Виноградова Ольга Ильинична Доцент факультета гуманитарных наук Национальный исследовательский университет «Высшая школа экономики» (Москва) olgavinogr@gmail.com

Бузанов Антон Олегович Студент I Курса Национальный исследовательский университет «Высшая школа экономики» (Москва)anton.buzanov.00@gmail.com

Генералова Софья Александровна Студентка I Курса 45 Национальный исследовательский университет «Высшая школа экономики» (Москва) sageneralova@edu.hse.ru

Оверникова Дарья Алексеевна Студентка I Курса Национальный исследовательский университет «Высшая школа экономики» (Москва) daovernikova@edu.hse.ru

Смилга Вероника Константиновна Студентка I Курса Национальный исследовательский университет «Высшая школа экономики» (Москва) smilgaveronika@gmail.com

Сигдел Элина Сушиловна Студентка II Курса Национальный исследовательский университет «Высшая школа экономики» (Москва) sigdelina@gmail.com

**POS tagger evaluation for the automated text analysis and identification of learner error**

**ABSTRACT**. Working with learner corpora requires elaborate NLP techniques such as POS-annotation. In this article a team of computational linguists presents their experience of choosing a POS-tagger for precise and effortless annotation of .txt files with Python3. Russian Error-Annotated Learner English Corpus (REALEC) is the underlying corpora to which text features the POS-tagger has to respond. After identifying four most promising Part of Speech Taggers our team conducted several sets of test and applied various criteria for evaluation of the taggers precision, speed and compatibility with Python scripts that are already used for the research. The description of tests and statistics along with evaluation of POStaggers such as PatternTagger, NLTK, SpaCy and TreeTagger and the conclusion our team arrived at are presented in the following article.

