RAOUL KARIMOV
Chelyabinsk State University

DOI: 10.15290/cr.2018.21.2.04

Combined Machine--Learning Approach to PoS-Tagging of Middle English Corpora

Abstract. This paper considers the problem of part-of-speech tagging in Middle English corpora (as well as historical corpora in general). Whereas PoS-tagging in general is now considered a solved problem for Modern English and is mainly achieved via hidden Markov models (HMM) and matrix-based word-to-vector conversions with every word in the dictionary being embedded into a single dimension, this approach relies on recurrent syntactic structures and context-free generative grammars and is therefore not applicable to older iterations of the English language due to irregular word order. As such, we believe that Middle English could be better handled by a morphographemic encoding and instance-based machine learning algorithms like SVM, random forests, kNN, etc. Using a moving-average method to generate multidimensional vectors giving a reliable numeric representation of character composition and sequences, we have achieved a precision and recall of 87.5% in classifying Middle English words by their part of speech while using a simplistic combined voting-based binary classifier. This result could be deemed satisfactory and encourages further research in the area.

Keywords: Instance-Based Learning, Corpus, Middle English, PoS-Tagging, Moving Average.

1. Introduction

Part of speech tagging is one of the central issues in the discipline known as natural language processing; it is quite frequently approached by means of the sequential-in-nature hidden Markov models that are basically designed to predict the probability of an event in a sequence given the previous events (Jurafsky 2008: 139). As Modern English follows a very strict word order, where, for instance, the definite article the and most other determiners are in over half of all cases followed by a noun, HMMs can be efficiently used to correctly classify words by their part of speech. Furthermore, PoS-tagging is assisted by finite-state transducers, which derive a given word's morphological properties by identifying grammatically-significant character sequences, or morphemes (Beesley and Karttunen 2004: 240). It is, however, a completely different story with Middle English, especially its earlier iterations (post-William the Conqueror's capture of Britain in 1066 and up to the issuance of Magna Carta Libertatum in 1215). The language, being characterized by less regular word order as well as rich morphology and very inconsistent orthography, where *scyl-*

de was the same was as *scilde* (shield), and *y-broht* was only slightly diachronically separated from *gebrouhte* (Pt. II of *bringan*, to bring), still presents a challenge for linguists working in the field of corpus linguistics and natural language processing, whereas not being extensively researched by machine learning community who prefer to allocate more resources, time, and scientific effort to modern languages.

We have, unfortunately, so far found no previous research what would be dedicated specifically either the lemmatization or the PoS-tagging of corpora in Middle English. Neural networks have been made for Slavic and other morphologically complex languages (Jędrzejowicz and Strychowski 2005: 200; Malouf 2016: 123), but utilized better-codified and larger-in-volume language data than we could ever afford in our Middle English effort. Nevertheless, PoS-tagging and other functions provided by NLP applications could be of great use for philologists studying language history who are currently restricted to corpora with limited annotation and as such have to perform annotation manually, which is both time-consuming and labor-intensive, thus reducing the overall efficiency of scientific work. With that in mind, we decided to find a way to automate the process of PoS-tagging by applying existing machine learning methodology.

For this study, the following was hypothesized: there should exist a simple instance-based machine learning method that would enable efficient PoS-classification of orthographically volatile Middle English words while trained on a small set of data. We believed that support vector machines (SVM), random forest models (RFM), k nearest neighbors (kNN), and Bayesian algorithms could all be used for such learning. The hypothesis is to be verified by means of 10-fold cross-validation, whereas the difference in results returned by various algorithms is to be t-tested by Student's method.

2. Theory and methodology

2.1. Research data

This study derives data for analysis primarily from the Helsinki Corpus of English Texts (Web 1), which contains about 450 texts of philosophical, literary, belletristic, epistolary, scientific, and religious nature, a total of 1.5 million words. The corpus is freely available via the Oxford Text Archives and is subdivided into multiple smaller divisions based on the diachronic classification of its texts, which in its turn is dual, as it considers both the creation of the original text and the creation of its manuscript that the corpus includes. Preliminary analysis of the corpus data and the preparation of training and test samples were done by consulting Mayhew and Skeat's *A Concise Dictionary of Middle English* (Mayhew and Skeat 1888). For the goal of this research paper, we limited ourselves to one of the texts from the corpus: Vespasian Homilies, ca. 1167, which partially reduced the overall orthographic and grammatical inconsistency that could be observed across the dialects of that time.

