THESIS PROSPECTUS: AUTOMATIC KEYPHRASE EXTRACTION FROM RUSSIAN SCHOLARLY PAPERS

Wienecke Y.A., Portland State University
whyves@icloud.com

Abstract

The automatic extraction of keyphrases from scholarly papers is a necessary step for many NLP tasks, including text retrieval, machine translation, text summarization, and the analysis of topic drift. However, due to variations in the linguistic characteristics of natural languages, digital representations of language, and computational processing approaches, automatic keyphrase extraction is a very difficult NLP task that performs worse than other NLP tasks. This research project expands on a previous research project, RusVectores, which analyses the temporal topic drift of scholarly papers published in English from three Russian NLP conferences. The goal of this project is to explore the challenges and efficiency of automatically extracting keyphrases from scholarly papers published in Russian through three approaches: TF—IDF, IDA, and Paragraph Vectors. Additionally, the effects of various approaches for keyphrase candidate selection and morphological normalization on the processing of Russian text will also be explored. The keyphrases and topics automatically extracted from each approach will be tabularly and graphically visualized through the web-based interface proposed by RusVectores.

Keywords: automatic extraction, keyphrase, morphological normalization, TF—IDF, LDA, Paragraph Vector

1. Introduction

Natural Language Processing (NLP) is a field of study that refers to the usage of computers for the automatic processing of natural language. Natural languages, such as English, Spanish, and Russian, are languages that have naturally developed over time for communication between humans, unlike programming languages or artificial languages. Programming languages are created with rigid and restrictive rules, whereas natural languages are complex and have many syntactic rules, semantic rules, and irregularities. The constant evolution of natural language throughout time and the regional and disciplinary variations makes it difficult to systematically process and analyze natural language for information retrieval (IR). Additionally, the expression of natural language differs depending on the medium used to covey meaning; the linguistic patterns in spoken language differ from patterns in written or typed language, and different computational approaches are needed to represent and process these languages. As a result of the variations in natural languages and approaches to computationally represent and process these languages, NLP tasks such as language performance may perform poorly:

> Early simplistic approaches, for example, word-for-word Russian-to-English machine translation, were defeated by *homographs*—identically spelled word with multiple meanings—and metaphor, leading to the apocryphal story of the Biblical, 'the spirit is willing, but the flesh is weak' being translated to 'the vodka is agreeable, but the meat is spoiled [1].'

Research communities primarily focus on the processing of the community's national language [2], but many influential papers in the field of NLP [3], [4], [5], [6] focus on the processing of the English language, and most models for the automatic extraction of keyphrases are adapted towards English texts [7]. This is notable for dealing with the processing of Russian texts because certain stages in the NLP pipeline are language-dependent, requiring special attention to the linguistic characteristics of the language that is being processed [8]. For example, decompounding, which separates compound words into individual words ('snowball' turns to 'snow' and 'ball'), improves the processing of compound rich languages such as Dutch and German, but requires knowledge of which words are compounds and the individual components that factor these compound words. This is a language-dependent process, whereas breaking down text into small chunks of size 'n' (n-grams) is a language independent process [8].

Typically, multiple texts are gathered into a centralized 'corpus,' which may be any combination of general or specific, diachronic or synchronic, monolingual or parallel, and static or dynamic/monitor [9]. General corpora aim to be representative of an entire language, while specific corpora are limited to a specific domain. Diachronic corpora collect text between a specific time span, while synchronic corpora take a snapshot of texts from a certain time. Monolingual corpora include text of a single natural language, while parallel corpora include text from multiple natural languages. Static corpora are of a fixed size, while dynamic corpora grow. NLP approaches for one type of corpora may not perform well for another corpora, and a machine learning model trained on one type of corpora may produce inadequate results when processing text from a different type of corpora [1]. Improving the efficiency and performance of NLP tasks requires consideration of the linguistic and corpus characteristics of the text being processed.

