Open Stylometric System WebSty: Towards Multilingual and Multipurpose Workbench

Maciej Piasecki
Faculty of Computer Science and Management
Wrocław University of Science and Technology
maciej.piasecki@pwr.edu.pl

Tomasz Walkowiak
Faculty of Electronics
Wrocław University of Science and Technology
tomasz.walkowiak@pwr.edu.pl

Maciej Eder
Institute of Polish Language
Polish Academy of Sciences and Pedagogical University of Kraków
maciej.eder@ijp.pan.pl

Abstract

WebSty is an open, web-based stylometric system designed for Social Sciences & Humanities (SS&H) users. It was designed according to the CLARIN philosophy: no need for installation, minimised requirements on the users' technical skills and knowledge, and focus on SS&H tasks. In the paper, we present its latest extension with several visualisation methods, techniques for the extraction of characteristic features, and support for multilinguality.

1. Introduction

Stylometry is based on the analysis of language features extracted from texts and aimed at tracing similarities between texts. It is used to identify groups of texts that exhibit subtle similarities hidden to the naked eye but traceable by multidimensional statistical techniques. A classical type of such an analysis is authorship attribution or an experimental setup in which anonymous (or disputed) texts are compared against a set of texts of known authorship, to identify the nearest neighbour relations (Stamatatos 2009). In Social Sciences & Humanities (SS&H) text analysis is becoming an interesting methodological proposition to assess textual similarities beyond authorship. In the study of literature, one might be interested in distant reading techniques to pinpoint genre characteristics, literary period, intertextuality, etc. In sociology, one might want to analyse textual biases in press materials from different press agencies, in psychology one might trace a change of the style as a function of the authors' age or correlations between a text and mental diseases (Le et al. 2011).

An application of the stylometric methods can be difficult for SS&H researchers, mostly because the combination of the variety of data formats, language tools, and data analysis tools is not straightforward, but also an application of the tools usually requires specialised knowledge and technical skills. Moreover, the entire NLP workflow is controlled by a large number of hyperparameters whose influence on the overall results of the stylometric analysis is complex.

WebSty[1] is an open stylometric system with the web-based user interface designed to be used without any installation, and which offers a variety of dedicated language processing tools, provides ready to use processing chains, and assists users in setting up the processing parameters. It was initially focused on processing texts in Polish[2] and offered a limited number of the visualisation and data analysis methods (Eder et al. 2017). Below we present a new version which has been expanded with a more flexible and efficient processing architecture, several visualisation methods and techniques for the extraction of characteristic features. The modular architecture of WebSty enabled adding support for more languages, namely English, German, Russian, Hungarian and Spanish in a relatively easy way.

Stylometric techniques are based on converting text documents or fragments into vectors of numerical values and next on processing the resulting vectors by data analysis method. The goal is to find similarities in the input data. This is often achieved by applying clustering algorithms that divide the vectors into similarity classes, e.g., documents of the same author.

[1] http://websty.clarin-pl.eu

[2] http://ws.clarin-pl.eu/websty.shtml

The paper is structured as follows. First, related solutions are analysed. Next, the architecture of the language processing is presented. It is followed by a detail description of the data analysis and visualization techniques.

2. Related Work

In spite of the long tradition of stylometry, there is only a limited number of online systems. The well known *Voyant*[3] (Sinclair & Rockwell, 2012) is an online tool for the limited statistical analysis of texts supplemented with a good GUI and several visualisation methods. A range of NLP tools was added on the a basis of the Stanford CoreNLP (Manning et al., 2014), e.g., PN recognition. However, the functionality of Voyant is based mostly on tracing word forms and their relative frequencies across text and limited to English. Only simple statistical measures: tf.idf and Z-score are available to compare word forms vs. documents. Popular *Stylo* (Eder et al. 2016) is a library in the R programming language for different stylometric tasks. It is designed to analyse shallow morphological features (function words and letter n-grams) harvested from the locally stored plain text files, but it can also be used to analyse preprocessed corpora. The package offers both selected exploratory methods, and supervised Machine Learning (ML) algorithms. It needs to be locally installed. *Mallet* (McCallum, 2002) is an off-line document classification system working on the basis of machine learning, but it is mostly used for the topic analysis.
Also, we can find on the Web a couple of simple online applications for the authorship attribution[4], e.g. *Signature* (only word-level features) and *AICBT* (limited number of feature types for English). There is a number of off-line applications, like *JGAAP* (an entire processing workflow), *JStylo* (McDonald et al., 2012) (rich set of feature types, recognition of obfuscation), and *StyleTool* (Maurer, 2017) (quite rudimentary). Neither of the discussed systems supports parallel processing of large amounts of data, nor they use multiple language tools and processing methods, and an advanced extraction of characteristic features.

3. Language Processing Architecture

A multi-user, web-based system generates problems related to the system availability and performance. The system should be *scalable*, *responsive* and *available* all the time. Language tools (LTs) have excessive CPU/memory consumption. Needless to say, the number of users and/or tasks at a given time is fairly unpredictable, which makes the allocation of resources even more problematic. WebSty architecture is presented in Fig. 1. It is based on a *service-oriented software* idea (Bell, 2010), that has gained great popularity, according to which each LT is implemented as a *microservice* (Wolff, 2016) and run as a separate process with the pre-loaded data models. The number of microservices run in parallel is limited by hardware. Each type of LT has its own queue. NLP microservice collects tasks from a given queue and sends back messages when the results are available.
The usage of microservices communicating via lightweight mechanisms solves the problems of a variety of programming languages used, and complexity of the tight integration. As the number of microservices run in parallel is limited by hardware, the queening system is crucial for the system performance and effective scalability. The most required and most frequently used LT microservices have to be run in several instances, and the queuing system acts as a load balancer. AMQP[5] protocol for the lightweight communication mechanisms and RabbitMQ broker are applied. An additional server grants access from the Internet and works as a proxy for the core system delivering REST API to WebSty. This allows for easy integration with almost any kind of applications. The exchange of data between microservices, i.e., input/output of the LT tools is done via a network file system. It makes the integration of new LT tools easier since they are mostly designed in a manner in which they expect a file at their input and produce files on their output.