For the initial machine learning effort, we would like to perform simple binary classification to see what results are reasonably achievable on a smaller training set. For this purpose, we made a

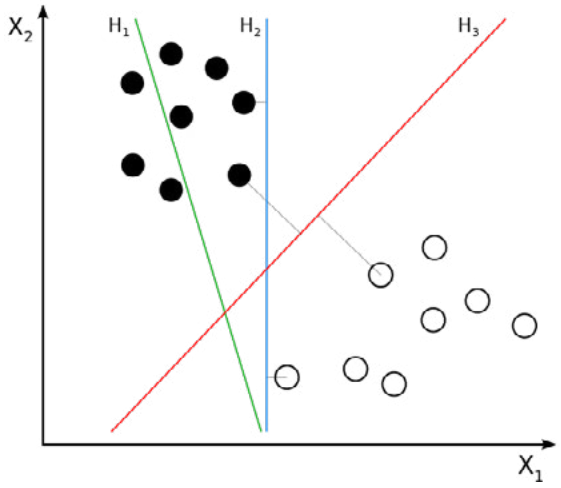
small 200-word set containing 110 verbs and 90 adjectives, all extracted from Vespasian Homilies as they are contained in the text, i.e. without lemmatization, spelling normalization, or any other pre-training rectification, which we believed would be inappropriate given that the ultimate goal of the research effort was (and is) to develop a technique that could be applied to raw corpus data. The set is made deliberately small so as to better optimize the methodology for training on limited data.

2.2. Algorithms Under Investigation

In this on-going research effort, we are investigating the capacities of several known instance machine-learning algorithms when applied to small Middle English vocabulary samples (for space considerations and for the sake of simplicity, we are not providing any detailed mathematical descriptions of those algorithms, see sources cited): Support Vector Machines, **K Nearest Neighbors, Random Forest Models, and** Multilayer Perceptron.

A support vector machine, or SVM, is in its most basic form (a binary linear machine) an algorithm that, given a set of vectors in *n*-dimensional vector space, finds a hyperplane that maximizes the margin between itself and the vectors on either side of it. Figure 1 below gives a good illustration of how it works:

