Big Data Approach to Developing Adaptable Corpus Tools

Andriy Lutskiv[1][0000-0002-9250-4075], Nataliya Popovych[2][0000-0001-6949-0771]

[1] Computer Systems and Networks Department
Ternopil Ivan Puluj National Technical University, Ternopil', Ukraine

l.andriy@gmail.com
[2] Department of Multicultural Education and Translation
State University "Uzhhorod National University"
Uzhhorod, Ukraine

nataliya.popovych@uzhnu.edu.ua

Abstract. Thesis deals with the development of corpus tools which allow building corpus of religious and historical texts. It is foreseen that the corpus has the features of data ingestion, text data preprocessing, statistics calculation, qualitative and quantitative text analysis. All these features are customizable. With Big Data approach is meant that corpus tools are treated as the data platform and the corpus itself is treated as a combination of data lake and data warehouse solutions. There have been suggested the ways for resolving algorithmic, methodological and architectural problems which arise while building corpus tool. The effectiveness of natural language processing and natural language understanding methods, libraries and tools on the example of building historical and religious texts' corpora have been checked. There have been created the workflows which comprise data extraction from sources, data transformation, data enrichment and loading into corpus storage with proper qualitative and quantitative characteristics. Data extraction approaches which are common for ingestion into data lake were used. Transformations and enrichments were realized by means of natural language processing and natural language understanding techniques. Calculation of statistical characteristics was done by means of machine learning techniques. Finding keywords and relations between them became possible thanks to the employment of latent semantic analysis, terms and N-gram frequencies, term frequency-inverse document frequencies. Computation complexity and number of information noise were reduced by singular value decomposition. The influence of singular value decomposition parameters on the text processing accuracy has been analyzed. The results of corpus-based computational experiment for religious text concept analysis have been shown. The architectural approaches to building corpus-based data platform and the usage of software tools, frameworks and specific libraries have been suggested.

Keywords: adaptable text corpus, Big Data, natural language processing, natural language understanding, statistics, machine learning, data mining, conceptual analysis, corpus-based translation studies, conceptual seme, componential analysis.

1 Introduction

Corpus-based approach for comparative translation studies analysis of religious and historical text units is to be done with a proper tool. Corpus allows to demonstrate typical and outline unusual or less frequently used lexico-semantic expressive means on a large amount of text data and underline rare or exceptional cases. Corpus-based analysis also makes both quantitative and qualitative analysis on the large amount of data possible. Comparative translation studies analysis of religious and historical texts is aimed at verifying possible inadequacies in translation by means of different translation studies approaches and theories of which theories of equivalence are the most effective ones. Corpus-based translation studies analysis enables to analyze a large amount of text data with better accuracy, higher efficiency, adaptable functionality, less ambiguity and within significantly smaller time frames. The whole workflow process is customizable. Building corpus can be done with the help of a proper corpus tool which can be developed using a combination of corpus linguistics and Big Data processing approaches. To process these data there should be used methods and tools relevant to Big Data and data analytics subject areas, such as machine learning, statistics and neural networks. The key feature of this system should be computational effectiveness. Thus, corpus tool can be treated as a data platform which allows ingesting, storing, processing and performing analysis.

Adaptable corpus comprises the features of parallel, multilingual, synchronic and diachronic corpora or corpus tool [1].

The aim of this article is to suggest the software tool which can comprise the quality or effectiveness, option-based functionality and will be task-based or adaptable to specific linguistic research tasks with higher level of big data workflow capacity and less ambiguity indicator.

2 Briefly on classifications of corpora and corpus tools

There are many classifications of corpora and software tools in today’s corpus linguistics literature. Kennedy G. in his Introduction to Corpus Linguistics (1998) gives corpora classification focused on design, development and their generations and is highly recommended as a basic beginner reference book in corpus linguistics. Lee D. and Weisser M. suggest very good and substantial classifications of corpus tools and corpora [2, 3]. Our classification is focused more on final user and divides corpus tools and corpora into five main groups. We classify them according to their functional features or properties, i.e., (A.) content-based, (B.) functionality-based, (C.) aim- or purpose-based, (D.) adaptable or of mixed features and functions and (E.) one task-based corpus software tools. Similar classification was given in our recent article on adaptable corpus tool. The generation-based type of corpora and corpus tools is not taken into account in this investigation [4]. One-task-based corpus software tools type is added to the classification.

This type comprises a great number of different linguistic tools, i.e., so-called corpus software tools directed toward the accomplishment of one task either linguistic or statistic in its nature. Among them are offline and web-based concordancers, text coding, (manual) annotation programs, text-analysis tools & search engines, tools & resources for transcribing, annotating or analyzing texts (inc. speech or audio-visual), stats tools and effect size calculators, taggers and POS taggers etc. [4] Within this group are offline and web-based concordancers like AntConc (v.3.5.8, February 18, 2019) , WordSmith Tools (v. 7, 2019), #LancsBox (v 4.0, 2018), JConcorder (ver. 1.beta.13, 2011), text coding, (manual) annotation programs, text-analysis tools & search engines like DART (ver. 3.0, 2019), Dexter and tools & resources for transcribing, annotating or analyzing texts (inc. speech or audio-visual) like CLaRK, ELAN (EUDICO Linguistic Annotator), GATE (General Architecture for Text Engineering), stats tools like Log-likelihood and effect size calculator, taggers like CLAWS, Stanford POS tagger and others [4-11, 15-17].

It is also worth mentioning that the Computational Linguistics in Ukraine has its long history and the issues of corpus-based language research have their systematic development that resulted in the works of the outstanding scholars Perebyjnis V., Klymenko N. [12], Karpilovska E., Dartchuk N. and others.

Besides different specialized dictionaries published by these Ukrainian linguists, there are also Ukrainian National Corpora which have been developed in several projects realized till nowadays and some of them are in the process of their development. Corpus of the Ukrainian Language (N. Dartchuk, O. Siruk, M. Langenbach, Ya. Khodakivska, V. Sorokin at the Institute of Philology of TKU of Kyiv) [13], Labratory of Ukrainian (Ukrainian) [14] and General Regionally Annotated Corpus of Ukrainian (GRAC) (Ukrainian) [15], to name just a few which are the most developed of the Ukrainian language corpora and corpus tools.

3 Main goals of building adaptable corpus for the analysis of religious and historical texts

Concepts, terms and notions of the religious and historic texts require interdisciplinary approach to their study. Lexicology, Cognitive Linguistics, Semantics, especially Conceptual Semantics and Cognitive Semantics, Semasiology, Neurolinguistics, Philosophy of Language and Pragmatics study the notion of concept from various points of view and are connected to Translation Studies by means of equivalence or adequacy theories. The latter focus on translation unit equivalence/adequacy. Such approaches to the semantic or conceptual meaning of the lexical unit as (1) componential analysis, (2) semantic triangle theories, (3) system of values theory and (4) conceptual analysis represent a set of criteria to translation quality assessment of the religious concepts.