The history of NLP in the English-speaking and Russian-speaking communities traces back to the 1950s [1], [2]. NLP was primarily characterized by theoretical linguistics, but has integrated the disciplines of machine learning, linear algebra, and statistics. Over the past two decades, the performance of NLP approaches has dramatically improved alongside improvements in hardware that enable the analysis of big data and the training of machine learning models. Previously, NLP was done through the usage of proprietary software with a graphical user interface, but current state-of-the-art approaches use programming languages for a finer control and customization of each stage in the NLP pipeline. Popular choices of programming languages for NLP include: Prolog, Python, Java, and R. However, the usage of programming languages leads to irreproducibility of research results for papers that do not provide open access to the source code due to the high level of customization from programming languages and the non-determinism of training for certain machine learning models [10], [11]. The lack of standardization and limitations of computational resources for training machine learning models leads the popularization of using programming language packages, such as python modules, and providing free and open access to the source code, datasets, and trained models used in research papers [12], [13], [11], [14], [15]. This current direction of the NLP community encourages collaboration, code reusability, and rapid prototyping of NLP approaches.

2. Motivation

The Russian Flagship Program at Portland State University is a four-year Russian-intensive program that concludes with a year-long capstone in a Russian-speaking country, in particular: Kazakhstan. During the year abroad, participants take classes to improve their proficiency in the Russian language. Participants take classes in Russian and may do research or work with a local organization in their profession, which requires prior knowledge of domain-specific terminology for communication. In the field of computer science, there are various translations for a single term; in some contexts, the phonetic spelling of a computer science term in Cyrillic is preferred over the Russian translation of said term, for example ‘софтвер’ versus ‘программное обеспечение’ for or ‘юзер’ versus ‘пользователь’. As an aspiring participant of the Russian Overseas Flagship capstone, I would like to understand the keywords and phrases used in Russian-speaking research communities. The goal of this research project is to continue previous work [2] in extracting keyphrases and analyzing temporal topical drifts from scholarly papers published from three Russian NLP conferences. My hope is to obtain a list of the salient domain-specific vocabulary necessary to understand my Russian-speaking peers and participate in a discussion about NLP in Russian.

3. Research Goals

On a high-level, this research project will consist of three steps, which will be detailed in the methods section: (1) creating a corpus, (2) extracting keyphrases from the corpus, and (3) visualizing the output from step 2. This project will follow the methods of Bakarov’s research [2] and will reuse code wherever possible. As such, the project will be developed in python, will use the genism implementations of keyphrase extraction algorithms [16], and will use the web-interface proposed by RusVectores for tabularly and graphically visualizing the extracted keyphrases and topics [17]. In this research paper, the corpus will consist of scholarly papers published in Russian from three Russian NLP conferences: Dialogue[1], AIST[2], and AINL[3]. For this research project, the term ‘documents’ will refer to entire scholarly papers, but can generally be understood in NLP as any context in which words appear, such as sentences or paragraphs. The source code, trained models, and database snapshots used in this research project will be freely and openly available from github. This prospectus paper will first provide a background of the text preprocessing and keyphrase extraction approaches that will be used. The next section will describe related research. Then, the proposed methods for this research project will be listed. The final section will list stretch goals that are out of the scope of this paper but are directions for future work.

4. Background

The goal of this project is akin to the creation of a back-of-the-book index [7], where the main terminology of a section in the book is presented as a list of key words and collocations. This list of keyphrases reflect the main topics and ideas of the section and can be considered as an extremely concise summary [7]. For scholarly papers, some authors provide a list of keywords following the abstract, but this is not standard for every paper. Furthermore, the creation of this keyword list is subjective to the author, so the keywords may be a poor summary of the paper [17]. Automatic methods of keyphrase extraction require a definition of what constitutes a potential keyphrase and how to classify these candidates as keyphrases or non-keyphrases. The features that define a keyphrase may be explicitly provided by the researcher, such as word frequency and uniqueness to the text, or may hidden and implicitly learned by a machine learning model through training. Keyphrase extraction is a necessary step in many NLP tasks, such as text retrieval, text summarization, and opinion mining [18], [19]. While keyphrase extraction is foundational, it is nevertheless one of the most difficult NLP tasks and has worse performance than other NLP tasks [7], [18]. Approaches for the automatic extraction of keyphrases generally follow three stages [20]: identifying a list of words or phrases that are candidate keyphrases, extracting

[1] http://www.dialog-21.ru/

[2] https://aistconf.org/

[3] https://ainlconf.ru/

features from candidate keyphrases, and ranking and selecting the correct keyphrases from candidates.