Figure 1. Separation of Vectors

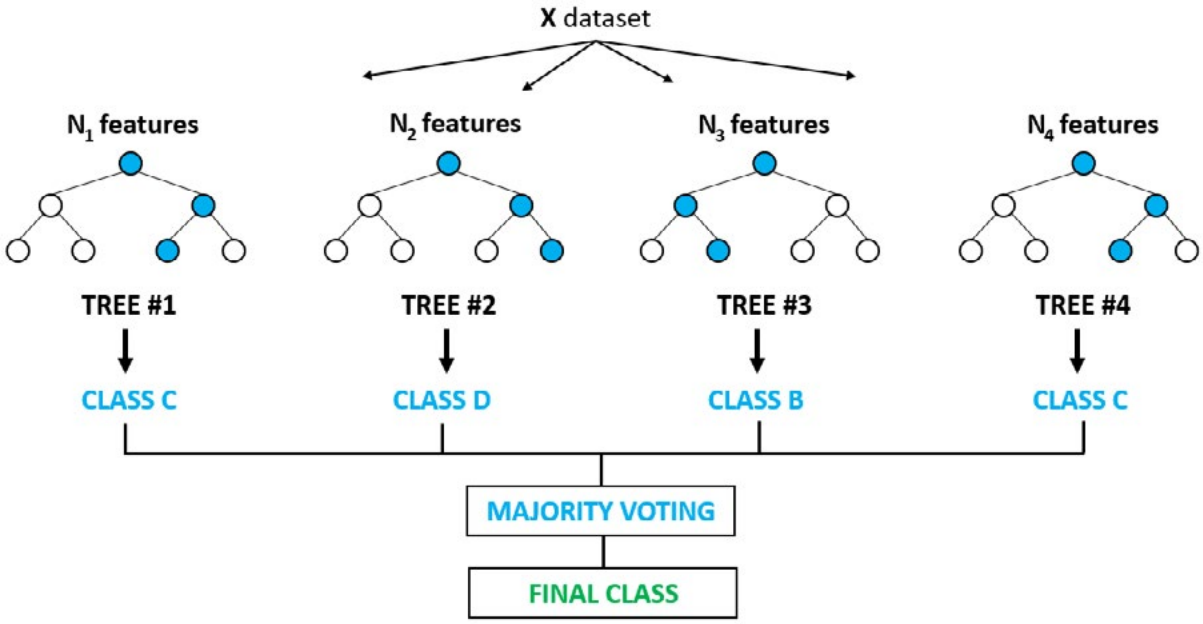


In the figure above, black and white dots represent linearly-separable vectors of different classes; the green line does not separate the classes; the blue one does with a minimum margin; the red one does with a maximum margin and is therefore deemed the best hyperplane. In case of linearly separable data, the maximum-margin hyperplane lies between two other planes that border the separated vector sets; vectors from the data set that belong to such other planes are called support vectors, hence the name of the method; for a detailed mathematical description, see (Christianini and Shawe-Taylor 2000: 94).

Random forest models, or RFM, generate decision trees, each of which is applied to classify a random subset of the training sample using a random partial set of attributes, or features (neither

the sample, nor the attributes are used entirely). When classifying a new instance, RFM gives preference to the class which the majority of such randomly-generated trees has voted for, i.e. performs simple majority voting (Breiman 2001: 2). Of the randomly-reduced set of attributes, the best one that is used for generation of tree nodes is often chosen on the basis of Gini impurity. Below is an illustration of how it works:

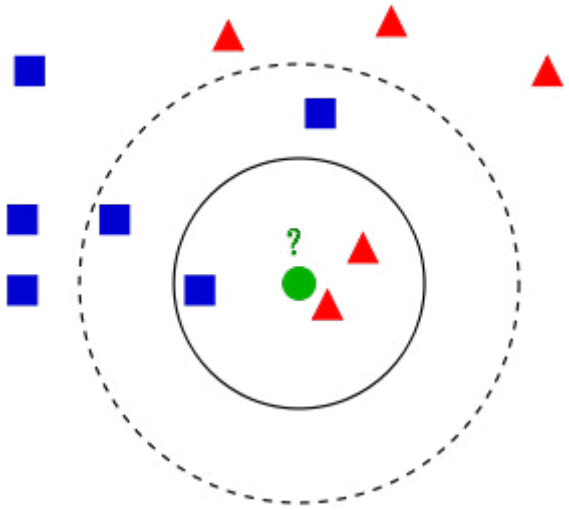
Figure 2. Random Forest Model



As can be seen in Figure 2, each decision tree is fed a part of the data set, drawn at random, and uses a random subset of features to perform the classification of an unknown-class instance. In tasks other than classification, e.g. regression, averaging can be used instead of majority voting. The random forest algorithm scales well, is efficient for any number of attributes or classes, is optimal for large and small data alike, and is good for both continuous and discreet attributes; while being computationally intensive is often cited as its main drawback, it does not seem to be problematic given our purpose to use deliberately small training sets.