Conceptual analysis is a type of approach applicable to study and define the concept relations and systems. The idea "*of concept system, which is one of the most central theoretical notions in the theory of terminology, is usually defined in terminological literature as a system of related concepts which form a coherent whole. Starting from the idea of system, concept systems could be regarded as systems consisting of several components (concepts) and their relations (concept relations). They are mental, i.e. abstract, artificial, theoretical, man- made systems. They are static because they represent the conceptual apparatus reflecting the knowledge which exists at a particular time. New data result in new concepts, and the emergence of new concepts changes existing concept systems as has repeatedly happened" in the history of different sciences*" [19].

Corpus-based conceptual analysis is a type of approach applicable to study and definition of concept relations, concept systems and the place of conceptual seme, i.e., the smallest meaningful particle in the system of concept relations and systems by means of an adaptable text corpus tool able to analyze a large amount of text data (BigData). It should be taken into account that any concept belongs to the system of concepts and has its relations. And a conceptual seme as a smallest conceptual unit has its important functional role in that big system of concepts.

3.1 Ukrainian translations of the Bible

It is very important to understand the context in which the concept is used: history, translations and cultural influences. Facts, which form the context for the Bible translation, play a very important role in the understanding of the Ukrainian translations of the Holy Bible.

There are few Ukrainian translations of the Bible which were made by Panteleimon Kulish, Ivan Pulu'j and Ivan Nechuy-Levitsky (1903) [20], Olexandr Gyzha (2006) [21], Ivan Ohiyenko (1962, 1988) [22], Rafail Turknonyak(2011)[23] and others. All these translations are not literal but literary translations. As an example of adaptable corpus content there were taken six editions of the Bible in three languages (English [24-26], Ukrainian [21-22] and Russian [27]) to create parallel searchable corpus for linguistic research goals.

The Ukrainian language which was used by translators is the language of different years, i.e., 1872-1903, 1966, 1988, 2006, 2011. Translators are influenced by countries of their origin and not all of them lived in Ukraine while translating the Bible.

3.2 Basic mathematical representation of the translation process

If we treat translation unit (term, word combination, set expression) x which belongs to set X_{source} of source language text as a translation unit and target text unit y which belongs to set Y_{target} of translated units, than translation can be treated as a matching between the elements of the sets X_{source} and Y_{target}:

$$F_{translation} : X_{source} \rightarrow Y_{target} \tag{1}$$

But if there is no literal translation used in the process of translation… and one word is translated by means of equivalent word, few words or none, than we can describe translation $\Phi_{translation}$ as matching graph between sets X_{source} and Y_{target}. Hence, translation can be treated as a matching φ of the elements of the sets X_{source} and Y_{target}:

$$\varphi \overset{\text{def}}{=} \langle \Phi_{translation}, X_{source}, Y_{target} \rangle, \Phi_{translation} \subseteq X_{source} \times Y_{target} \tag{2}$$

Translation of each translation unit can be written as:

$$\begin{aligned}&(x \in X_{source}) \xrightarrow{\Phi} (y \in Y_{target}) \vee \\ &\vee (\{y_1, y_2, \dots, y_n\}, y_i \in Y_{target}) \vee (y \in \varnothing)\end{aligned} \tag{3}$$

But due to translation of historical, religious and other ancient texts' analysis linguist faces the problem of the unknown sources or intermediaries of the source languages. When translators are making literary translations they have the possibility to compare the Bible translations made in different languages (e.g. while translating the Bible into Ukrainian in some translations there could have been used the Old Slavic, the Ancient Greek or the Ancient Hebrew texts as the sources). Thus, set X_{source} comprises of few subsets:

$$X_{source} = \{X_1, X_2, \dots, X_n\} \tag{4}$$

While working with translations from ancient languages such as Ancient Greek into modern languages (e.g., English, Ukrainian, Russian, German etc.), we are facing the problem of the absence or incorrect named entity recognition (NER), stop words, lemmatization etc. These problems can be solved by preparing special dictionaries and written rules for lemmatizers by linguists in the collaboration with computer engineers.

All these facts should be taken into account while developing customizable corpus software. Proper and customizable corpus should suggest for professional linguist a set of different options. Among these options could be:

1. **Different types of text data ingestion** can be fully automated, semi-automated or manual. For example, some texts cannot be automatically ingested into corpora because of text recognition problem (e.g. ancient texts), some texts have improper symbols in the text format (Ukrainian "i" is substituted by a Latin), some languages may be not supported by modern natural language processing (NLP) libraries and so on.
2. **Text processing** is aimed at dividing text into paragraphs, sentences and word tokenization, finding collocations (words co-occurrences) and colligations (words co-occurrences in some special grammar constructions), term lemmatization (grouping

together the inflected terms for further analysis as a single item), concordance (search word in a context), Part-of-speech tagging.

3. **Semantic tagging** of each part of the corpus (e.g. UCREL Semantic Analysis System [28]). This process is based on results that are obtained from the previous stage. This option allows adding supplementary meta information to simplify text understanding.
4. **Qualitative and quantitative analyses based on different statistical characteristics** allow finding terms and lemmas frequencies per corpus and per document, mutual information value in collocation (allows finding proper collocation), strength of relations in collocations, keywords, the importance of the document in the corpus of texts, N-grams frequencies. They can annotate according to different levels of linguistic analysis of the corpus or text (phonetic, semantic, lexical, pragmatic, discourse or stylistic annotation) and find semantically close documents and semantically close terms.
5. **Comparison with different translations of the same text and map terms in different languages** means that there should be parallel multilingual corpus available that allows comparing characteristics of the same terms in several languages.
6. Texts that are stored and analyzed in the corpus should be chosen only by linguists to build proper dependencies and lead to proper statistics to prevent side effects in statistics.
7. Linguists can choose proper calculation methods of text preprocessing and analysis. They should be customizable.

The main objective of the given research is to develop supported tool for professional linguists and translators. This development comprises the right choice and justification of the proper mathematical apparatus (mathematical models, computational methods), proper algorithms and their effective implementations in software libraries and implement needed functionality.

4 Mathematical model of the corpus

To provide the needed level of qualitative and quantitative text analysis accuracy (concordance analysis, keywords and keywords clusters, collocations, relations between terms, relations between documents, relations between terms and documents etc.) input data should be ingested and preprocessed in the most appropriate way. It is recommended to use good techniques and choose correct input parameters in the methods which are used. Data ingestion is a multi-stage transformation of the input documents. It is worth describing the above mentioned processes by means of mathematical models.

Natural language can be treated as a set of terms or words: $\boldsymbol{T} = \{t_1, t_2, \ldots, t_n\}$, $n \in \boldsymbol{N}$.