4.1 The Identification of Keyphrase Candidates

The first stage in the pipeline for keyphrase extraction is identifying a list of keyphrase candidates. This step can be broken down into text preprocessing and keyphrase candidate extraction, and will process a stream of raw text into a stream of n-gram tokens that represent cleaned output for the next stage in the pipeline. The approaches used for this step depends on the language of the text being processed and the keyphrase extraction model that will be used in later stages. For the processing of texts in the English language, minimal preprocessing is required and sometimes even preferred [21]. This is because English is an analytic language [21], which means that syntactic relationships between words is primarily reflected by the usage of word order and helper words. However, languages with a richer morphology or heavy use compounding may require extra steps to prepare the text for later steps in the NLP pipeline [8], [22]. Languages with complex character encodings, such as Chinese and Arabic, and hyphenated text can present additional challenges for parsing a raw text into useful tokens. Errors in this stage of the pipeline will propagate through subsequent stages, so this state has a large impact on the overall performance of the automatic extraction of keyphrases [10], [21].

4.1.1 Text Preprocessing

Typical steps for the preprocessing of text involve stemming, lemmatization, parts-of-speech (POS) tagging and word sense disambiguation (WSD). Stemming and lemmatization fall under the umbrella of morphological normalization, where inflected words are simplified to their basic, ‘dictionary’ form. Stemming is a simple form of morphological normalization that involves removing suffixes and/or prefixes and may produce non-existent words. Lemmatization is a more intense process and involves the usage of a trained model (machine-learning approach) or lexical database (knowledge-based approach) to reduce a given word to its dictionary form [23]. Lemmatization may have a prerequisite processing step of POS tagging, where a different trained model or lexical database is used to assign each word a token that reflects the word’s grammatical role in the sentence. Coarse-grained POS tagging involves identifying the part-of-speech of a word, such as noun, adjective, and verb, while fine-grained POS tagging includes additional syntactical information, such as case, gender, and number [22]. POS tagging may also be a prerequisite for WSD for the disambiguation of homonyms, which are words that have multiple possible meanings. For example, the homonym ‘fly’ may refer to a noun or a verb, and if a verb, the number may be singular or plural. For morphologically rich languages such as Russian, a simple sentence can have a large amount of potential tag variants [22]. As an example of the outputs after morphological normalization, the sentence “the geese went wherever mother goose goes” after stemming may return “the gees went wherev mother goos goe.” Given the same sentence with the following POS tags: “DET NOUN VERB ADV NOUN NOUN VERB,” a lemmatizer may return “the goose go wherever mother goose go.” Morphological normalization reduces the occurrences of word form variations, which may improve the performance of certain NLP tasks. However, normalized words lose the meaning intrinsic in form inflection, which can introduce ambiguity and negatively impact performance for NLP tasks that depend on information about word inflection [21]. The data preprocessing stage can involve a mixture of steps to process raw text input into output tokens, but each preprocessing step may introduce errors or ambiguity, which may impact the performance of later stages in the NLP pipeline.

4.1.2 The Extraction of Keyphrase Candidates

The automatic extraction of keyphrases candidate from a text can be accomplished through a variety of heuristics, such as filtering out words that do not meet an minimum length threshold, filtering out text in a different language, removing stop words, chunking certain parts-of-speech sequences, and n-grams [8], [20]. Stop words are commonly used words that have little meaning or are clearly not keyphrases, and a list of language-dependent stop words must be explicitly defined or imported from a programming language module. For example, the sentence “The Catcher in the Rye” may be reduced to simply “Catcher Rye.” Chunking parts-of-speech sequences uses the information gathered from the POS tagging step and extracts, for example, sequences of adjacent nouns and adjectives as keyphrase candidates. N-grams involve focusing on a sequence of tokens through a sliding window of n-sized chunks, where chunks may be bytes, characters, or words. In the case of using word n-grams, a unigram refers to a single word, a bigram refers to two adjacent words, and an n-gram refers to n sequential words. A common way of representing the relationship of keyphrase candidates to documents is the bag-of-words or bag-of-n-grams model, where a corpus is characterized by a document-keyphrase frequency matrix with each cell containing the frequency of a keyphrase in a document. The resulting output of this stage is some intermediate representation that captures the relationship of keyphrase candidates to documents, but keyphrases must still be ranked in order to determine which candidates are actual keyphrases.