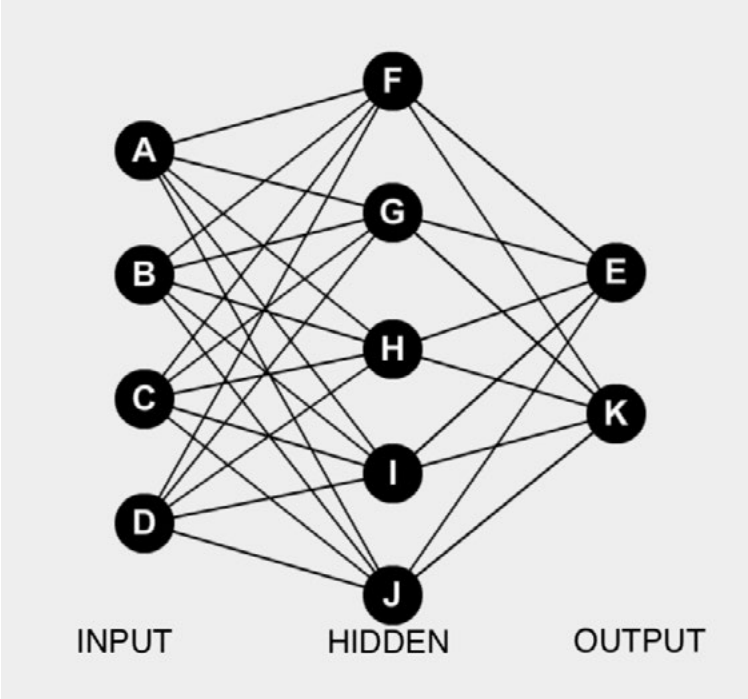
K nearest neighbors is one of the simplest yet very efficient instance-based learning algorithms and classifiers that uses only basic majority voting (Aha 1991: 38). A new instance is vectorized and then placed in an n-dimensional space of correctly classified vectors (e.g. the training-sample vectors), whereby the algorithm calculates the Euclidean or Manhattan distance from this new vector to the known-class vectors to identify a number of nearest neighbors. The number is usually odd and is referred to as k, hence the name of the algorithm. Whichever class has the most vectors among those k nearest neighbors, that class is assigned to the new vector as well. In Figure 3 below, the green vector in the center would be classified as a red triangle if $k = 3$, or as a blue square if $k = 5$.

Figure 3. K Nearest Neighbors



Multilayer perceptron, or MP, is the most complex algorithm of the above, and is a feedforward artificial neural network (Teijiro et al. 2012: 758); unlike SVMs, it can be used on data that is not linearly separable. It builds a network consisting of three or more layers (input, output, and at least one hidden layer) containing nodes with weights adjusted by a sigmoid function. Weight adjustment is done by the backpropagation of error. MLPs are successfully used in scientifically complex prediction problems like Seyed (2015: 62).

Figure 4. Schematic representation of an MLP with three separately-weighted hidden ode layers



Middle English, despite being rather inconsistent both grammatically and orthographically, did have regular morphs that are still referred by historical linguistics as the primary categorial markers. Hypothetically, if we were able to generate word-vectors such that similar character sequences occurring in similar intra-lexeme positions would produce closely-positioned vectors, then a vector-based machine learning algorithm such as SVM or an RFM should be able to correctly link together words that have similar graphical clusters at the beginning (the prefix) and/or at the end (the suffix), which in many (though not all) cases would suffice for part-of-speech classification.

2.3. Methodology

As has already been mentioned above, the method to use had to focus on the recurring sequences of symbols observed in words and signifying its PoS-category. As such, we had to find a simplistic yet efficient method that would enable us to represent words in a vector form that would reliably shaped by both the character composition of, and character-specific position in, a given word. Takala cites several methods of vector-word embedding: regular embedding (one dimension per word for n top-ranking tokens contained in the dictionary, with the rest being assigned to a single dimension associated with rare words), stem+ending embedding (stem vectors concatenated with ending vectors), and moving-average method that uses relatively small dimensionality to collect information from all parts of a word (Takala 2016: 179). In our pursuit of making such representations that would not require any pre-specified rules while also not being calculation-intensive, we decided to choose the latter.

The moving-average representation is essentially a vector of n dimensions, where n = number of characters in the alphabet, with each dimension being assigned to a single character. A word representation $w = (w_a, w_b, \ldots w_z)^T$:

$$W_\alpha = \sum \frac{(1-\alpha)^c \alpha}{z}, \quad (1)$$

where *c* is the character index (1 for the first symbol in the word, 2 for the second one, etc.), α is a hyper-parameter to control the decay, and Z is a normalizer proportional to the word length (which we decided to be the word length itself, i.e. 4 for word). Thus, each word-vector contains a weighted sum in each dimension representing any character that found in the word, and 0 in the rest of dimensions. The operation is repeated backwards, and the new vector is concatenated to the previous one so that *word* is represented as *word* ⁀ *drow*. Takala also suggests concatenating a third vector which only contains character counts; for now, we decided not to use that option.