This research describes multilingual corpus, thus set $\boldsymbol{L}$ is a superset over the $\boldsymbol{T}$ -sets that comprises terms from all languages: $\boldsymbol{L} = \{\boldsymbol{T_1}, \boldsymbol{T_2}, \ldots, \boldsymbol{T_n}\}$. As the next step, we consider the relations between terms and different elements of corpus for only one language. There is a subset $\boldsymbol{T'} = \{\boldsymbol{t'_1}, \boldsymbol{t'_2}, \ldots, \boldsymbol{t_n}\}$, $n \in \boldsymbol{N}$ of set $\boldsymbol{T}$ of terms that belong to book ingested into corpus. It is supposed that the whole corpus of one language is a set $\boldsymbol{A}$ that comprises books (set $\boldsymbol{B}$). Books are well structured and divided into chapters or documents (set $\boldsymbol{C}$), so each book is a set that contains a subset of chapters. Chapter contains sentences (set $\boldsymbol{S}$), sentence consists of natural language terms. Although the words in the sentence are arranged in a certain order, we consider elements of set $\boldsymbol{S}$ as pairs where first component is the term and second one — its position in the sentence $(t'_{ijkr};r)$. These relations can be described as follows:

$$\boldsymbol{A} = \{\boldsymbol{B_1}, \boldsymbol{B_2}, \ldots, \boldsymbol{B_n}\},\ \boldsymbol{B_i} = \{\boldsymbol{C_{i1}}, \boldsymbol{C_{i2}}, \ldots, \boldsymbol{C_{im}}\},\ i \in \overline{\boldsymbol{1},n},$$

$$\boldsymbol{C_{ij}} = \{\boldsymbol{S_{ij1}}, \boldsymbol{S_{ij2}}, \ldots, \boldsymbol{S_{ijp}}\},\ j \in \overline{\boldsymbol{1},m},$$

$$\boldsymbol{S_{ijk}} = \{(t'_{ijk\boldsymbol{1}};\boldsymbol{1}),(t'_{ijk\boldsymbol{2}};\boldsymbol{2}),\ldots,(t'_{ijkr};r)\}, k \in \overline{\boldsymbol{1},p}, \tag{5}$$

$$\boldsymbol{T'} = \{t_{\boldsymbol{1}}, t_{\boldsymbol{2}}, \ldots, t_h\},\quad h \in \overline{\boldsymbol{1},q},$$

$$n,\quad m,\quad q,\quad p,\quad r \in \boldsymbol{N},\ \boldsymbol{S} \subset \boldsymbol{C} \subset \boldsymbol{B} \subset \boldsymbol{A},\ \boldsymbol{T'} \subset \boldsymbol{A}$$

Note that each set has its own index which points to the position. This is very important for navigating through the terms in corpus. When a book is ingested into the corpus, sentence, term tokenization will be created, then, as a result, a subset of terms ($\boldsymbol{T'}$) will be obtained.

In (5) n — number of the books in the corpus, m — number of the chapters in any book of the corpus, p — number of the sentences in any chapter, r — number of the ordered terms in any sentence, q — number of unique terms in the corpus. Consider $z^{C_{ij}}$ — number of terms in j-chapter of i-book, and z^{B_i} — number of terms in i-book, $z_{t'_h}$ - number of h-term t'_h.

Based on the analysis of all ordered terms t' of pairs $\{(t'_{ijk\boldsymbol{1}};\boldsymbol{1}),(t'_{ijk\boldsymbol{2}};\boldsymbol{2}),\ldots,(t'_{ijkr};r)\}_{ijk}$ there can be obtained frequencies for each term in the chapter $v_{t'}^{(C_{ij})}$:

$$\left\{\left(v_{t'_{\boldsymbol{1}}}^{(C_{ij})}\right),\left(v_{t'_{\boldsymbol{2}}}^{(C_{ij})}\right),\ldots,\left(v_{t'_h}^{(C_{ij})}\right)\right\},\quad v_{t'_i}^{(C_{ij})} = \frac{z_{t'_h}}{z^{C_{ij}}}, \tag{6}$$

$$i \in \overline{\boldsymbol{1},n},\ j \in \overline{\boldsymbol{1},m},\ h \in \overline{\boldsymbol{1},q},\ n,\quad m,\quad q \quad \in \quad \boldsymbol{N}$$

And frequency for each term in the book $v_{t'}^{(B_i)}$ can be obtained as given below:

$$\left\{\left(v_{t'_1}^{(B_i)}\right),\left(v_{t'_2}^{(B_i)}\right),\ldots,\left(v_{t'_h}^{(B_i)}\right)\right\},\quad v_{t'_i}^{(B_i)}=\frac{z_{t'_h}}{z^{B_i}}, \tag{7}$$

$$i\in\overline{\mathbf{1},n},\ h\in\overline{\mathbf{1},q},\ n,\quad q\quad\in\quad\boldsymbol{N}$$

Keywords represent the most important lexical units of the text under analysis. There are different methods of keyword search described and suggested by V. Lytvyn, V. Vysotska, D. Uhryn, M. Hrendus, O. Naum, U.Shandruk, P.Pukach, O. Brodyak, especially for Slavic languages [29-31].

In this investigation Latent Semantic Analysis (LSA) [32] is applicable to find the most important concepts in different languages and is suggested as the most universal and performance effective method among other methods for the outlined tasks. It comprises such steps as finding importance of words in text corpus and building term-document or lemma-document matrix. Importance of the words in the documents can be obtained by calculation of frequencies or TF-IDF (term frequency–inverse document frequency) [32, 33]. TF-IDF is more accurate method in comparison to frequency calculation. Finding keywords comprises the removal of stop-words, finding lemmas, calculating TF-IDFs of each lemma and ranging lemmas by TF-IDF criteria.

Thus, to find keywords (set $\boldsymbol{T'}_{\boldsymbol{KW}}$) stop-words should be removed and lemmas for the rest of the terms are to be found. To remove stop-words it is advisable to use the dictionary (set $\boldsymbol{T'}_{SW}$) predefined by the linguist:

$$\boldsymbol{T'}_{KW}=\boldsymbol{T'}\setminus\boldsymbol{T'}_{SW},\ \boldsymbol{T'}_{KW}\subset\boldsymbol{T'},\ \boldsymbol{T'}_{SW}\subset\boldsymbol{T'}, \tag{8}$$

Lemmas are to be found among the terms belonging to the set $\boldsymbol{T}$ of the whole language and are to be compared with the keywords $\boldsymbol{T'}_{\boldsymbol{KW}}$ obtained according to language rules:

$$\boldsymbol{F}_{\boldsymbol{lemmatization}}:\boldsymbol{T'}_{KW}\rightarrow\boldsymbol{T''}_{KW}, \tag{9}$$

Having done lemmatization, the obtained set of lemmas $\boldsymbol{T''}_{KW}=\{\boldsymbol{t''}_1,\boldsymbol{t''}_2,\ldots,\boldsymbol{t''}_h\}$ analogous to (6) lemmas' frequencies are to be calculated. It is considered that $w^{\boldsymbol{B}_i}$ is number of chapters of *i*-book, and $w^{\boldsymbol{B}_i}{}_{t''_h}$ — number of some *h*-lemma s t''_h in *i*-book.