4.2 Feature Extraction

The next stage in the pipeline is feature extraction, where an input representation of keyphrase candidates is ranked and processed into an output representation of keyphrase candidate vectors. Under the vector space model for automatic indexing [24], each candidate keyphrase will be represented as a unit vector that has a dimensionality equivalent to the number of features that will be used for judging the importance of each candidate. Features can be grouped into four categories: statistical, syntactical, structural, and external [18], [20]. The simplest features are statistical features, which include word length and word frequency. The most prevalent statistical feature is Term Frequency—Inverse Document Frequency, which is the product of the frequency of the keyphrase candidate in a document and the inverse frequency of the candidate appearing in other documents [3]. That is, a keyphrase is more important if it frequently appears in a document, but infrequently appears in other documents. Syntactical features assign importance to a keyphrase according the grammatical arrangement of the keyphrase, such as POS sequences or suffix sequences. Structural features relate to the location in a document in which a keyphrase candidate appears, such as in the title or abstract versus in a body paragraph. External factors involve the usage of resources outside of the corpus, like calculating a frequency ratio according to occurrences on Wikipedia pages [18]. The features mentioned here are explicitly defined, but recent unsupervised approaches to features extraction mentioned in the next paragraph may use machine learning to discover latent, ‘hidden’ features that are not explicitly provided or known beforehand [25]. The importance of this stage is to associate keyphrase candidates with enough features to make an informed decision about whether the candidate is an actual keyphrase or not.

4.3 Ranking and Selecting Keyphrase Candidates

Once each keyphrase candidate is represented as a vector of features, the next stage in the pipeline is to rank and choose actual keyphrases from the candidates. There are various approaches to rank and select keyphrases, including the usage of any combination of heuristics, graph algorithms, or machine learning models. This is the final stage of the NLP pipeline for the automatic extraction of keyphrases, and the expected output is a list of keyphrases, which may be limited to a list of the highest ranked ‘N’ keyphrases. Some approaches compare the semantic similarities of keyphrases and groups the keyphrases into ‘k’ groups, with each group representing some abstract topic that unites keywords.

4.3.1 Heuristic Approach

The simplest approaches of keyphrase candidate ranking and selection use heuristics to rank keyphrases by features. A bag-of-words or bag-of-n-grams model that weighs keyphrase importance through TF—IDF is one extremely simplistic heuristic that can outperform other approaches for a small, domain-restricted corpus [2], and will be used as the first approach for the automatic extraction of keyphrases in this research project. This approach involves sorting the document-keyphrase frequency matrix (bag-of-words/bag-of-n-grams) by highest TF—IDF score and selecting a list of top N keyphrases. Without morphological normalization from the preprocessing stage, TF—IDF performs poorly on Russian texts because each word in Russian has many forms and each form has a low frequency in each text, so a frequently occurring candidate may turn into multiple non-frequently occurring candidates [19]. A slightly more complex heuristic introduces the idea of topic modeling, where a document is represented as a set of k topics and each topic is characterized by N keyphrases. The parameter ‘k’ is chosen by the programmer, therefore is a ‘hyperparameter,’ and can be evaluated and optimized by either human judgement, perplexity [4], or some other topic coherence measurement [26]. These models assume that semantically related keyphrases appear in the same document [27], so documents with similar topics will have a similar set of keyphrases. For example, in the purely statistical topic modeling approach of Latent Semantic Analysis (also referred to as Latent Semantic Indexing, LDA/LSI), a document-keyphrase TF—IDF matrix is factored into a latent semantic space through the dimensional decomposition method Singular Value Decomposition (SVD) [28]. This semantic space is composed of three TF—IDF weighted matrices: a document-topic matrix, a topic-topic square matrix, and a keyphrase-topic matrix [29]. The document-topic and keyphrase-topic matrices can be sorted by highest TF—IDF score and joined to obtain the top N keyphrases for k topics in each document of the corpus.

4.3.2 Graph Approach

TF—IDF and LDA/LSI are purely statistical methods for ranking and selecting keyphrases. More complex approaches for this stage of the NLP pipeline involve the usage of graph and machine learning models. The graph model uses the ranking method TextRank [6], which is inspired by the early google algorithm for ranking a page, PageRank [30]. In the TextRank approach, a document is represented as a graph with keyphrases candidates as vertices and keyphrases candidate co-occurrences within a certain word distance window as edges that connect vertices. Adjacent vertices ‘recommend’ each other, so the importance of a keyphrase candidate vertex is determined by its rank, which is the number of edges it has, and the rank of adjacent vertices that recommend it [6]. Although the usage of graph models for the automatic extraction of keyphrases in documents published in Russian was quite popular as of 2011 [7], it is not an approach that will be used in this research project.