Before the experiment was conducted, we had done a limited normalization of spelling: both thorn and eth had been replaced with the cluster *th*, whereas ash, æ, had been replaced with *ae*, and yogh, ȝ had been replaced with *g*. We also removed diacritics and decapitalized all the words in both of our sets. Thus, we came to an alphabet of 26 characters, which resulted in word-vectors in a 52-dimensional space over the field of real numbers (26x2). Two sets were then made, each containing 100 instances of 52 numeric attributes and one binary attribute POS {VERB, ADJ} to test the ability of support-vector machines to effectively differentiate these two parts of speech

based on instances extracted from a Middle English Text. All machine learning algorithms were run in the Weka data-mining environment (Frank and Witten 2016: 365).

3. Experimentation and discussion

As mentioned above, training and verification by 10-fold cross-validation were performed on a small 200-word sample containing 110 verbs and 90 adjectives from a single Middle English texts. All the four models discussed in Section 2.2 were first trained and verified separately; below are the confusion matrices for all models.

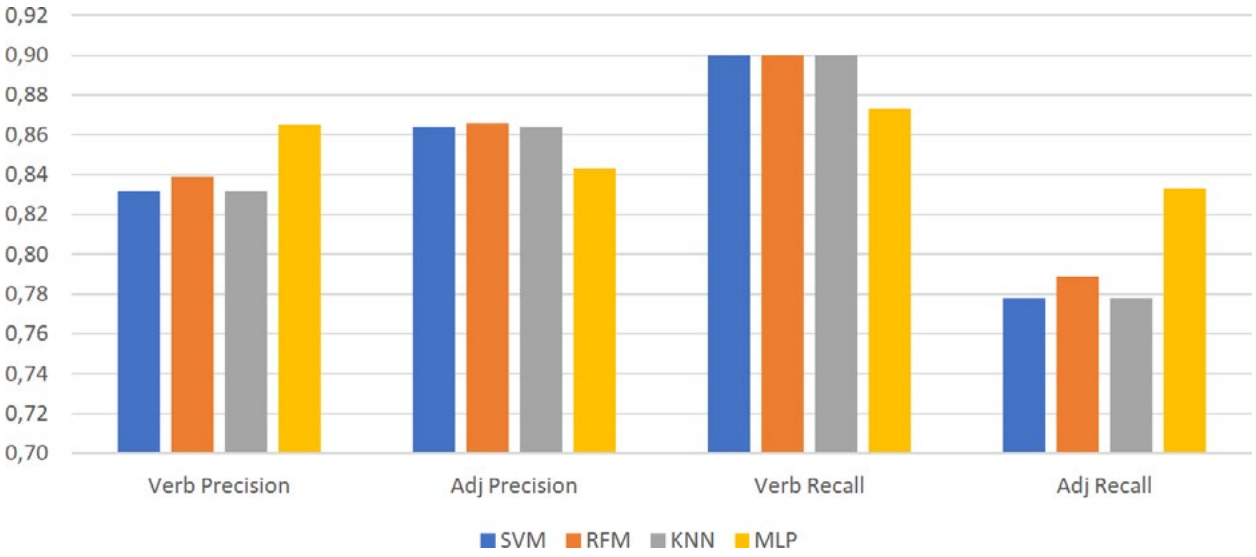
Table 1. Confusion matrices for each model

MLP			**kNN**			**SVM**			**RFM**		
verb	adj	cas	verb	adj	cas	verb	adj	cas	verb	adj	cas
96	14	verb	99	11	verb	99	11	verb	99	11	verb
15	75	adj	20	70	adj	20	70	adj	19	71	adj

Note: cas means "classified as". Values in bold are the best values achieved across algorithms.

To better illustrate how precise each algorithm was in separating verbs from adjectives, we plotted a precision-and-recall histogram for each of them, see Figure 5 below.