And now we can obtain importance of each lemma in any chapter by calculating TF-IDF $\xi_{t''}^{(C_{ij})}$:

$$\left\{\left(\xi_{t''_1}^{(C_{ij})}\right),\left(\xi_{t''_2}^{(C_{ij})}\right),\ldots,\left(\xi_{t''_h}^{(C_{ij})}\right)\right\},\quad \xi_{t''_i}^{(C_{ij})}=\frac{z_{t''_h}}{z^{C_{ij}}}\cdot\boldsymbol{log}\frac{\left|w^{B_i}\right|}{\left|w^{B_i}{}_{t''_h}\right|}, \tag{10}$$

$$i\in\overline{\mathbf{1},n},\ j\in\overline{\mathbf{1},m},\ h\in\overline{\mathbf{1},q},\ n,\quad m,\quad q\quad\in\quad\boldsymbol{N}$$

To find relations between terms (t'_h) which are converted to lemmas (t''_h) and documents cosine similarity metrics can be used [32]. This metrics is commonly used in natural language analysis domain. Term-document matrix $\boldsymbol{M}$ containing TF-IDF values is formed to find cosine similarity. Documents and terms are represented as vectors of TF-IDF values, i.e., each column represents document and each row represents term (11).

$$\boldsymbol{M} = \begin{array}{cc} \begin{array}{cccc} \boldsymbol{C_{i1}} & \boldsymbol{C_{i2}} & & \boldsymbol{C_{im}} \end{array} & \\ \begin{pmatrix} \xi_{t''_1}^{(C_{i1})} & \xi_{t''_1}^{(C_{i2})} & \cdots & \xi_{t''_1}^{(C_3)} \\ \xi_{t''_1}^{(C_{i1})} & \xi_{t''_1}^{(C_{i2})} & \cdots & \xi_{t''_1}^{(C_3)} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_{t''_h}^{(C_{i1})} & \xi_{t''_h}^{(C_{i2})} & \cdots & \xi_{t''_h}^{(C_3)} \end{pmatrix} & \begin{array}{c} t''_1 \\ t''_2 \\ \\ t''_{h1} \end{array} \end{array} \tag{11}$$

Cosine similarity between terms (12) and documents (13) is represented as follows:

$$\cos(\varphi) = \frac{t''_{h-1} \cdot t''_h}{t''_{h-1} \cdot t''_h}, \tag{12}$$

$$\cos(\varphi) = \frac{C_{im-1} \cdot C_{im}}{C_{im-1} \cdot C_{im}}, \tag{13}$$

Large matrix dimensions are to be seen as a disadvantage of this method. LSA method [33, 32] enables reducing both computational complexity and information noise by using singular value decomposition (SVD). SVD is based on using matrices $\boldsymbol{U}$, $\boldsymbol{S}$ and $\boldsymbol{V}^{\boldsymbol{T}}$:

$$\boldsymbol{M} \approx \boldsymbol{U} \cdot \boldsymbol{S} \cdot \boldsymbol{V}^{\boldsymbol{T}}, \tag{14}$$

Matrix $\boldsymbol{M}$ has dimensions $h \times m$, $\boldsymbol{U}$ is an $m \times l$ matrix, $\boldsymbol{S}$ is a $l \times l$ diagonal matrix, $\boldsymbol{V}^{\boldsymbol{T}}$ is a $l \times m$ matrix. SVD is parameterized with a parameter l, $l \le m$. By these parameters we set how many concepts are to be analyzed. There are two options:

1. If $l = m$, then we obtain original matrix exactly.
2. If $l < m$, then we obtain low-rank approximation of the original matrix. Usually l is chosen to be smaller than m. SVD ensures that the approximation will be the closest possible to the original matrix.

Hence, finding relations between terms and documents in LSA is based on finding cosine similarities between smaller matrices. In LSA all calculations are done with matrix $h \times l$ but not with matrix of original size $h \times m$.

Other very important features of the corpus are collocations and colligations. The process of obtaining collocations and colligations is based on finding ***N***-grams (a contiguous sequence of ***N*** terms) and computing probabilities of this *N*-grams in the corpus. The highest probabilities $P\left(\left(t'_{ijk\ \ v+\boldsymbol{1}};\ \ v+\boldsymbol{1}\right)\middle|\left(t'_{ijkv};\ \ v\right)\right)$ that the term $\left(t'_{ijk\ \ v+\boldsymbol{1}};\ \ v+\boldsymbol{1}\right)$

follows the term $(t'_{ijkv};\ v)$ point on the most used collocations and colligations in the natural language. There are usually 2-grams (bigrams) and 3-grams (trigrams) used in practice as follows:

$$\{(t'_{ijkv};\ v),(t'_{ijk\ v+1};\ v+1)\},$$
$$i\in\overline{1,n},\ j\in\overline{1,m},k\in\overline{1,p},\ v\geq 1,\ v+1\leq r, \tag{15}$$
$$n,\ m,\ p,\ v,\ r\ \in N$$

$$\{(t'_{ijkv};\ v),(t'_{ijk\ v+1};\ v+1),(t'_{ijk\ v+2};\ v+2)\},$$
$$i\in\overline{1,n},\ j\in\overline{1,m},k\in\overline{1,p},\ v\geq 1,\ v+2\leq r, \tag{16}$$
$$n,\ m,\ p,\ v,\ r\ \in N$$

Finding *N*-grams is one of the most time and memory consumable processes which are described by S. Ryza, U. Laserson, S. Owen, and J. Wills [32]. This task can be completed only after finding sentences and terms. The probabilities of collocation are calculated according to the formula given below:

$$P\left((t'_{ijk\ v+1};\ v+1)\,|\,(t'_{ijkv};\ v)\right)= \\ =\frac{Count\left((t'_{ijkv};\ v),(t'_{ijk\ v+1};\ v+1)\right)}{Count(t'_{ijkv};\ v)} \tag{17}$$
$$i\in\overline{1,n},\ j\in\overline{1,m},k\in\overline{1,p},\ v\geq 1,\ v+1\leq r,$$
$$n,\ m,\ p,\ v,\ r\ \in N$$

Corpus should contain much additional and useful information that describes its elements (part-of-speech tag, types of sentences etc.). This information will appear in the corpus after text data enrichment. This text data enrichment or augmentation process [34] comprises different types of tagging and text mark-up operations. Due to this research, data enrichment is achieved by using POS-taggers and sentence taggers. We obtain the set of tagged sentences $\tilde{S}_{ijk}$ and tagged ordered terms $(\tilde{t}'_{ijkr};r)$ in the sentences as follows:

$$F_{tagging}: S_{ijk}\to\tilde{S}_{ijk},\ \tilde{S}_{ijk}=\{(\tilde{t}'_{ijk1};1),(\tilde{t}'_{ijk2};2),\ldots,(\tilde{t}'_{ijkr};r)\}, \tag{18}$$
$$i\in\overline{1,n},\ j\in\overline{1,m},k\in\overline{1,p},$$

Result set of the corpus will be larger than the set of input data and will be based on the elements of additional dictionaries, different transformations and mappings of the source set elements.

5 Suggested implementation of the corpus tools

Building corpus of religious and historical texts is impossible without usage of Big Data techniques and methods which are usually used for building data lakes and data warehouses.

Figure 1 shows general preparation workflow and usage of the corpus from the point of view of Big Data engineering. After Extract-Transform-Load (ETL) phase data sources from different formats and structures become a set of structured text documents. Depending on the document structure and type of source data (image, text, xml etc.) this phase can be automated or semi-automated. For example, linguists should fulfill some manual tasks to divide structureless text with some special delimiters but well-structured documents can be ingested automatically.

Phase “preprocessing” is used to enrich [34] and prepare text for statistical analysis. This phase comprises text tokenization into sentences and terms, removing useless information, filtering stop words, lemmatization of terms, obtaining collocations and additional information about every term (different tags), numeric characteristics calculation of every term etc.