**Keywords**:Learner Corpora; POS-tagger; NLP; POS-tagging approaches; Automated Grammar Checker

Computational linguistics is a rapidly developing field, and the results of research in this area may be used for a wide range of purposes. Moreover, both theoretical and practical applications of this research often require developing software, which is a complicated process where any subsequent step could be complex. An example of such a step is choosing a POS tagger in order to assist in any further text processing, which we will attempt and detail in this study. In our research we relied mainly on the following source: Cambridge Handbook of Learner Corpus Research. In its section Learner Corpora and Natural Language Processing Detmar Meurers provides an extensive description and comparison of the existing methods of NLP applied to learner corpora, including 46 the various approaches to texts’ annotation such as using POS-taggers to assign part of speech tags to tokens (Meurers, 2015). The main goal of this study was to choose a suitable POS tagger to use as the basis for ADWISER (Automated Detection of Writing Inaccuracies for Students of English in Russia), an application being developed in HSE School of Linguistics. The first steps of the procedure were to choose several taggers based on their technical characteristics and usability and to use those taggers to process two sample texts. After that, two human annotators evaluated and compared the results, and the conclusion as to which tagger to use was made. ADWISER, the application that is being developed using the POS tagger chosen as a result of this research, is an automated grammar checker. Unlike applications such as Grammarly, ADWISER will be specifically tailored towards the needs of students who have Russian as their first language, as it will be trained on a corpus of English academic essays by Russian students and will use models designed for targeting Russian-specific L1-influenced grammar mistakes, such as erroneous positioning of discourse-navigating adverbial modifiers (compare and contrast “we also can notice” and “мы также можем отметить”). Part-of-speech tagging is essential for its development and functioning, since the models ADWISER is based on use POS wildcards, and assigning wrong parts of speech to words would dampen the effectiveness of error-scanning the text. Technical evaluation of taggers To create a test sample of taggers, we first investigated their technical availability, looking into taggers which were easy to download and set up regardless of the operational system, as well as possible to be integrated into a Python program. Based on these characteristics, we have narrowed down the list of taggers to three: TextBlob, Spacy and TreeTagger. TextBlob is a Python (2 and 3) library for processing textual data which provides a simple API for diving into common natural language processing tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. It is available for all PC operating systems (Windows, Linux, MacOS), with instructions provided on the official website, and easy to set up using pip installation. Besides POS tagging, TextBlob is also able to perform other tasks, such as lemmatization, translation and sentiment analysis. TextBlob includes two different taggers: Pattern and NLTK, with NLTK as the default one. NLTK Tagger uses NLTK’s TreeBank tagger and requires Numpy to be used, and the PatternTagger uses the same implementation as the pattern library. Similar to tokenizers and noun phrase chunkers, you can explicitly specify which POS tagger to use by passing a tagger instance to the constructor. SpaCy is an open-source software library for advanced Natural Language Processing, written in the programming languages Python and Cython. The library is published under the MIT license and currently offers statistical neural network models for English and other languages. It uses the same tagset as TextBlob. After tokenization, spaCy can parse and tag a given document. This is where the 47 statistical model comes in, which enables spaCy to make a prediction of which tag or label most likely applies in this context. Customising spaCy requires a rather large corpus, with REALEC being perfectly suitable, since as of now, there are 2 million words in the examination essays folder, and combined with essays of graduate students there are texts with total of 3,5 million words. TreeTagger is a POS tagger that was developed in 1995 at the Institute for Computational Linguistics of the University of Stuttgart. The tagger can be trained to process texts in any language as long as there is a lexicon and a manually tagged trainer corpus available, and is installable on numerous operational systems, namely, Windows, PC-Linux, Mac OS-X, ARM64, ARMHF, ARM-Android, and PPC64le-Linux. Moreover, the website provides clear instructions for all operational systems, as well as parameter files for several languages. TreeTagger is available with PENN and BNC tagsets, the latter being a major advantage, as the BNC tagset is extremely comprehensive. Four taggers: evaluation of practical performance Having described the technical characteristics of each tagger, we can now proceed with the practical part of our evaluation. We have chosen at random two student essays from Russian Learner Annotated Corpus -- DOv\_5\_2 and DOv\_18\_2 -- for each tagger to annotate. The first text consisted of 292 words and contained numerous mistakes and typos. The second text consisted of 339 words and was mostly correct grammatically. The performance of each tagger on each text was then checked and evaluated by two human annotators. Then the best applicable tagger was chosen on the basis of annotators’ reviews. Pattern Tagger: results evaluation PatternTagger showed the result of 25 mistakes while annotating DOv\_5\_2 and 22 mistakes while annotating DOv\_18\_2, which means that its percent of efficiency was 93% and 91,5% grammatically incorrect and grammatically correct text respectively. PatternTagger seems not to use context in its POS-analysis at all. The tagger confuses homoforms of such parts of speech as nouns and verbs, adverbs, determinants and other parts of speech. It also tags all capitalized nouns as proper nouns without distinguishing between real proper nouns and the nouns are capitalized because of their position in the beginning of the sentence. What is more, this tagger uses an imperfect PennTreebank