[3] Voyant: http://docs.voyant-tools.org, CoreNLP: https://nlp.stanford.edu/software/, Mallet: http://mallet.cs.umass.edu
[4] Signature: http://www.philocomp.net/humanities/signature.htm, JGAAP: https://github.com/evllabs/JGAAP, JStylo: https://psal.cs.drexel.edu/index.php/JStylo-Anonymouth, StyleTool: https://github.com/lnmaurer/StyleTool
[5] AMQP: https://www.amqp.org, RabbitMQ: https://www.rabbitmq.com

The most common approach to the communication with web applications, namely REST API (REST), is synchronous. A thread producing the request is blocked until the response has not been returned to the client. In the case of requests that can be served in short response time, it is a very useful solution. However, when the response time is increasing, this can cause errors. First of all, the number of threads on the server side is limited, so the increasing response time could result in approaching this limit. Secondly, a response longer than the HTTP client timeout (usually equal to 189 s) causes the timeout limit error on the client side and breaks the connection. Thus, the receiving of the results fails (Walkowiak, 2014). That is why we use asynchronous way of communication in a polling-like way in WebSty. The client (written in JavaScrip) keeps checking, whether the server has already finished the processing. To keep the user informed about the processing status, HTTP API provides the information concerning the advancement of processing in percentages. The other problem with REST API can be caused by a large input data volume. To prevent this, we implemented different methods for uploading corpora. WebSty allows for loading text files one by one, in one ZIP file (for texts a ZIP file is usually ten times smaller than original documents). In the case of really very large corpora, it is possible to download them first into a CLARIN-PL Cloud storage[6] and next to inform WebSty (via REST API) about the corpus name in the cloud.

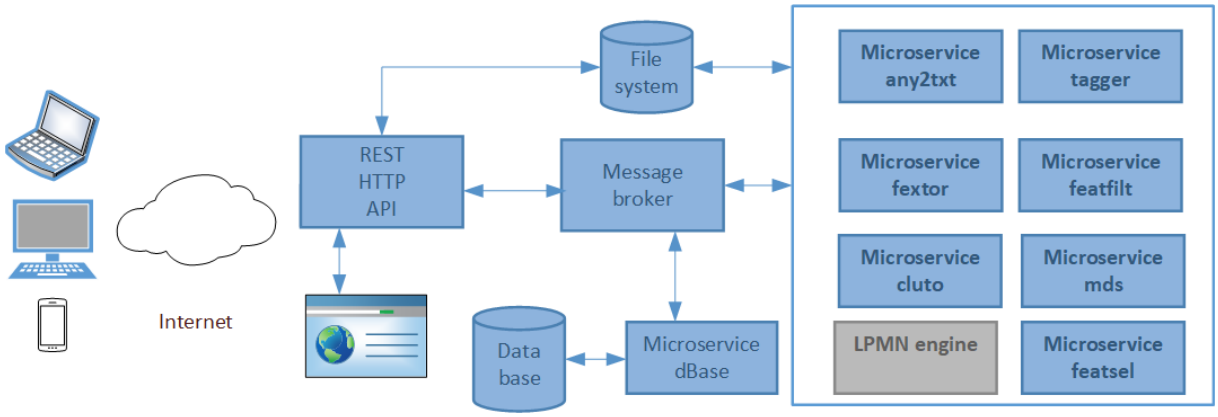

Figure 1. Language processing architecture

To achieve high availability, the system was deployed on a scalable hardware and software architecture that forms a private cloud (ten Blade Servers, connected by the fast fiber channel with highly scalable midrange virtual storage designed to consolidate workloads into a single system for the simplicity of management). XENServer controls each machine and forms a private cloud. Each frequently used NLP microservice is deployed on a separate virtual machine (Walkowiak, 2016).

WebSty requires to run NLP and ML tools is a specific order. Very often this is not a simple chain of tools, but a workflow of tools (Walkowiak, 2017). Therefore, we developed (Walkowiak, 2018) a human-readable orchestration (Peltz, 2003) language, that allows for the description of the WebSty processing tasks. It is called *Language Processing Modelling Notation* (LPMN). An exemplar of the LPMN statement for WebSty is presented in Fig. 2. Each microservice is defined by its name (for example: *any2txt*, *tagger*, *fextor* in Fig. 2). Data can be loaded from different sources, which are defined by dedicated LPMN statements. For example *urlzip* in Fig. 2 definies the URL of a ZIP file with input corpus. A dedicated microservice (LPMN engine in Fig. 1) is used to process LPMN tasks and acts as an orchestrator for other microservices.

The LPMN engine processes tasks of different sizes and computational complexity. Therefore, it is needed to prevent every tasks from blocking the small ones. To achieve this, we added a dedicated scheduling algorithm that prevents large files and large corpora (including a large number of files) from blocking the NLP microservice queues in the message broker. The engine checks the queue size, and if it has exceeded a predefined threshold (different for large files and for large corpora), the processing (sending tasks to the queue) is delayed for a given amount of time. As a result, simple tasks (for example: processing of one, small text file) are processed by LPMN engine in the time shorter than 6 s,

[6] http://nextcloud.clarin-pl.eu

even if the LTC is busy with processing very large corpora. The experiments, showed that the delay caused by the scheduling algorithm is smaller than 1% of overall processing time.
The LPMN engine registers tasks in the database (see Fig. 1). The recorded information allows for collecting statistics about the WebSty usage (see also the conclusions).

```
urlzip("http://ws.clarin-
pl.eu/public/teksty/2mini.zip")|any2txt|div(20000)|tagger({"lang":"polish"}
)|fextor2({"features":"base                interp_signs                bigrams",
"base_modification":"startlist","orth_modification":"startlist","lang":
"ud","filters":{"base":[{"type":"lemma_stoplist","args":{"stoplist":"@resou
rces/fextor/ml/polish_base_startlist.txt"}}]}})|dir|out("output_fextor")|
featfilt({"similarity":"cosine","weighting":"all:tf","filter":"min_tf-1
min_df-1"})|cluto({"no_clusters":2,"analysis_type":"plottree"})
```

Figure 2. Exemplar LPMN for WebSty analysis

4. Data Analysis

4.1 Basic options

WebSty development follows the CLARIN recommendations to make language tools available on Internet and develop research web based applications (Wittenburg et al., 2010), as a way to the elimination of the problems caused by the necessity of installing language tools (LTs) and possessing the required computational power. WebSty allows users to process data online without a need to bother about technicalities. However, some level of understanding the processing mechanisms is required to fully operate the application on the level of its user interface.
The user interface has been developed in HTML5 and JavaScript technology, using REST web service to run and control the language processing workflow on the server side. Firstly, the user must select the language of the text to be analysed (Fig. 3). Next, the number of groups into which the input corpora will be divided by the assumed clustering algorithm (see Sec. 4.6). Moreover, the input texts can be automatically divided into smaller parts of the approximately equal size set up by the user.
The key issues in the stylometric analysis are: definitions of features for the description of texts and methods for their further processing. WebSty offers a large set of features and weighting methods (presented in Sec. 4.3 and 4.5). Therefore, the four predefined sets of features and weightings are available for users, namely for the analysis of: Authorship, Grammatical style, Content and Classical Authorship (the last one is based on the most frequent words only). By selecting each of them the appropriate set of features and weighting methods is automatically set up for processing.