Preprocessing is implemented with the specific NLP and NLU libraries. Current implementation of the corpus tool is oriented on the English, Ukrainian and Russian texts’ processing. For the English language text processing it is suggested to use Stanford Core NLP 3.9.2 which is better NLP and NLU library. For the Ukrainian and Russian texts’ processing there is advisable to use LanguageTool 4.7. Feature extraction (N-grams, frequencies of terms, TF-IDF etc.) was done with Apache Spark MLLib, with usage of parallel and distributed capabilities of this library. LanguageTool and Stanford Core NLP was embedded into Apache Spark pipelines data processing. The effectiveness of Stanford Core NLP in the processing of the English texts is higher than that of processing Ukrainian and Russian texts by means of the LanguageTool because of different internal processing algorithms. Stanford Core NLP uses a well-trained neural network to do lemmatization, whereas LanguageTool uses vocabulary-based approach. Stanford Core NLP effectiveness is based on well trained models which can be observed only on its well supported languages: Arabic, Chinese, French, German, and Spanish.

Table 1 shows the results of average computation time of data processing under the same conditions run on Apache Spark 2.3.2 in local mode (JDK 1.8.0-231). Thus we can assume that text processing and text analyzing without usage of well-trained models will take the same average time as the Ukrainian and the Russian texts processing do. As it is shown on Table 1, there are given the cases of processing times for stop-words filtering, tokenization and lemmatization without N-gram extraction. N-gram processing time was not taken into account because their extraction was implemented with Apache Spark MLLib. These results show the effectiveness of NLP libraries.

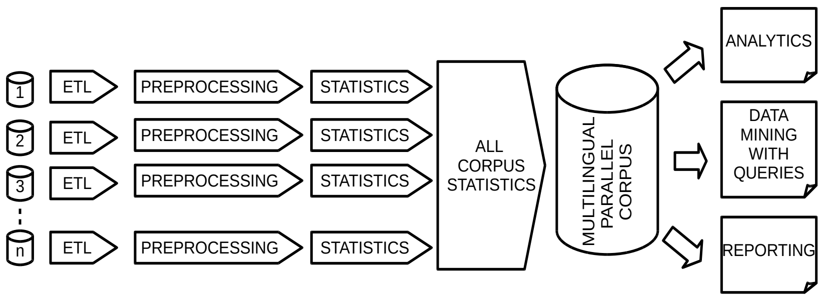

Fig. 1. Corpora tool data processing workflow

Table 1. Comparison of the average execution data processing time on different texts

Documents	Doc. Number	PC1 (intel i5-2300, 16GB)	PC2 (intel i7-8565U, 16GB)
English text1 [24]	1189	1M8.395S	44.148S
English text2 [25]	1189	1M1.407S	39.538S
English text3 [26]	1336	1M27.183S	53.605S
Ukrainian text1 [21]	1189	4M56.589S	3M3.604S
Ukrainian text2 [22]	1189	5M3.484S	3M8.59S
Russian text1 [27]	1189	10M54.735S	7M5.913S

There are two types of statistics, i.e., statistics per document and statistics per whole corpus. Statistics per document is calculated once when linguist adds a new document to the corpus. This type of statistics comprises term frequencies, N-grams (collocations) frequencies, results of LSA (term-document matrix, results of SVD decomposition).

Frequencies of terms per chapter in the book, frequencies per book, finding N-gram, calculating N-gram frequency and TF-IDFs were done with Apache Spark MLLib which provides specific libraries featured by high level of parallelization.

LSA is implemented with the libraries from Apache Spark MLLib which contains SVD. To obtain better performance of Spark MLLib in SVD-computation native operating system libraries ARPACK, BLAS and LAPACK were used. Among BLAS and LAPACK, which are well known basic linear algebra libraries, ARPACK is a Fortran77 parallel implementation of subroutines to solve large scale eigenvalue problems, especially eigenvalue decomposition. Usage of these native libraries is very important for speedup that can be achieved up to 50%-70% [35].

Statistics for corpus data should be recalculated when every new document appears in the corpus. All described phases are implemented as a pipelines in Apache Spark 2.3.2.

Thus, multilingual parallel corpus persists data of:

– source texts in unchanged immutable state;
– processed annotated and tagged texts;
– additional metadata which maps source and processed texts;

– statistics, analytics and reports on each document and the whole corpus.

These are different types of data which persist in the different types of storages. Conceptually it is the combination of data lake and data warehouse approaches, i.e., unstructured raw data stored with processed results and statistics for analysis. Distributed file system and relational database management system (RDBMS) are used to provide this functionality. In our case, the distributed file system is HDFS, but also it can be Amazon Web Services S3 object storage. Parquet file format, which is compressed and efficient columnar data representation for parallel and distributed data processing, is used for storing files on HDFS. PostgreSQL 9.6 is used as RDBMS. It also allows to do fast analytics on cached data and has a large amount of special text functions.

5.1 Data processing workflows

In our research corpus tools are treated as a type of data platform which comprises data processing software, storage system, software to make queries and which provides a user interface to interact with a platform.

Data extraction, transform, load and processing. Data processing software works according to the workflow which is illustrated by Figure 1 and is described above. The implementation of this workflow is based on the principle of data locality. This principle is provided by Apache Spark and HDFS architectures. Data extraction depends on the type of sources. Texts of the Bible were chosen for the demonstration in SQLite format. The extraction from UTF-8 text files, wikipedia-xml files and from PDF-documents, which are based on the usage of Apache Tika library, were also implemented. But in any case the linguist has to check the correctness of source documents. SQLite-files are extracted to PostgreSQL database which is supported by Apache Spark as data source for building Datasets. SQLite files can be extracted with Apache Spark only in local-mode execution. In cluster-mode these options are not appropriate. Apache Sqoop tool was also tested for data extraction from SQL to CSV files into HDFS. Advantage of this tool is the possibility to extract data from any SQL-data source into Parquet file on HDFS with required SQL-queries. In our case, it is decided to extract SQLite data into PostgreSQL to simplify ETL process and to read data into Apache Spark from PostgreSQL directly.

Internally, in Apache Spark it is used Spark SQL to process ingested datasets. To increase effectiveness of parallelization and avoid Spark Data Skew Problem [32] Spark broadcast function is used. Results of processing are stored on HDFS in Parquet-format and in PostgreSQL.

Storage system. As mentioned above when text is added to corpus, corpora tool makes all needed calculations and stores obtained results. There is statistics per document and statistics for all corpora, which should be recalculated after adding of every new document. Time of recalculation depends on the size of all corpus and complexity of query. Developed corpora tool is based on batch data processing principle. This principle means that new batch data processing which comprises recalculation of the whole statistics, starts after each new ETL of a new document finished.

Thus results of Apache Spark data processing and all data needed for calculations as a set of DataFrames are stored in the Parquet format on the HDFS and the data needed for fast user data queries are stored to PostgreSQL tables. Stored DataFrames are needed for the whole corpus statistics recalculation. Tables in RDBMS allows to build user interface to interact with the corpus.

Thus, combination of distributed file system and RDBMS provides reliable and fast data lake with capability to make fast queries and get analytics. At this stage of the corpus tool development analytics and reports are implemented with Apache Superset.