Figure 5. Class-specific precision and recall values for different algorithms



As can be seen from the graph, adjectives are universally classified more precisely than verbs but show considerably lower recall. Interestingly, kNN and SVM return identical results for both classes; verb recall and adjective precision have the same or nearly same values across algorithms except MLP, which is superior to others in terms of verb classification precision while somewhat worse in terms of verb recall. It is also the only algorithm to show high recall for ad-

jectives at the cost of somewhat lower precision. Table 2 provides detailed classification results for each algorithm.

Table 2. Detailed accuracy by class for each model

TP Rate	FP Rate	Precision	Recall	F-Msr	MCC	ROC Area	PRC Area	Class
Support Vector Machines								
0.900	0.222	0.832	0.900	0.865	0.687	0.839	0.804	VERB
0.778	0.100	0.864	0.778	0.819	0.687	0.839	0.772	ADJ
0.845	0.167	0.846	0.845	0.844	0.687	0.839	0.790	Weighted Average
Multilayer Perceptron								
0.873	0.167	0.865	0.873	0.869	0.707	0.909	0.919	VERB
0.833	0.127	0.843	0.833	0.838	0.707	0.909	0.872	ADJ
0.855	0.149	0.855	0.855	0.855	0.707	0.909	0.898	Weighted Average
Random Forest								
0.900	0.211	0.839	0.900	0.868	0.697	0.940	0.951	VERB
0.789	0.100	0.866	0.789	0.826	0.697	0.940	0.923	ADJ
0.850	0.161	0.851	0.850	0.849	0.697	0.940	0.938	Weighted Average
K Nearest Neighbors								
0.900	0.222	0.832	0.900	0.865	0.687	0.839	0.810	VERB
0.778	0.100	0.864	0.778	0.819	0.687	0.839	0.767	ADJ
0.845	0.167	0.846	0.845	0.844	0.687	0.839	0.790	Weighted Average

In this table, values in bold are the best precision, recall, and ROC values achieved by means of all the four algorithms tested. Note that the power of algorithms is highly contextual: MLP is better at handling adjectives, whereas RFM returns higher ROC for any class, and SVM/kNN deal with verbs more efficiently. This suggests that different algorithms might perform better or worse depending on the class, the metric, or other factors, which is why we propose making a combined

algorithm. To that end, we decided to combine all the four algorithms to make a majority-voting meta-classifier, whose results are shown below.

Table 3. Detailed accuracy by class for the majority-voting classifier

TP Rate	FP Rate	Precision	Recall	F-Msr	MCC	ROC Area	PRC Area	Class
0.909	0.167	0.870	0.909	0.889	0.747	0.871	0.841	VERB
0.833	0.091	0.882	0.833	0.857	0.747	0.871	0.810	ADJ
0.875	0.133	0.875	0.875	0.875	0.747	0.871	0.827	Weighted Average

Apparently, using a combined classifier did improve both precision and recall for both classes, which suggests it is a recommendable approach for further machine-learning efforts in this research. A weighted-average precision and recall of 0.875, we believe, indicates that the combined model showcases a sufficient capability of predicting the part of speech of a given word when trained on 52-dimensional word-vectors generated by the moving-average method. However, a few problems should be highlighted.

First it would be useful to note that verbs generally demonstrate better results than adjectives with all the algorithms, which we think is due to the sampling method: as we did not lemmatize or otherwise normalize the form of words we tested the approach on, some adjectives in both sets were given in the superlative form. Given the absence of superlative rhoticism (Ilyish 1968: 104) in Middle English, such adjectives would bear the suffix *-st-*, which coincided with the verbal 2SG suffix; as second-person verbs were abundant in the sample due to the nature of Vespasian Homilies, where the author directly addresses the reader and uses the 2SG personal pronoun þu, this could have resulted in a consistent association of the character string *-st-* with verbs, which affected superlative adjectives.

Second, it should be borne in mind that the experiment was oversimplified and reduced to two parts of speech, one of which (the verb) is known to be the most morphologically complex and rich in most languages of the world, thus boasting better and more indicative character-string markers. On the other hand, ME nouns and adjectives did share many of their case-specific suffixes, which would probably result in multiple confusions of these two parts of speech should both be included in the experiment. This might mean that the included algorithms might not make a sufficient PoS-tagging tool despite the morphological richness of Old and Middle English, necessitating the implementation of non-character-based means, e.g. hidden Markov models.