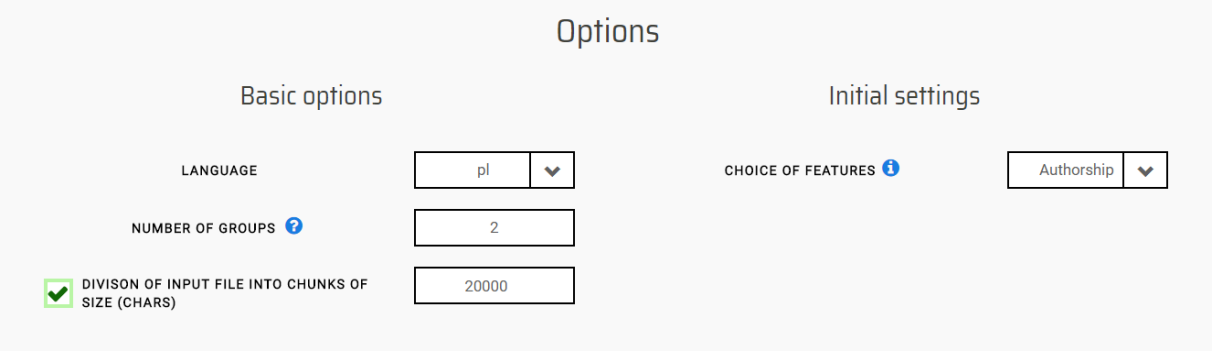

Figure 3. Basic WebSty options

4.2 Input data

Documents can be uploaded in many formats, e.g. MS Word, PDF, plain texts etc. The format of each document is automatically detected and the text content extracted. For larger data sets, a connection

between WebSty and the D-Space-based[7] public repository of CLARIN-PL was built: data sets deposited in the repository can be selected for processing in WebSty. However, due to the users' demands, it is also possible to upload documents from a ZIP file identified by its URL or ZIP files that are locally stored on the user's disk (the last option is limited concerning both the data volume and the number of files). In the case of very large corpora or private corpora (that are less convenient to be stored in the repository), users can also use the CLARIN Cloud – a NextCloud-based[8] private storage provided by CLARIN-PL. To use it, the user has to first log into the CLARIN-PL single authorization system (Pol, et al. 2018), and next he can select files from the storage (Fig. 5). It is worth to emphasis that logging into CLARIN-PL is necessary only for accessing the CLARIN Cloud and some restricted resources in the repository, WebSty as a system is completely open.