system of tags, which lacks some crucially important tags, such as tags for pronouns other than personal ones and tags for 3-rd person plural verbs . These deficiencies result in serious mistakes in tagging, such as possessive pronouns being tagged as adjectives and all the 3-rd person plural verbs being tagged as infinitive forms or non-3rd person singular present verbs. NLTK: results evaluation NLTK tagger showed better results than PatternTagger in tagging both grammatically correct text and text with multiple mistakes and typos. It made 19 mistakes annotating DOv\_5\_2 and 14 mistakes annotating DOv\_18\_2, which 48 yields the percent of efficiency at 92,5% and 95,5% for grammatically incorrect and grammatically correct text respectively. NLTK tagger does a fairly good job using the context to distinguish between homoformic nouns and verbs. The tagger gets less confused by spelling mistakes and typos than PatternTagger, which is also based on TextBlob: it corrects 10 mistakes in POS-tagging made by PatternTagger. However, some spelling mistakes can misdirect even NLTK-tagger: for example, it treats the misspelled adverb “typicaly” as a noun. What is more, despite being able to work with the context, the tagger still confuses adverbs and determinants with other parts of speech. Moreover, just as PatternTagger, NLTK tagger treats all the capitalised nouns as proper nouns, completely disregarding the fact that capitalisation may simply be the result of them being the first words of the sentences. Finally, the tagger uses the same PennTreebank system of tags as the PatternTagger, the imperfections of which have been discussed earlier. Spacy: results evaluation SpacyTagger demonstrated relatively high percents of efficiency in comparison with other taggers evaluated. It made 21 mistakes annotating DOv\_5\_2 and 7 mistakes annotating annotating DOv\_18\_2, which makes its percent of efficiency 93% and 98% for grammatically incorrect and grammatically correct text respectively. However, Spacy still confuses all the types of homoforms possible and gets misguided by capitalised nouns in the beginning of the sentence, just as NLTK and PatternTagger. The efficiency of Spacy can be explained by an extremely primitive system of tags that it uses. Unlike all the other taggers evaluated, the system of tags that Spacy uses provides only basic POS information, having no tags for specific characteristics of any part of speech. The primitiveness of the system of tags leads to crucial mistakes in POS recognition. For example, Spacy, having no specific tags for relative pronouns, tends to tag them as nouns. It seems that the main purpose of the tagger was recognition of the syntactic roles of words in the sentence. Spacy handles this task quite well, but it makes the tagger unsuitable for a more comprehensive POS analysis. TreeTagger: results evaluation Despite being the oldest of all taggers evaluated, TreeTagger seemed to show consistently better results. The tagger made 3 mistakes annotating DOv\_5\_2 and 9 mistakes annotating DOv\_18\_2, which means that one annotator has estimated its effectiveness at 99%, and the other, at 97%. These numbers are not only extremely high, but also do not differ a lot, meaning that, unlike spaCy, TreeTagger is not influenced by spelling mistakes and typos (the only exception being the noun “trend” misspelled as “tend”, which all taggers including TreeTagger treated as a verb). In comparison to other taggers, TreeTagger fared better in distinguishing pronouns from other parts of speech, displayed a better understanding of verb forms. However, it is worth mentioning that these advancements were likely brought about by the comprehensiveness of BNC tags, the tagset that TreeTagger uses. Out 49 of the three tagsets we have dealt with in this study, BNC is the only one which has a highly developed set of tags for pronouns. Furthermore, its tagging system for verbs is extremely extensive, as it contains 25 verb tags, including separate tags for verbs be, do and have. While these may seem redundant at first glance, a tagging system that reflects the diversity of grammatical forms is actually necessary for the tagger to function correctly, which is evident in how TreeTagger seems to overcome the problems other taggers experienced with pronouns. It is necessary to note that TreeTagger is not completely immune to imperfections, the most concerning of them being the flawed tagging system for noun numbers. For example, the tagger treats the word “people” as a collective noun rather than the plural of “person”. Apparently, this might be connected with the fact that “people” can be used as a singular noun and agree with the verb “is”. Other, albeit small, problems include sometimes tagging adjectives as determiners and not always being able to handle possessive forms despite having the tags for it. Nevertheless, what sets it aside from other taggers, even ones as efficient as Spacy, was the tagging system which allowed for deep understanding of the syntactic structure of the text. Conclusion To achieve the goal we set, we have evaluated the technical characteristics of PoS taggers available on the Internet, have run an experiment using two sample texts taken from REALEC, and have evaluated the results using human annotators. As a result of careful consideration, we were able to choose TreeTagger as the PoS tagger for ADWISER. The reasons for choosing it were its availability and easiness to set up, its consistently high efficiency, and that it used a comprehensive set of BNC tags. While the two TextBlob taggers and spaCy are on the same technical level as TreeTagger, we have rejected them based on considerable flaws in their evaluation, which were clearly caused by underdeveloped tagsets. While these criteria may not be extensive, we hope that authors of further research may consider using these when choosing a PoS tagger for their projects. The next steps in our research will be concerned with further development of ADWISER, including creating models for catching grammar mistakes characteristic of those who have Russian as their first language.

References

1. Meurers, D. Learner Corpora and Natural Language Processing. 2015. In The Cambridge Handbook of Learner Corpus Research. Cambridge University Press. 2015
2. Granger, Sylviane, Gilquin, Gaёtanelle, Meunier, Fanny. The Cambridge Handbook of Learner Corpus Research. Cambridge University Press. 2015