User interface. A user interface to interact with a platform and which provides ability to ingest data and to make queries was built as a separate microservice. This microservice is developed with Spring Boot 2.0 framework. Implementation of the microservice has been done conforming to methodology [33]. ETLs which are performed by Apache Spark are triggered through Livy service which provides REST-interface to Apache Spark and allows to submit tasks. Thus user uploads file for process and submits ETL task through microservice application. Queries can be done by interaction with this microservice which uses PostgreSQL as data source. At this paper main attention is pointed on building data workflow and text processing so user interfaces are mentioned only as a part of data platform. General architecture of the corpus data platform is shown on Figure 2.

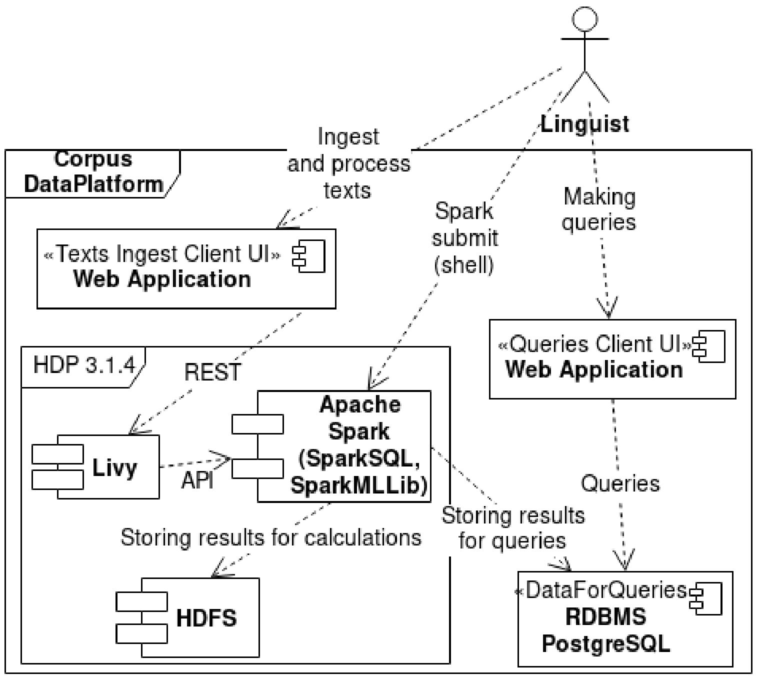
Linguist
Corpus DataPlatform
Ingest and process texts
Spark submit (shell)
Making queries
«Texts Ingest Client UI» Web Application
«Queries Client UI» Web Application
HDP 3.1.4
REST
Livy
API
Apache Spark (SparkSQL, SparkMLLib)
Queries
Storing results for calculations
Storing results for queries
HDFS
«DataForQueries» RDBMS PostgreSQL

Fig. 2. Corpora tool with implementation details

5.2 Corpus data platform implementation details

Corpora data platform is developed on the top of the Cloudera HDP 3.1.4 software stack. This stack provides:

- HDFS 3.1.1 for storing data;
- MapReduce2 3.1.1;
- Apache Spark 2.3.2 provides ETLs, data preprocessing, statistics and analytics (with a Livy server);
- Apache Sqoop 1.4.7 for ETL data in SQL format;
- Apache Yarn 3.1.1 cluster task scheduler;
- Apache Superset 0.23.0 for visualization and analytics.

There are also different supporting and cluster management tools (Ambari, Ranger, Zookeeper etc.). All this software stack is based on free and open source software and is to be installed on CentOS 7 Linux. Deployment was done into private cluster. Computational experiment was run in Apache Spark cluster mode.

Described implementation of the corpus tool is based on batching data processing, but to provide faster results is better to choose event-based lambda architecture [36] while adding new texts to the corpus. In lambda architecture delta rules are used for data processing models to achieve robustness, automation and efficiency. Any change in the state of data is an event to the system and as a matter of fact it is possible to give any command, queried or expected to carry out delta procedures as a response to the events on the fly.

All data processing workflows are implemented with Java 8 programming language and appropriate software libraries:

- Apache Spark 2.3.2 to process data in a parallel and distributed manner;
- LanguageTool for the Ukrainian and Russian languages tokenization, lemmatization, POS-tagging;
- Stanford Core NLP for the English language tokenization, lemmatization, POS-tagging;
- Statistics, feature extractions which were designed with distributed machine learning Apache Spark MLLib;
- Spring framework 5.1 for managing Java application context, i.e., manage beans' lifecycles, working with configuration, network communication (Spring WebMVC), working with database (Spring Data);
- Additional libraries for logging (slf4j, log4j2), wikipedia-xml extraction (cloud9, bliki), data mapping into xml and json (jackson), testing (junit, mockito) etc.

Microservice which provides user interfaces implemented with Spring Boot 2 framework, Spring Data, Spring WebMVC, Spring Cloud etc.

6 Computational experiment

Corpus of different editions, translations and languages of the Bible was compiled to verify the suggested approach. Few examples of investigated texts are shown in Table 2. Due to this investigation every Bible edition is treated as a subcorpus, i.e., a set of chapters. Each chapter has its own sentences and terms. After ETL the most important keywords, term POS-tags, relations between terms and other features for each chapter are obtained. After statistical processing each subcorpus (book) and its documents (book chapters) has its own characteristics.

Table 2. Input text data characteristics

No.	Book	Year of publishin g	Target translate d language	Number of books	Number of Chapters	Number of stories	Number of verses
1	[24]	2009	English	66	1189	1139	31102
2	[25]	2011	English	66	1189	1139	31102
3	[26]	2011	English	77	1336	1252	35488
4	[21]	2019	Ukrainian	66	1189	1175	31160
5	[22]	1962	Ukrainian	66	1189	-	31170
6	[27]	2014	Russian	66	1189	1011	31163

Corpus comprises a set of Bible editions (subcorpora) and allows to compare translations in different languages.

In this experiment it is very important that the context of the translation is taken into account. To provide literary translation many translators were oriented on different source languages, i.e., Ancient Hebrew, Ancient Greek and Old Slavic or intermediary languages. The translation of chapter titles does not match and differs. It is very often the fact that the source language of the translated text is or unknown or is very difficult to identify.

At the stage of ETL well-structured books of the Bible have been obtained. There were taken the books which were used for mobile applications in SQLLite format. Text data were imported into RDBMS PostgreSQL. Hence, data ingestion wasn't too complicated, since Apache Spark allows to use JDBC-connection as a source.

There are two important factors which are to be taken into account while processing books:

Some books are logically subdivided into stories, but the number of stories depends on translation and varies from zero to 1139. Some books contain less number of translated books. Due to these two factors and in order to provide more accurate results it is suggested to divide each book into documents by chapter criterion and to prepare custom ETLs for different types of books.

At this stage of ETL the obtained meta information about the book (author, publisher, source filename, edition, year of publication etc.) and tuples which comprise the title of the chapter and its text.

After ETL there are to be done the following processing steps:

- sentence and word tokenization (results are shown in Table 3);
- calculation of terms frequencies;
- collocation search (N-grams with high probabilities);
- POS-tagging and sentence tagging;
- stop words removal;
- lemmatization;
- calculation of TF-IDFs;
- building of term-document matrix with TF-IDFs;
- singular value decomposition with obtaining low-dimensional term-document matrix representation.