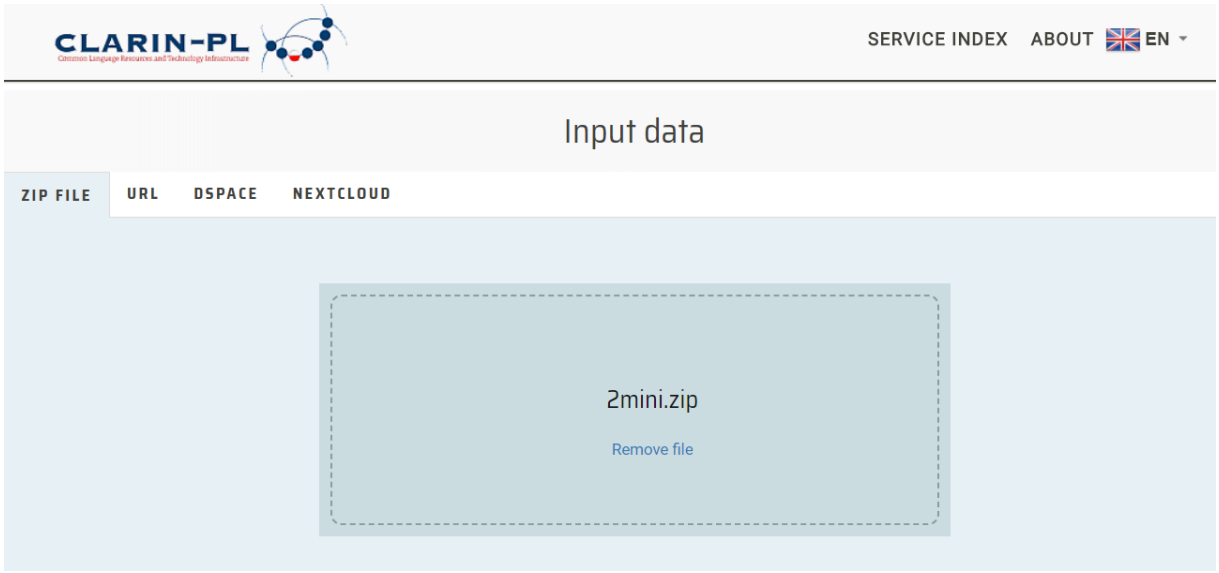

Figure 4. ZIP file input data interface

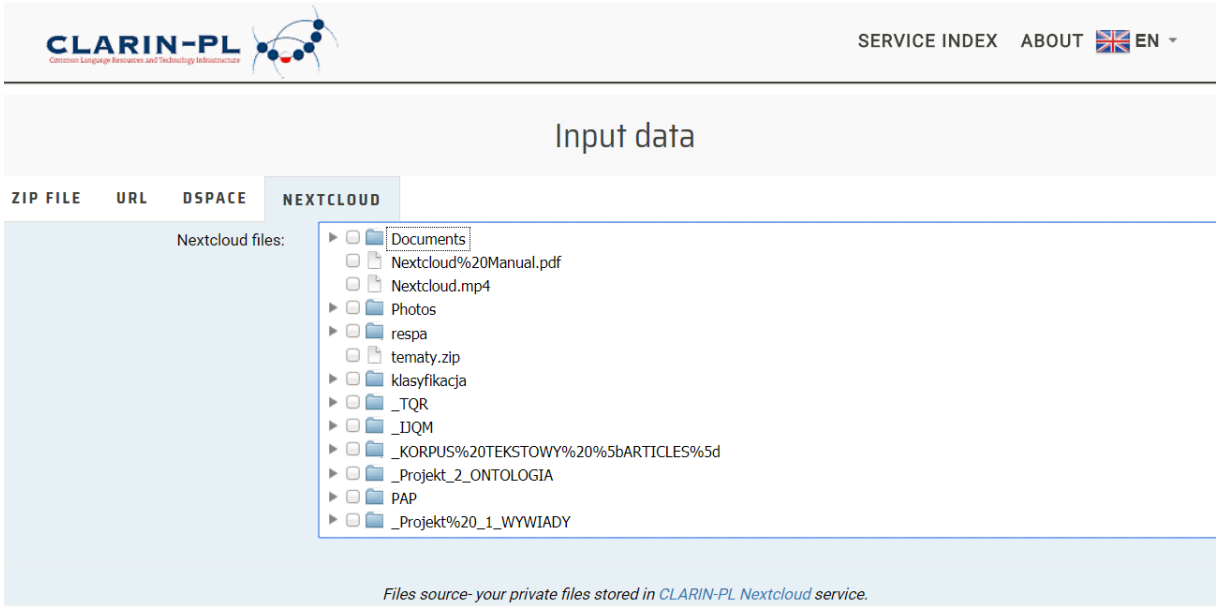

Figure 5. Selecting input data from CLARIN Cloud research storage

[7] https://clarin-pl.eu/dspace/
[8] https://nextcloud.clarin-pl.eu

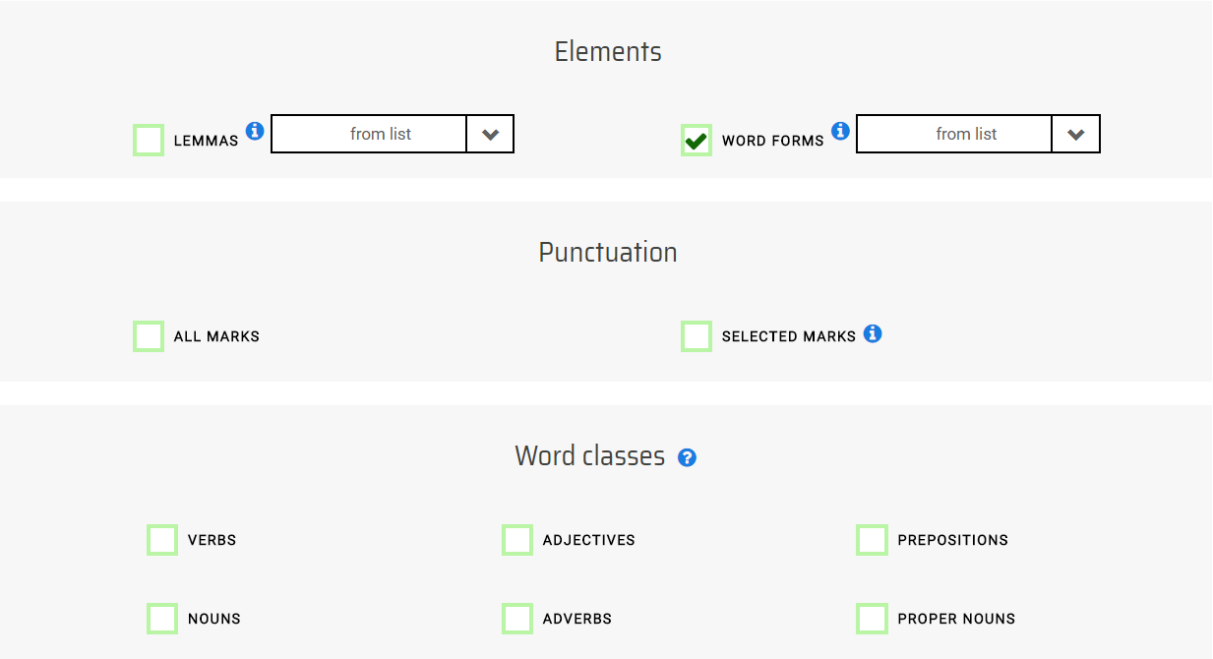

Figure 6. Feature selection panel

4.3 Document features

Text documents or their fragments (if the automatic division option has been switched on, see Fig. 3) are first converted into *feature* vectors of numerical values, which are next filtered, transformed and finally processed by the various data analysis methods. The ultimate goal is to group the vectors into similarity classes by the clustering algorithms (i.e. an unsupervised approach).
The features are defined on the basis of the characteristic elements of the text linguistic structure and also of the document. The initial values of the features are frequencies of these elements. Next processing techniques are used to clean the vectors from noise and finally to compare them.
The **features** should reveal properties of a text that are characteristic for its author and his style, and they should not be correlated with its semantic content. A feature can refer to any level of the language analysis but should be based only on LTs that express a relatively small error. WebSty (Fig. 6), implements features based on the frequency of: word forms (words in text), punctuation, lemmas (basic morphological forms), grammatical classes (a rich tagset), Parts of Speech (as sets of grammatical classes), grammatical categories, bigrams and trigrams of grammatical classes, and semantic types of proper names. The lemmas and grammatical classes are obtained by the application of a morpho-syntactic tagger, occurrences and types of proper names come from a Named Entity Recogniser.

4.4 Multilinguality

WebSty was developed for Polish (Eder et al. 2017), and, initially, all grammatical features were based on the Polish National Corpus tagset[9] (Przepiórkowski et al., 2012). Next, it has been expanded with support for English[10]. For this, we used spaCy[11] package for PoS tagging and lemmatization (Honnibal, Johnson, 2015). This process was continued by converting WebSty architecture into a **multilingual system**. It was quite simple, because even in the original monolingual version of the system the phases of text preprocessing and feature extraction were separated and performed by the two different modules. In the multilingual version, a third module was introduced in between, as it has been assumed that the annotated text delivered on the input of the feature extraction module has the same annotation format

[9] http://nkjp.pl/poliqarp/help/ense2.html
[10] http://ws.clarin-pl.eu/webstyen.shtml?en
[11] https://spacy.io

independently from the language, see below. So the task of the third added module – a transformation module – is to convert the automatically annotated text obtained from the preprocessing into the format expected by the extraction module.
As a result, the contemporary version of WebSty is capable of analysing texts in Polish, English, German, Russian, Hungarian and Spanish. The extension depends on the existing taggers and Named Entity Recognizers for the supported language. Due to the different tagsets used by the taggers, we selected Universal Tagset[12] (Petrov et al., 2012) as the input format for the feature extraction. The transformation module converts the original tagest of a tagger (a native format) into Universal Tagset. As the size of the latter is limited, it expresses quite coarse-grained classification (especially in comparison to the PNC format), so the conversion is a lossy process which introduces some generalisation[13].
As a default tagger we use UDPipe[14] (Straka & Straková, 2017) which has models trained for a large set of languages. In fact, we initially hoped to use UDPipe as the only and universal tool for preprocessing. However, the accuracy of tagging and lemmatization is significantly smaller than the one expressed by solutions dedicated to the individual languages. That is why, finally, whenever possible we use taggers dedicated for different languages and of better accuracy. For instance, for Polish WCRFT tagger (Radziszewski, 2013) with a converter from the NCP to Universal Tagset was selected and PurePos[15] tagger for Hungarian (Orosz & Novák, 2013).

4.5 Filtering, weighting and similarity

The features which are suspected to introduce too much noise or not be relevant to the goal of the analysis, can be **filtered** out on the basis of: their raw value (e.g., minimal number of documents), weighted value (after preprocessing) or their type (e.g., specified lemmas, grammatical classes, bigrams, etc.).
Raw frequencies are often skewed, e.g., by the document length, document content, or by the general properties of a given very frequent lemma. WebSty (Fig. 7) offers several **weighting methods**: *tf* (normalised text frequency), *tf.idf*, vector normalisation, PMI (Pointwise Mutual Information) simple and discounted, and *tscore*. As the number of features can be very high, a few dimensionality reduction techniques were included: *SVD* (Singular Value Decomposition), *LSA* (Latent Semantic Analysis) (Landauer & Dumais, 1997) and Random projection.