4.3.2 Supervised Machine Learning Approach

Machine learning approaches can be divided into supervised or unsupervised approaches. Supervised approaches for classifying keyphrases require a dataset that is tagged with known keyphrases beforehand, such as the list of keywords that some scholarly papers provide. Using this marked dataset, a machine learning model repeatedly tries to classify a candidate keyphrase according to some number of hidden features and slightly adjusts the hidden feature parameters. The machine learning model becomes increasingly better at correctly identifying a keyphrase, thus 'learning' through training on the dataset. Depending on the type of machine learning model used for training, classification can be done through a generative or discriminatory approach. Discriminatory classifiers are more accurate, but usually take longer to train and thus require more training data. These classifiers work best with large training datasets and learn the boundaries between classifications. A popular discriminatory model may use linear regression to converge to a local or global maximum classification accuracy. Generative classifiers are overall less accurate, but train faster, require less training data, and develop a foundational understanding of each classification in order to differentiate. The naïve-Bayes model is a popular probabilistic generative classifier that assumes that features are independent of each other. Once trained, these classifiers are able to create new documents that match a certain classification, which is the explanation for how the classifier is 'generative' [31].

4.3.3 Unsupervised Machine Learning Approach

Like supervised models, unsupervised models may also be discriminatory or generative, and they do not require a tagged dataset. However, while TF—IDF, LSA, and TextRank are all unsupervised methods, these methods do not involve the usage of a machine learning model.

4.3.4.1 Probabilistic LSA (pLSA/pLSI)

An example of an unsupervised machine learning model is probabilistic LSA (pLSI, also referred to as the aspect model), which is an alternative to LSA and is characterized as a generative statistical model. This model learns the probability distribution of keyphrases in a document, as calculated by TF—IDF, to understand how to classify keyphrases and generate novel keyphrase documents through statistical inference [28]. This model represents each document as a fixed set of topics, with each word in the document having been generated by one topic. However, pLSI does not provide a probability distribution of topics over documents, which means that each document can have completely different topics and that there is no way to assign probabilities to a document not in the training set. This leads to a linear growth of total topics in a corpus with respect to the number of documents, and overfitting, which is when the model becomes excellent at classifying documents from the training set, but poor at classifying new documents [4].

4.3.4.2 Latent Dirichlet Allocation (LDA)

The weaknesses of pLSA form the basis for Latent Dirichlet Allocation (LDA) [4], a generative probabilistic model that is a standard for topic modeling and will be used in this research project. LDA solves the problem of linear total topic growth and overfitting by introducing two hidden corpus-level parameters, the Dirichlet priors. One parameter is associated with the probabilistic distribution of topics over documents, such that all documents are a mixture of k topics. The other parameter is associated with the probabilistic distribution of all words over these k topics. In summary, for LDA, all documents in a corpus are represented as a mixture of probabilities of k topics, where each keyphrase in a document is generated by one the k topics. In order to avoid assigning new words from a document outside of a training set a probability of zero, an additional method called ‘smoothing’ is used to ensure that all words get a positive probability [4]. Whereas LSA used SVD to create a latent semantic space, from which a list of topics with related keyphrases are created, LDA can use statistical inference and machine learning to generate a list of topics with related keyphrases. LDA is an approach that generates a list of topics and keyphrases from a domain, but it works off the assumption that the topic distribution and keyphrase distribution for all documents in a corpus are similar. However, for creating a more generalized automatic keyphrase extractor, this assumption does not hold, especially as a data set grows and changes over time. Research in crafting hierarchical LDA address this limitation by generating topic distributions from samples at each step from the root of a tree to a leaf [32].

4.3.4.3 Statistical Language Models

Other research unrelated to LDA that attempt to address the problem of the automatic extraction of keyphrases from multiple domains expands topic modelling into statistical language modelling [33]. Language models attempt to capture context by representing a document as a probability distribution of word sequences, which may be single words or n-grams. One approach of using language models for the automatic extraction of keyphrases involves training a language model on a foreground corpus, which are the documents from which to extract keyphrases, and a separate language model on a background corpus, which is a more general corpus, in order to determine the importance of a keyphrase candidate. For example, a foreground corpus may consist of research papers from one conference and the background corpus may consist of research papers in general [33]. In this approach, the language models learns to judge keyphrase candidates according to two abstract features: phraseness, which is the how much a candidate can be considered a phrase, and informativeness, which is how much a candidate summarizes the key ideas of a set of documents [33]. However, as the size of the corpus vocabulary grows, the number of dimensions of the language model increases exponentially, which is referred to as the curse of dimensionality [34], [35].