Table 3. Results of text tokenization

No.	**Book**	**Number of unique terms with stopwords**	**Number of unique lemmas (without stopwords)**	**Total number of terms**	**Total number of sentences**
1	[24]	14948	9897	1429388	81428
2	[25]	14716	9747	1452104	79832
3	[26]	13988	10088	1802496	67574
4	[21]	41743	19805	1078446	64962
5	[22]	42887	20000*	1125518	69448
6	[27]	41997	16239	1124732	80202

Fifth book published in 1962, so the set of terms comprises a large number of the old Ukrainian words and the corpus tool sets upper bound up to 20000 terms. This example shows the complexity of automatic processing of non-modern language.

While doing SVD it is important to choose proper value of *k* in finding relevancies between term and terms (Table 4), document and documents, term and documents as well as making search queries in corpus. But finding better numerical metrics of accuracy of concept search it has appeared to be a very complicated task due to many factors which must be taken into account. In this investigation *k*-value is taken as equal to 100. It is quite reasonable as a result if it is to compare with number of unique lemmas in every document and the size of processed input data (Table 3). Number of unique lemmas represents the dimension of lemma-document matrix. Thus the use of *k*=100 means that computational complexity is reduced approximately to 100 times. The use of *k*=100 resulted to be very close to *k*=1000. Hence, is reasonable to reduce dimensionality.

Table 4. Results of LSA calculation in finding related terms

Book	**Num. Documents**	**Results of finding top 10 related terms to term "light", "світло", "свет" (term and its weight)**
[24]	1189	[light, 0.99999], [darkness, 0.57538], [bearing, 0.487875], [expanse, 0.4852], [shine, 0.386439], [create, 0.36975], [vegetation, 0.36695], [formless, 0.35512], [wildlife, 0.345030], [lesser, 0.34467]
[25]	1189	[light 0.99999], [darkness 0.5586], [vault 0.464004] [teem 0.39937], [vegetation 0.374906], [formless 0.372928], [sky 0.36825], [winged 0.344664], [shine 0.32292], [hover 0.31357]
[26]	1336	[light 0.99999], [darkness 0.52926], [yielding 0.42156],[firmament 0.38736],[shine 0.350623], [see 0.32817],[earth 0.32739], [herb 0.308865],[winged 0.299326], [thing 0.294953]
[21]	1189	[світло 0.99999], [світлий 0.62616], [пітьма 0.53209],[бог 0.33081],[людина 0.322283], [день 0.313664],[шлях 0.304283], [земля 0.290772],[мова 0.28927],[бачити 0.28903]
[22]	1189	[світло 0.99999], [світлий 0.70187], [темрява 0.443529], [темрявий 0.406461], [день 0.315974], [бог 0.315765], [темнота 0.31495], [мати 0.31098], [земля 0.30418], [людина 0.30102]
[27]	1189	[свет 0.99999], [света 0.99284], [тьма 0.74948], [суд 0.58115], [ведомый 0.56315], [искать 0.56206], [миро 0.55932], [праведность 0.55761], [мир 0.55613], [слепнуть 0.54661]

Developed adaptable corpus allows choosing the custom *k*-value. Choosing better numerical metrics of similarity between results obtained due to the experiments with different *k* are of great importance and will be completed in later investigations.

Proper *k*-value also takes an important role when relevancies between term and terms, document and documents, term and documents are to be found. Search queries in corpus have to be found as well. Table 5 presents the top 10 relevant terms fixed as searchable terms.

Table 5. Results of LSA calculation with different *k*-value

Book	Num. Documents	k	Top 10 terms for the first concept in a descending order
[24]	1189	100	offering, king, david, son, male, priest, clan, bull, israel, saul
		1000	weaned, resounding, forevermore, plowmen, singers, sustainer, detailed, dampen, pillow, bowstr
[25]	1189	100	weaned, surpassing, clash, plowmen, preside, slur, preeminent, enact, beheld, compacted
		1000	weaned, surpassing, clash, plowmen, preside, slur, preeminent, enact, beheld, compacted
[26]	1336	100	mesech, plower, mower, sheave, backbiteth, shade, anointest, comfortedst, relieveth, vilest
		1000	mesech, mower, sheave, plower, backbiteth, shade, anointest, comfortedst, relieveth, vilest
[21]	1189	100	вийти, піти, чоловік, нарід, побачити, місто, пос-лати, батько, час, стати
		1000	вийти, піти, чоловік, нарід, побачити, місто, пос-лати, батько, час, стати
[22]	1189	100	трубний, дивніший, пишнитися, заспокоювати, подякування, повіквічне, однокупно, хермонська, оберемок, порвати
		1000	трубний, дивніший, заспокоювати, пишнитися, подякування, хермонська, повіквічне, однокупно, порвати, оберемок
[27]	1189	100	послать, оставить, посол, больший, тома, царь, иерусалим, слуга, отдать, услышать
		1000	послать, оставить, посол, больший, тома, царь, иерусалим, слуга, отдать, услышать

The accuracy of results shown in table 4 also depends on value *k*. Computational system which has been developed for carrying out these experiments is based on HDP 3.1. Computational cluster is deployed in private cloud with manage nodes (master nodes) in OpenStack environment [37] and computational nodes are bare-metal nodes. The suggested solution can be also deployed in the public clouds such as Amazon EMR with distributed storage on Amazon S3[38].

7 Conclusions and future work

Developed corpus data platform is capable to satisfy specific linguist needs and combines multiple functions of parallel, multilingual and diachronic corpora. Adaptability of the corpus is provided by the ability to set custom parameters for every text document and the capability to make quantitative and qualitative analyses. It allows linguists to compare multilingual translations of the same texts. Suggested approaches and methods of corpus data platform development have been verified by implementation support of the Ukrainian, Russian and English languages and based on the example of the Bible text analysis. In comparison to analog corpora this tool has the same basic functionality, but its adaptability gives new opportunities for text analysis by customization usage of specific data mining methods and custom dictionaries.

It has been also suggested to represent elements of the corpus as a number of related sets, i.e., terms, sentences, chapters, books etc. These elements are represented in mathematical model of the corpus.

There have been compared and analyzed different NLP and NLU libraries which allow scientists to provide investigations with needed basic functionality and could be extended by adding new language rules, especially those of the ancient languages.

