Bart de Moor, Joos Vandewalle (2002) *Least Square Support Vector Machines*

[27] Zsolt Tóth, László Kovács (2013), Fonetikai tulajdonságok alapú ábécé készítése [Phonetic features based alphabet], *Multidiszciplinári tudományok*, vol. 3, no. 1. pp. 317-326.