4.3.4.4 Neural Net Language Models

To address the curse of dimensionality problem encountered with statistical language modeling, researchers designed the Feedforward Neural Net Language Model (NNLM) [35], mimicking the organization of neurons in the human brain. NNLMs are graphs that are characterized by a layer of vertices that take word feature vectors as an input, a layer of vertices that create an output, and an intermediate number of ‘hidden’ layers that map the input layer to the output layer, and each layer may contain a different number of vertices. NNLMs solve the curse of dimensionality by learning to represent words as a low dimensional vector, also referred to as a word embedding, whereby similar words are represented by similar vectors [34], [5]. Recurrent NNLMs (RNNLMs) [36] introduce loops that connect the hidden layer to itself, acting as a short-term memory. While NNLMs and RNNLMs greatly improve the performance of various NLP tasks, the introduction of the hidden layer makes training a slow and computationally expensive process due to dense matrix multiplications [5]. Mikolov’s research in [5] and [37] dramatically reduced this limitation by removing the hidden layer and training a NNLM on simpler models that are less accurate, but quicker to train. This research produced two similar models: the Continuous Bag-of-Words (CBOW) and Continuous Skip-gram models. The prior tries to guess a word by using the context of surrounding words, without placing importance on the order of words, much like the bag-of-words representation of documents. The latter, however, tries to generate a context of words, given a single word in the same sentence. Later implementations of CBOW and Skip-gram generalize from learning word embedding to learning sentence embeddings and document embeddings, such as the Paragraph Vector algorithm, also known as doc2vec [15]. This research project will use doc2vec, which assigns an id to a paragraph and uses this paragraph id in the CBOW and Skip-gram algorithms. For CBOW, a paragraph id and set of words are used to guess a missing word, while for Skip-gram, a paragraph id is used to generate a set of words [37]. Current state-of-the-art approaches use a more complex version of NNLMs called a Deep Neural Net Language Model (DNNLM) [38], but they are outside of the scope of this paper. DDNLMs have multiple hidden layers and are supposed to work best with as little preprocessing as possible, because extra hidden layers theoretically are able to handle spelling and morphological variations [21]. However, while this is true for the processing of English text, lemmatization is shown to improve performance for Russian text [21]. NNLMs and DDLMs work better with a large data set to train upon [37], which can be a problem for underrepresented languages or domains in the NLP space. There is overall less text easily accessible for training and validating language models in the Russian than in English [19], [21], [39].

4.4 Performance Evaluation

The evaluation of the performance for the automatic extraction of keyphrases consists of two steps: (1) comparing the extracted keyphrases with a gold standard list of keyphrases, and (2) calculating the precision, recall, and F-score of the output [18]. For the first step, the simplest yet most time-consuming option is to use human judgement. An alternative is to use an algorithm to automatically directly compare the extracted keyphrases with the gold standard to find exact matches, but this may return false negatives for keyphrases that are slightly different due to spelling, morphology, or being off by a word (ex. ‘keyphrase extraction’ & ‘automatic keyphrase extraction’). In order to consider partial matches, [18] proposes the usage of machine translation metrics. For the second step, the F-score (also known as F1-score) represents the overall accuracy of the keyphrase extraction, and it is calculated from precision and recall. Precision is the percentage of correct keyphrases out of all keyphrases extracted, while recall is the percentage of correct keyphrases extracted out of all keyphrases that should have been extracted. The author provided keyword list may function as the gold standard list of keyphrases, but this presents two problems for the Russian-language scientific papers. In particular, not all scientific papers have an author provided keyword list, and not all keyword lists are in Russian or provide a Russian translation. The purpose of this research project is to explore the implementation of various approaches to the automatic extraction of keyphrases from Russian text, therefore the numerical evaluation of the performance for each approach is out of the scope for the limitations listed above. For this reason, performance will be judged from a purely qualitative standpoint.

5. Related work

This research project is inspired by a previous paper concerned with the automatic extraction of keyphrases from English-language papers published from three Russian NLP conferences: Dialogue, AIST, and AINL [17]. The work in [17] is part of an ongoing project to create a search engine specifically for Russian NLP papers [2] and a free and open-source toolkit for utilizing and visualizing distributional semantic models [40]. This research project will expand upon the methods used in these papers and will focus exclusively on papers published in Russian.