References

1. Kibler, S., Zinsmeister, H.: Corpus Linguistics and Linguistically Annotated Corpora Bloomsbury Academic, New York, London, 21-156. (2015) http://dx.doi.org/10.5040/9781472593573
2. Lee, D.Y.W.: What corpora are available? In: McCarthy M. and O'Keeffe A. (eds) The Routledge Handbook of Corpus Linguistics. Routledge, Abingdon,107-121. (2010)
3. Weisser, M.: Manual for the Dialogue Annotation & Research Tool (DART). Version 3.0. http://martinweisser.org/publications/DART_manual_v3.0.pdf. Accessed 15 Dec 2019. (2019)
4. Lutskiv, A., Popovych, N.: Adaptable Text Corpus Development for Specific Linguistic Research. In: Proceedings of IEEE International Scientific and Practical Conference Problems of Infocommunications. Science and Technology, Kyiv, 217-223. (2019)
5. Kruger, A., Wallmach, K., Munday, J. (eds): Corpus-Based Translation Studies. Research and Applications.Continuum, London. (2011)

6. Anthony, L.: A critical look at software tools in corpus linguistics. Linguistic Research 30(2):141-161. (2013). https://doi.org/10.17250/khisli.30.2.201308.001
7. Anthony, L. (February 18, 2019) AntConc (Windows, Macintosh OS X, and Linux) Build 3.5.8. http://www.laurenceanthony.net/software.html. Accessed 14 Feb 2020
8. Scott, M.: WordSmith Tools Manual. (2019). https://lexically.net/downloads/version7/HTML/index.html. Accessed 15 Dec 2019
9. Brezina, V., Timperley, M., McEnery, T.: #LancsBox v. 4.x [software]. (2018) http://corpora.lancs.ac.uk/lancsbox. Accessed 15 Dec 2019.
10. Garretson, G.; Dexter. Tool for analyzing language data. (2011). [Online]. Available: http://www.dextercoder.org/index.html. Accessed 15 Feb 2020.
11. Francis, W.N., Kucera, H.: Standard American English or the Brown Corpus. (1979). http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM#t1. Accessed 15 Feb 2020
12. Klymenko, N.: Structural Linguistics. Linguistic Synergetics. Cumputational Linguistics. Selected Works. Dmytro Burago Publishing House. Kyiv, 445-567. (2014)
13. Dartchuk, N.: Corpus of the Ukrainian Language (Ukrainian). (2002-2020). http://www.mova.info/corpus.aspx?l1=209. Accessed 15 Feb 2020
14. Laboratory of Ukrainian (Ukrainian). https://mova.institute/. Accessed 15 Feb 2020
15. General Regionally Annotated Corpus of Ukrainian (GRAC) (Ukrainian). http://uacorpus.org/, http://www.parasolcorpus.org/bonito/run.cgi/first_form. Accessed 15 Feb 2020
16. Godfrey, J.J.: (1997) Air Traffic Control Complete. Linguistic Data Consortium. https://catalog.ldc.upenn.edu/LDC94S14A Accessed 20 Feb 2020.
17. Bennett, Ch., Rudnicky, A. I.: The Carnegie Mellon Communicator Corpus. In: Proceedings of the International Conference of Spoken Language Processing. Denver, Colorado, 341-344. (2002).
18. Opus, the Open Parallel Corpus. http://opus.nlpl.eu/.Accessed 05 Feb 2020.
19. Nuopponen, A.: Begreppssystem für Terminologisk Analysis (Concept Systems for Terminological Analysis). Acta Wasaensia 38. 266, (1994).
20. Kulish, P. O.: Nechuy-Levȳts'kyj IS, Pulyuj I (transll), Svyate Pȳs'mo Staroho i Novoho Zavitu. Movoyu rus'ko-ukrayins'koyu (2010) Prostir, Kyiv. 852+249.
21. Novitniĭ pereklad Bibliyi Oleksandra Hȳzhi: Druk KT Zabelina-Fil'kovs'ka, Kyiv. 1210, (2013).
22. Ohiyenko, I.: (transl) Bibliya Abo Knȳhȳ Svyatoho Pȳs'ma Staroho ĭ Novoho Zapovitu. Iz movȳ davn'oyevreĭs'koyi ĭ hrets'koyi na ukrayins'ku doslivno nanovo perekladena. UBT. Kyiv.1529, (1962).
23. Turkonyak Roman, Iyeromonakh o. Rafayil (transl) Bibliya (chetvertȳĭ povnȳĭ pereklad z davn'-ohrets'koyi movȳ): Ukrayins'ke Bibliĭne Tovarȳstvo. L'viv. ukrbible.at.ua/load/zavantazhiti_ukrajinsku_bibliju/skachaty.../7-1-0-165. Accessed: 10 Feb 2020
24. Christian Standard Bible, Holman Bible Publishers, Nashville, 1920, (2011).
25. Holy Bible: King James Version, Study Edition, Containing The Old Testament, Apocrypha, and New Testament. American Bible Society.1462, (2011).
26. Holy Bible. New International Version (2011). Zondervan Publishing House.1140.
27. Byblyja Novj russkyj sovremennj perevod Slovo Zhyzny. Mezhdunarodnoe Byblejskoe Obshhestvo Biblica, 998, (2014).
28. Rayson, P., Archer, D., Piao, S., McEnery, T.: The UCREL Semantic Analysis System. In: Proceedings of Beyond Named Entity Recognition. Semantic Labelling for NLP Tasks. Workshop, LISBON, Portugal, 7-12. (2004)

29. Lytvyn, V., Vysotska, V., Uhryn, D., Hrendus, M., Naum, O. Analysis of statistical methods for stable combinations determination of keywords identification. Eastern-European Journal of Enterprise Technologies. 2/2(92): 23-37, (2018).
30. Shandruk, U.: Quantitative Characteristics of Key Words in Texts of Scientific Genre (on the Material of the Ukrainian Scientific Journal). In: Proceedings of CEUR Workshop, Vol. 2362, 163-172, (2019).
31. Lytvyn, V., Vysotska, V., Pukach, P., Brodyak, O., Ugryn, D.: Development of a method for determining the keywords in the slavic language texts based on the technology of web mining. EasternEuropean Journal of Enterprise Technologies, 2(2-86):4-12, (2017).
32. Ryza, S., Laserson, U., Owen, S., Wills, J.: Advanced Analytics with Spark. Patterns for Learning from Data at Scale, 2nd ed. O'Reilly MediaInc. Sebastopol, 115-136, (2017).
33. Jurafsky, D., Martin, J.H.: Speech and Language Processing, 2nd ed. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. (2009).
34. Data Quality Management In: Multi-Domain Master Data Management. Advanced MDM and Data Governance in Practice, 131-160, (2015). http://www.odbms.org/wp-content/uploads/2015/09/Multi-Domain-Master-Data-Management_Ch9.pdf. Accessed 28 Feb 2020. https://doi.org/10.1016/B978-0-12-800835-5.00009-9
35. Zuling, Kang: Using Native Math Libraries to Accelerate Spark Machine Learning Applications. Cloudera Data Science (2019). https://docs.cloudera.com/documentation/guru-howto/data_science/topics/ght_native_math_libs_to_accelerate_spark_ml.html. Accessed 28 Feb 2020.
36. Kiran, M., Murphy, P., Monga. I., Dugan, J., Baveja, S.S.: Lambda architecture for cost-effective batch and speed big data processing (2015). In: Proceedings of IEEE International Conference on Big Data (Big Data). Santa Clara, CA, USA. 2785-2792. DOI:10.1109/BigData.2015.7364082
37. Lutskiv, A.: Provisioning Hortonworks Data Platform onto OpenStack with Terraform (2018) https://dataengi.com/2018/09/21/terraform-hdp/. Accessed 28 Feb 2020.
38. Amazon EMR (2020) https://aws.amazon.com/emr/. Accessed 28 Feb 2020.
39. Mykhalyk, D.: Spark DataSkew Problem. (2019) https://dataengi.com/2019/02/06/spark-data-skew-problem/. Accessed 28 Feb 2020.
40. The twelve-factor app (2017). https://12factor.net/. Accessed 28 Feb 2020.