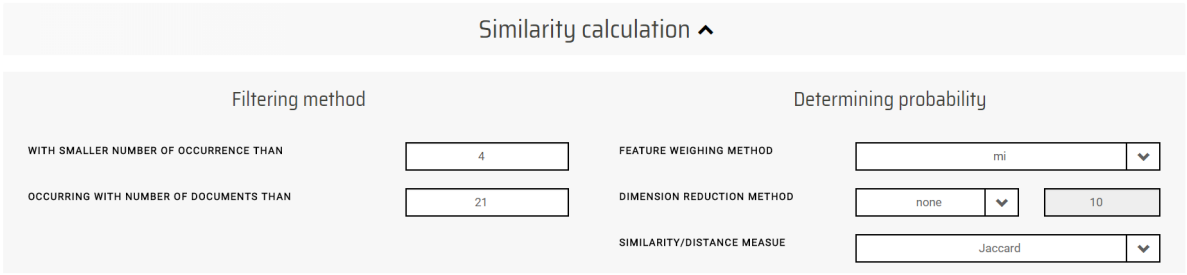

Figure 7. Filtering, weighting and similarity interface

Text similarity is computed from transformed vectors by several measures: *cosine*, *Dice*, *Jacquard*, *ratio* (a heuristics measuring the average ratio of commonality), *shd* (a heuristics measuring the

[12] http://universaldependencies.org/u/pos/
[13] Following an interesting suggestion of one of the reviewers, we can consider whether it is possible to process and compare texts in several languages in the same corpus, at the same time. Technically this is possible if only we limit the text representation only to the grammatical features. However, it should be noted that as stylometric methods can also be used to identify the source language of the translated texts, so we can expect that texts would be first clustered into groups corresponding to their languages.
[14] https://github.com/ufal/udpipe/
[15] https://github.com/ppke-nlpg/purepos

precision of mutual rendering of the two vectors). Several data analysis algorithms are based on the distance measure between vectors:

- *Manhattan, Canberra, Euclidean*,
- *Simple* (L1 on vectors normalised by a square root function) (Eder, 2016)
- *Burrows's Delta*,
- *Argamon* (Euclidean distance combined with Z-score normalisation),
- and *Eder's delta* (Eder, 2016).

WebSty also provides psychologically motivated conversion of similarity to a distance measure by arc cosine function.

4.7 Clustering

For **clustering** vectors, the combined *agglomerative-flat* clustering method from Cluto (Zhao & Karypis, 2005) was selected as it merges two perspectives: a pairwise hierarchy of similarity and a flat division into a predefined, expected number of clusters. The clustering is controlled by the three parameters: a number of clusters, a similarity measure and a clustering criterion function. A proper selection of the criterion function is a complicated issue. Thus, some ready to use defaults are provided in WebSty.

5. Data Visualisation and Exploration

The data analysis process produces (Fig. 8): similarity/distance values between texts (documents or parts) represented by a 2D matrix and the created clusters. The latter can be downloaded as an XLSX file or presented in a graphical form as a **dynamic dendrogram** – an interactive binary tree, whose nodes (individual texts or subtrees) can be collapsed or unfolded (JavaScript and *D3.js*[16] *library*). The **similarity results** are presented as: a *heatmap* (a matrix showing similarity by colours, see Fig. 9) and a *schemaball*. In the schemaball plot (Fig. 10) the user can select a file name and analyse the similarity of texts by presented connections, their colour, and thickness. For larger text collections, a multidimensional scaling can be very helpful to visualise a set of multidimensional vectors in 2D or 3D space (interactive presentation). WebSty offers four methods of the multidimensional scaling:

- *metric* – preserving distances,
- *non-metric* – preserving orders in distances, *t-distributed Stochastic Neighbor Embedding* (Maaten et al., 2008) – preserving similarities,
- and *spectral embedding* (Belkin et al., 2003) – preserving the local neighbourhood.

Results after scaling are presented (Fig. 11) as points in the 2D space (*3D.js library*) or in the 3D space. The interactive 3D plot utilises the *three.js* library, based on *WebGL* and using the user's graphic card 3D acceleration.

In response to the frequent users' questions: what features are responsible for determining a given cluster, we added a module for the **selection of important features** (Fig. 12). It is based on a set of statistical and ML-based methods. It also assumes that enough training data is provided. The implemented methods include: *statistical tests* (for example Mann-Whitney), *information metrics* (for example InfoGain), *recursive feature elimination* using supervised classifiers (like Naive Bayes) and *feature importance* assessment implemented by the tree-based classifiers, e.g. Random Forest.

[16] D3.js: https://d3js.org/, Three: https://threejs.org/, WebGL: http://www.khronos.org/registry/webgl/specs/latest/