Another question that is yet to be answered is whether multi-class machines would be as efficient as binary ones. SVMs are primarily intended for binary classifications and utilize a breakdown approach when it comes to more than two classes; MLP might behave differently on a larger sample. Further experiments are needed to find out if they could be efficient in our case should

we try to train those models and/or the combined classifier to recognize all parts of speech. This is complicated by cross-PoS homography that could sometimes be observed in historical English texts, where words of different parts of speech could have the same or near-same graphical representation. This reiterates the need for including non-morphology-based techniques such as HMMs, which, however, need a different vector representation.

Finally, we have to mention the grammatical ambiguity that has existed since the appearance of analytical perfective structures in the English language, where the second participle of a verb could be used as both a non-finite component of a verbal compound, or as an attribute, in which case it would fulfill the role of an ordinary adjective. This is essentially a problem of grammatical classification rather than machine learning, yet it could adversely affect the latter.

4. Conclusions

This paper analyzes multiple classifiers and a combination thereof that use multidimensional word-vectors generated by means of a moving-average formula applied to every word in a set in direct and reverse order to create a vector reflecting both the character composition of, and the weighted character-specific position in, a given word. All the models are cross-validated on a small verb-and-adjective set sampled from a Middle English corpus. Despite somewhat worse results for adjectives, a combined voting-based classifier compensates for it by taking into account the MLP output, where VERB/ADJ precision and recall values are balanced. The results obtained encourage further research and experimentation in this area; however, they probably indicate the need for complementing the approach with another model that does not rely on character strings, e.g. HMM. This will be the basis for our future efforts in the pursuit of creating a model that is able to correctly classify words by their part of speech or other morphosyntactic properties, in Middle English and other historical languages that share the same properties: morphological richness and orthographic inconsistency. The current findings, however, are deemed satisfactory given the small sample size.

References

Aha, David W., Kibler, Dennis, Albert, Marc K. 1991. Instance-based learning algorithms. *Machine Learning* 6-1, 37-66.

Beesley, Kenneth R., Karttunen, Lauri. 2004. Finite-State Morphology. *Journal of Computational Linguistics* 30-2, 237-249.

Breiman, Leo. 2001. Random Forests. https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf (19 April, 2018).

Christianini, Nello, Shawe-Taylor, John. 2000. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge: Cambridge University Press.

Frank, Eibe, Witten, Ian H. 2016. *Data Mining: Practical Machine Learning Tools and Techniques*. Burlington: Morgan Kaufmann.

Ilyish, Boris A. 1968. *History of the English Language*. Moscow: Vysshaya Shkola.

Jędrzejowicz, Piotr, Strychowski, Jakub A. 2005. Neural Network Based Morphological Analyser of the Natural Language. *Intelligent Information Processing and Web Mining. Advances in Soft Computing* 31, 199–208.

Jurafsky, Dan, Martin, James H. 2008. *Speech and Language Processing.* New Jersey: Prentice Hall.

Malouf, Robert. 2016. Generating morphological paradigms with a recurrent neural network. *San Diego Linguistic Papers* 6, 122–129.

Mayhew, Anthony L, Skeat, Walter.1888. *A Concise Dictionary of Middle English From A.D. 1150 to 1580.* Oxford: Clarendon Press.

Seyed, Hamid H., Mahdi, Samanipour. 2015. Prediction of Final Concentrate Grade Using Artificial Neural Networks from Gol-E-Gohar Iron Ore Plant. *American Journal of Mining and Metallurgy* 3-3, 58-62.

Takala, Pyry. 2016. Word Embeddings for Morphologically Rich Languages. *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 177-182.

Teijiro, Isokawa, Naruhiko, Nishimura, Nobuyuki, Matsui. 2012. Quaternionic Multilayer Perceptron with Local Analyticity. *Information* 3, 756-770.

Web 1 – Helsinki Corpus of English Texts.www.helsinki.fi/varieng/CoRD/corpora/HelsinkiCorpus (4 April, 2018).