In [2] and [41], extracted keyphrases are used to compare topics and keywords throughout time and between different conferences. This involves training separate machine learning models on conference papers for each year of a conference or training separate machine learning models on papers from each conference. After extracting keyphrases with these models, the topic distributions are graphically and tabularly visualized and statistically compared to provide inside into the similarities of topics in a year and between years. [14] and [10] explore the impact of preprocessing and keyphrase extraction approach on the performance of the automatic extraction of keyphrases, and offer a standardized python module for the benchmarking these variations.

6. Methods

1. Create a corpus of NLP papers published in Russian from Dialogue, AIST, and AINL.
    a. Download all papers published from Dialogue, AIST, and AINL.
    b. Extract metadata and text from each paper with the pdfminer3[4] and/or Neural-ParsCit[5] python modules:
        i. Title, abstract, keywords, references and author names, affiliations, emails.
        ii. Body text, headers, abstract, introduction, related work, conclusion.
        iii. Exclude: tables, figures, captions, equations, notes, copyright, references.
    c. Use the langid[6] python module to detect paper language.
    d. Filter out papers not written in Russian.

2. Preprocess documents
    a. Use UDPipe for POS tagging [13].
    b. Use either Yandex's pymystem3[7] or pymorphy2[8] python module for lemmatization.

3. Extract keyphrase candidates from each document.
    a. Remove stop words.
    b. Extract n-gram-of-words from each document: unigrams, bigrams, and trigrams.

4. Rank and select keyphrases using genism[9] python module implementations.
    a. Optimize the hyperparameter k for minimum perplexity.
    b. Use TF—IDF, LDA, and Paragraph Vector (doc2vec).
    c. Select N top keywords.

5. Use the RusVectores methods for visualizing topics distributions and keywords.

7. Stretch Goals

- Compare the drift of conference topics per year and between conferences.
- Use numerical evaluation methods to evaluate and compare the performance of each keyphrase extraction model.
- Compare topics and keywords between Russian-language and English-language papers.
- Apply this automatic keyphrase extraction technique with conferences for other professions.

[4] https://pypi.org/project/pdfminer3/
[5] https://github.com/WING-NUS/Neural-ParsCit
[6] https://github.com/saffsd/langid.py
[7] https://pypi.org/project/pymystem3/
[8] https://github.com/kmike/pymorphy2
[9] https://pypi.org/project/gensim/

References

[1] P. M. Nadkarni, L. Ohno-Machado and W. W. Chapman, "Natural language processing: an introduction," *J Am Med Inform Assoc,* vol. 18, no. 5, pp. 544-551, Sep.-Oct. 2011.

[2] A. Bakarov, A. Kutuzov and I. Nikishina, "Russian Computational Linguistics: Topical Structure in 2007-2017 Conference Papers," in *Dialogue-2018*, Moscow, 2018.

[3] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation,* vol. 28, no. 1, pp. 11-21, 1972.

[4] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research,* vol. 3, pp. 993-1022, 3 Mar. 2003.

[5] T. Mikolov, K. Chen, G. S. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *Proceedings of the International Conference on Learning Representations*, 2013.

[6] R. Mihalcea and P. Tarau, "TextRank: Bringing Order into Texts," in *Conference on Empirical Methods in Natural Language Processing*, Barcelona, 2004.

[7] А. С. Ванюшкин and Л. А. Гращенко, "Методы и алгоритмы извлечения ключевых слов," *Новые информационные технологии в автоматизированных системах,* no. 19, pp. 85-93, 2016.

[8] J. Kamps, C. Monz, M. Rijke and S. B, "Language-dependent and Language-independent Approaches to Cross-Lingual Text Retrieval," in *Comparative Evaluation of Multilingual Information Access Systems*, Berlin, 2014.

[9] A. Gries, "What is Corpus Linguistics?," *Language and Linguistics Compass,* pp. 1225-1241, 5 Mar. 2009.

[10] F. Boudin, H. Mougard and C. D, "How Document Pre-processing affects Keyphrase Extraction Performance," in *Proceedings of the 2nd Workshop on Noisy User-generated Text*, Osaka, 2016.

[11] M. Fares, A. Kutuzov, S. Oepen and E. Velldal, "Word vectors, reuse, and replicability: Towards a community repository of large-text resources," in *Proceedings of the 21st Nordic Conference of Computational Linguistics*, Gothenburg, 2017.