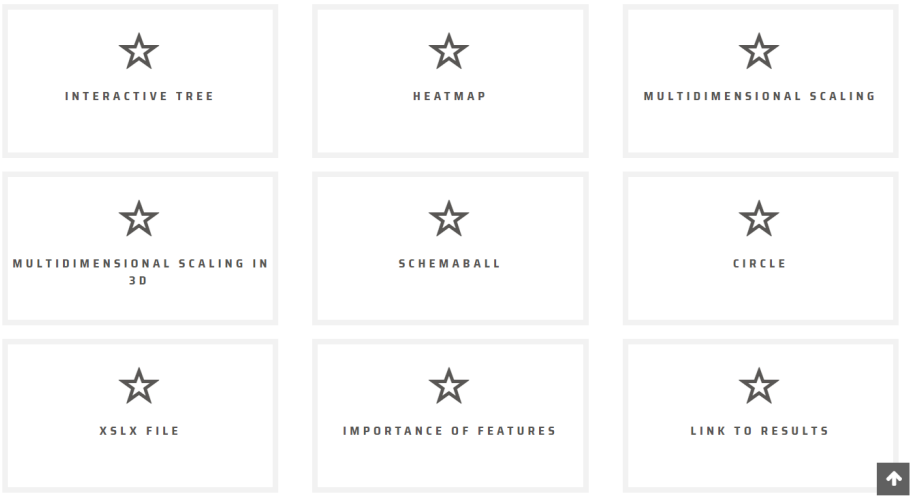

Figure. 8. WebSty results interface

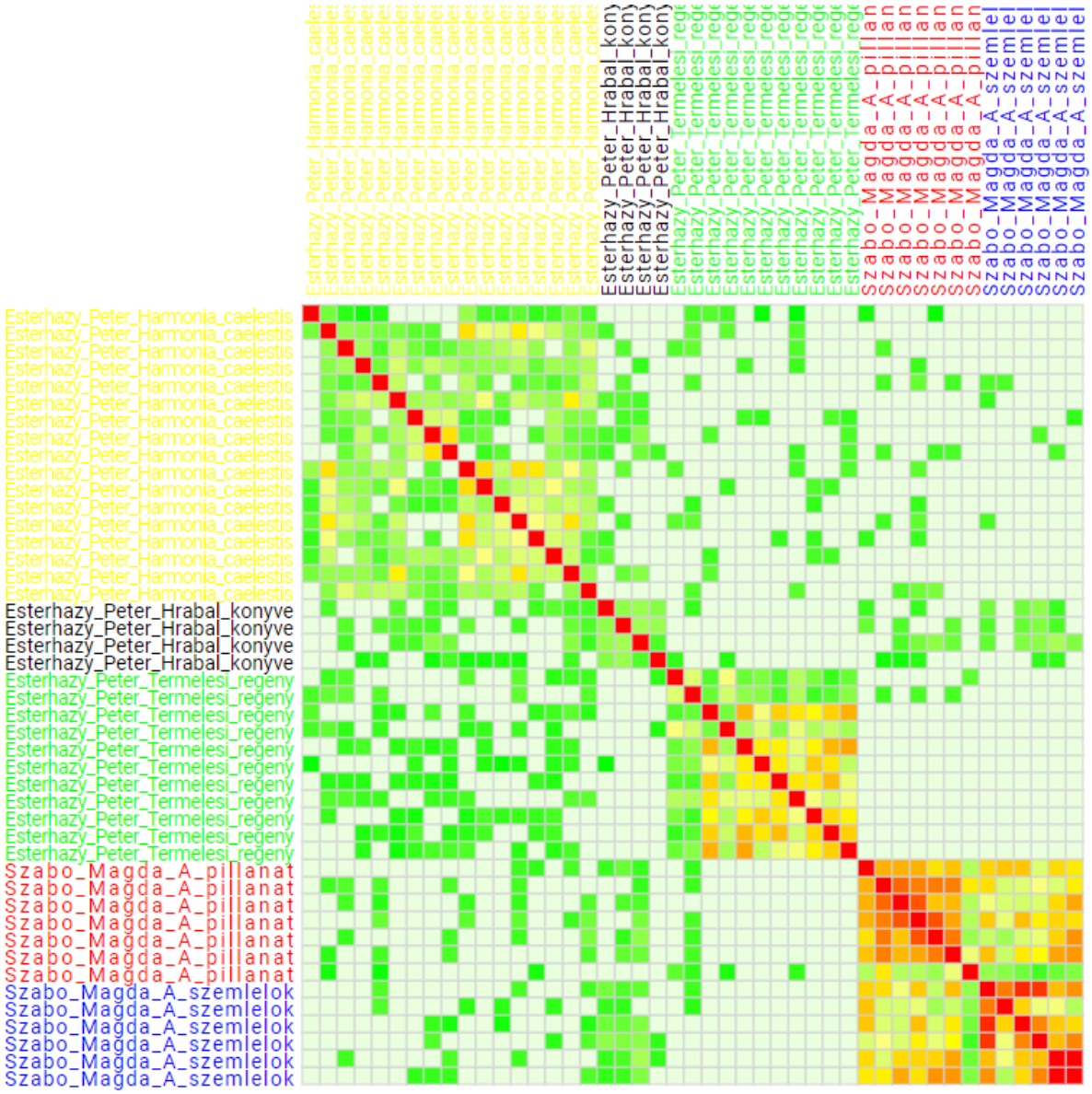

Figure 9. Similarity results in the form of a heatmap (5 books of 2 Hungarian authors, divided into chunks of size 20kB, coloured according to the book titles)

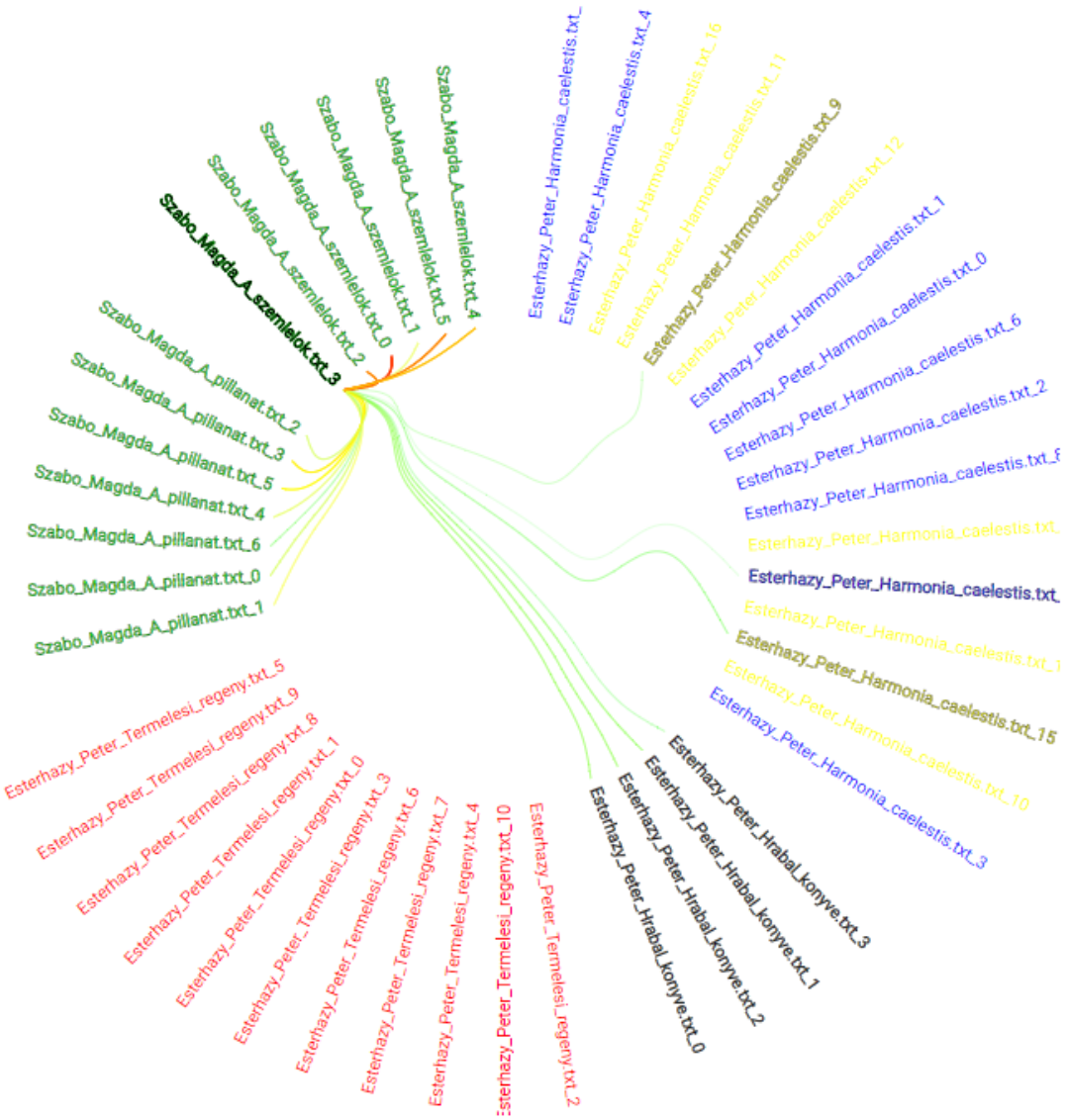

Figure 10. Similarity results in the form of a schemaball (the same corpus as in Fig. 9)

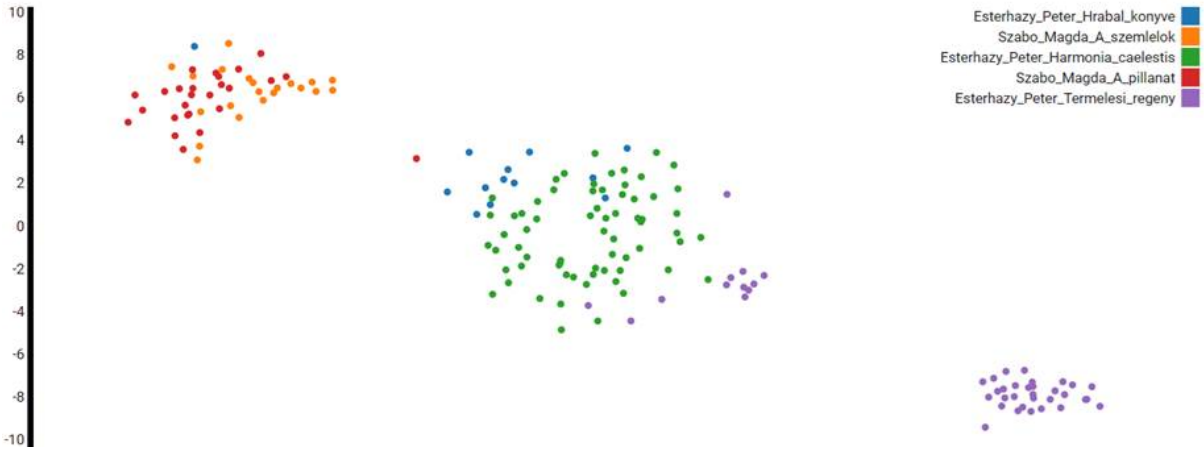

Figure 11. Distance results in the form of 2D plot (5 books of 2 Hungarian authors, divided into chunks of size 20kB, visualised using t-distributed Stochastic Neighbor Embedding)

Example results of the selection of the important features for the corpora of 5 books of 2 Hungarian authors (*Peter Esterhazy* and *Magda Szabo*), divided into chunks of the size 80kB) by the Mann-Whitney statistical test are presented in Tab. 1.

Feature	p-value	Mean in	Std in	Mean out	Std outside
base:csak	6.35E-10	0.008345	0.030443165	0.425654	0.102972377
base:míg	9.13E-10	0.02729	0.087679375	0.725347	0.241345316
bigrams:SCONJ_PROPN	2.62E-09	0.019812	0.047598872	0.557377	0.284573009
base:mikor	3.05E-09	0.007198	0.032770479	0.780015	0.378869513
bigrams:SCONJ_VERB	3.67E-09	0.017319	0.045162571	0.370154	0.15555791
base:aki	6.01E-09	0.015045	0.039769617	0.430457	0.174866081
base:ha	1.9E-08	0.033247	0.0841063	0.340515	0.127223237
base:közül	2.86E-08	0.013428	0.064323646	0.754823	0.4528463
bigrams:PRON_PROPN	4.79E-08	0.022123	0.093301757	0.505833	0.305402958
base:fel	5.34E-08	0.059155	0.111206063	0.609583	0.27429157
base:vele	6.19E-08	0.021742	0.073760873	0.527269	0.316427109

Table 1 Most important features that differentiate *Peter Esterhazy* from *Magda Szabo* (identified by the Mann-Whitney test)

6. Conclusions and further development

WebSty was implemented as a part of the CLARIN-PL infrastructure and made publicly available. Some features, like the connection to the repository and the cloud storage, are dedicated to the CLARIN-PL users. WebSty (different versions) has been already applied to several research tasks from the area of SS&H, as well as used in teaching. Among its applications, it is worth mentioning the literary analysis of the styles of web blogs (Maryl et al., 2016).