[12] M. Lui and T. Baldwin, "angid.py An Off-the-shelf Language Identification Tool," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju, 2012.

[13] M. Straka and J. Strakova, "Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe," in *Proceedings of CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, Vancouver, 2017.

[14] F. Boudin, "pke: an open source python-based keyphrase extraction toolkit," in *Proceedings of the 26th International Conference on Computational Linguistics: System Demonstrations*, Osaka, 2016.

[15] Q. Le and T. Mikolov, "2014," in *Proceedings of the 31st International Conference on Machine Learning*, Beijing, 2014.

[16] R. Rehurek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Valletta, 2010.

[17] I. Nikishina, A. Bakarov and K. A, "RusNLP: Semantic search engine for Russian NLP conference papers," in *Proceedings of AIST-2018*, Moscow, 2018.

[18] K. Hasan and V. Ng, "Automatic Keyphrase Extraction: A Survey of the State of the Art," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, 2014.

[19] С. О. Шереметьева and П. Г. Осминин, "Методы и модели автоматического извлечения ключевых слов," *Bulletin of the South Ural State University. Ser. Linguistics,* vol. 12, no. 1, pp. 76-81, 2015.

[20] О. С. Недильченко, "Этапы и методы автоматического извлечения ключевых слов," *Молодой учёный,* vol. 22, no. 156, pp. 60-62, Июнь 2017.

[21] A. Kutuzov and E. Kuzmenko, "To Lemmatize or Not to Lemmatize: How Word Normalisation Affects ELMo Performance in Word Sense Disambiguation," in *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*, Turku, 2019.

[22] A. A. Sorokin, "Improving Neural Morphological Tagging using Language Models," in *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conderence "Dialogue 2018"*, Moscow, 2018.

[23] A. R. Pal and D. Saha, "Word Sense Disambiguation: A Survey," *International Journal of Control Theory and Computer Modeling (IJCTCM),* vol. 5, no. 3, July 2015.

[24] G. Salton, A. Wong and C. S. Yang, "A Vector Space Model for Automatic Indexing," *Information Retrieval and Language Processing,* vol. 18, no. 11, pp. 613-620, 1975.

[25] J. G. Dy and C. E. Brodley, "Feature Selection for Unsupervised Learning," *Journal of Machine Learning Research,* vol. 5, pp. 845-889, 2004.

[26] H. M. Wallach, I. Murray, S. R and M. D, "Evaluation Metrics for Topic Models," in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, 2009.

[27] P. W. Foltz, "Latent semantic analysis for text-based research," *Behavior Research Methods, Instruments, & Computers,* vol. 28, no. 2, pp. 197-202, Feb. 1996.

[28] T. Hofmann, "Probabilistic Latent Semantic Indexing," in *SIGIR '99*, Berkley, 1999.

[29] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman, "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science,* vol. 41, no. 6, pp. 391-407, Sep. 1990.

[30] L. Page, S. Brin, R. Motwani and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," *Stanford InfoLab,* Nov. 1999.

[31] A. Ng and M. I. Jordan, "On Discriminative vs Generative classifies - A comparison of logistic regression and naive Bayes," in *Advances in neural information processing systems*, 2001.

[32] T. L. Griffiths, M. I. Jordan, J. B. Tenenbaum and D. M. Blei, "Hierarchical Topic Model and the Nested Chinese Restaurant Process," in *Advances in neural information processing systems*, 2004.

[33] T. Tomokiyo and M. Hurst, "A Language Model Approach to Keyphrase Extraction," in *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment*, 2003.

[34] K. Jing, J. Xu and B. He, "A Survey on Neural Network Language Models," 2019.

[35] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, "A Neural Probabilistic Language Model," *Journal of Machine Learning Research,* vol. 3, pp. 1137-1155, 2003.

[36] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[37] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, Lake Takoe, 2013.

[38] E. Arisoy, T. N. Sainath, B. Kingsbury and B. Ramabhadran, "Deep Neural Network Language Models," in *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, 2012.

[39] T. Kenter, L. Jones and D. Hewlett, "Byte-Level Machine Reading across Morphologically Varied Languages," in *The Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[40] A. Kutuzov and E. Kuzmenko, "WebVectores: a toolkit for building web interfaces for vector semantic models," in *International Conference on Analysis of Images, Social Networks and Texts*, Yekaterinburg, 2016.

[41] S. Gollappi and X. Li, "EMNLP versus ACL: Analyzing NLP Research Over time," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, 2015.