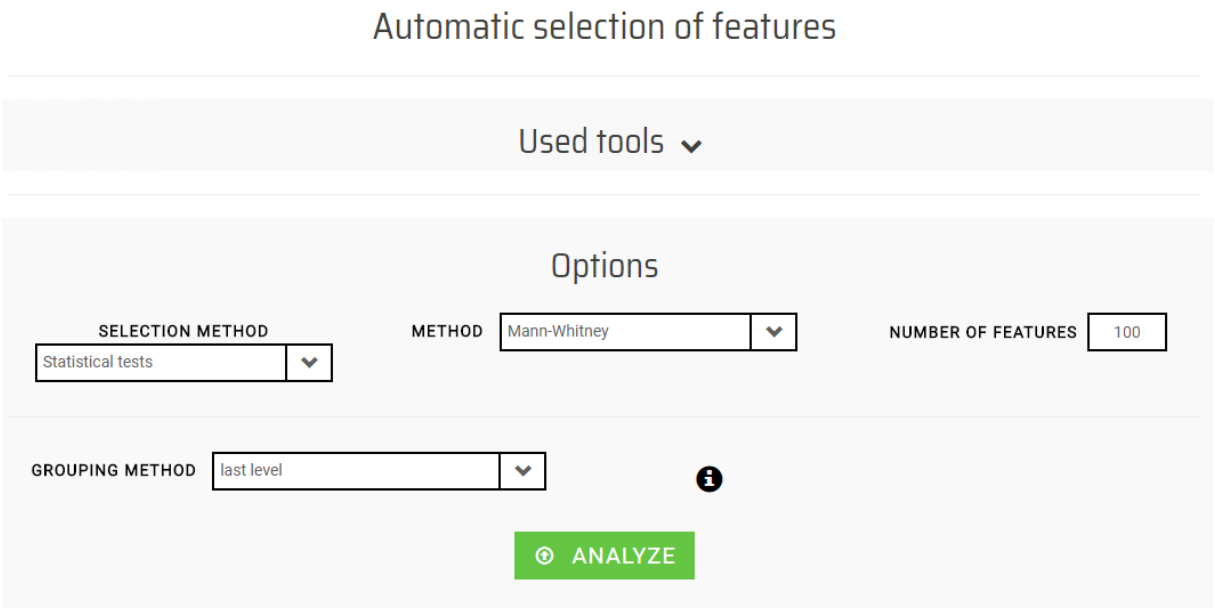

Figure 12. Importance of features interface

During the last seven months (from September 2017 to March 2018) WebSty processed ca. 2900 sets of text documents. This was 9.8 GB of texts in total, and more than 840,000 files. The largest data set consisted of ca 19,000 files and the largest one concerning its size included ca. 170 MB.
Due to the asynchronous communication and the scalable architecture based on microservices, WebSty allows for processing corpora of really large sizes. There are three factors that limit the size of an input dataset. First of all, the maximum file size (in practice 2GB of zipped text corpus) that can be uploaded to CLARIN-PL file repository[17]. Secondly, the patience of a user can determine the maximal processing time. Up till now the longest processing in WebSty was equal to 18 hours. Thirdly, the memory required to keep raw frequencies matrix. This is the most limiting factor. Now, we have assigned 64 GB of memory to virtual machines that host microservice instances responsible for processing raw frequencies matrix. The size of such a matrix depends on the number of files in the given dataset and the number of features used for the documents. That is why the most extensive feature types (all word forms or all word lemmas) are not available in the GUI. In the case of choosing recommended features (selected by default in GUI) the limit is ca. 500,000 files in one corpus.
So far, a proper usability evaluation has not been performed for WebSty. Its fast and continuous evolution was one of the reasons for this. However, only during the last year, WebSty was used during three large CLARIN-PL training workshop by more than 100 users (the vast majority of them were scholars from SS&H) under the careful observation of the WebSty developers. The experience collected during such classes was the main source of inspiration for the changes and expansions introduced. Nevertheless, proper usability evaluation is planned to be conducted.
WebSty has been so far focused on unsupervised processing by clustering. We are working on an extended version offering support for applications of the supervised approach in which classifiers are trained by ML on the basis of manually annotated data sets, e.g., sets of texts annotated with the authors' names. We also working on extending WebSty to a system enabling unsupervised and supervised semantic analysis of text data sets, e.g., identification of text fragments that are related to situations of specific types or specific phenomena. Topic analysis will also be included into WebSty as a tool of the double roles: a tool by itself and also as a tool for data preprocessing.

Bibliography

[Bell, 2010] Bell, M. (2010). *SOA Modeling Patterns for Service-Oriented Discovery and Analysis*. Wiley & Sons

[17] http://nextcloud.clarin-pl.eu

[Belkin & Niyogi, 2003] Belkin, M., Niyogi, P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6): 1373–1396.

[Eder et al., 2017] Eder, M., Piasecki, M. and Walkowiak, T. (2017). An open stylometric system based on multilevel text analysis. *Cognitive Studies | Études cognitives*, 2017(17), https://doi.org/10.11649/cs.1430.

[Eder et al., 2016] Eder, M., Rybicki, J. and Kestemont, M. (2016). Stylometry with R: a package for computational text analysis. *R Journal*, 8(1): 107–121, http://journal.r-project.org/archive/2016-1/eder-rybicki-kestemont.pdf.

[Honnibal & Johnson, 2015] Honnibal, M. and Johnson, M. (2015). An Improved Non-monotonic Transition System for Dependency Parsing. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, 1373-1378

[Landauer & Dumais, 1997] Landauer, T. and Dumais, S. (1997) A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition. *Psychological Review*, 1997, 104, pp. 211-240.

[Le et al., 2011] Le, X., Lancashire, I., Hirst, G. and Jokel, R. (2011). Longitudinal detection of dementia through lexical and syntactic changes in writing: a case study of three British novelists. *Literary and Linguistic Computing*, 26(4): 435–461.

[van der Maaten & Hinton, 2008] van der Maaten, L.J.P.; Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9 (Nov), pp.: 2431–2456.

[Maryl et al., 2016] Maryl, M., Piasecki, M. & Młynarczyk, K. (2016) Where Close and Distant Readings Meet: Text Clustering Methods in Literary Analysis of Weblog Genres. In Eder, M. & Rybicki, J. (Eds.) *Digital Humanities 2016 Conference Abstracts*, Jagiellonian University and Pedagogical University, pp. 273-275.

[Maurer, 2017] Maurer, Leon (access Apr. 2017) Web page of the StyleTool program URL: https://github.com/lnmaurer/StyleTool

[McCallum, 2002] McCallum, A.K. (2002) MALLET: A Machine Learning for Language Toolkit. Web page of the system. URL: http://mallet.cs.umass.edu.

[McDonald et al., 2012] McDonald, A., Afroz, S., Caliskan, A., Stolerman, A. and Greenstadt, R. (2012) Use Fewer Instances of the Letter "i": Toward Writing Style Anonymization. *PETS* 2012

[Manning et al., 2014] Manning, Ch. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J. and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Association for Computational Linguistics (ACL) 2014 – System Demonstrations*, ACL.

[Orosz & Novák, 2013)] Orosz, G. and Novák, A. (2013) PurePos 2.0: a Hybrid Tool for Morphological Disambiguation. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2013)*, page 539–545, Hissar, Bulgaria, 2013. INCOMA Ltd. Shoumen, BULGARIA.

[Peltz, 2003] Peltz, Ch. (2003). Web services orchestration and choreography. *Computer*, vol. 36, no. 10, pp. 46–52

[Petrov et al., 2012] Petrov, S., Das, D., & McDonald, R. (2012) A Universal Part-of-Speech Tagset. In *Proceedings of LREC 2012*.

[Pol et al., 2018] Pol M., Walkowiak T., Piasecki M. (2018). Towards CLARIN-PL LTC Digital Research Platform for: Depositing, Processing, Analyzing and Visualizing Language Data. In *Reliability and Statistics in Transportation and Communication*. Lecture Notes in Networks and System, Springer International Publishing, vol. 33.

[Przepiórkowskiet al., 2012] Przepiórkowski, A., Bańko, M., Górski, R. L. and Lewandowska-Tomaszczyk, B. (eds) (2012). *Narodowy Korpus Języka Polskiego*. Warszawa: PWN.

[Radziszewski, 2013] Radziszewski, A. (2013). A Tiered CRF Tagger for Polish. In *Intelligent Tools for Building a Scientific Information Platform*. Studies in Computational Intelligence, vol. 467, pp. 215–230.

[Sinclair et al., 2012] Sinclair, S., Rockwell, G. and the Voyant Tools Team (2012) Voyant Tools (web application). URL: http://docs.voyant-tools.org

[Stamatatos, 2009] Stamatatos, E. (2009). A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3): 538–556.

[Straka & Straková, 2017] Straka, M. and Straková, J. (2017) Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. ACL, pp. 88–99.

[Walkowiak, 2014] Walkowiak, T. (2014), Behavior of Web Servers in Stress Tests. In *Advances in Intelligent Systems and Computing* Vol. 286, Springer, pp. 467–476.

[Walkowiak, 2016] Walkowiak, T. (2016). Asynchronous System for Clustering and Classifications of Texts in Polish. In: *Proceedings of the Eleventh International Conference on Dependability and Complex Systems DepCoS-RELCOMEX*, 2016, Springer International Publishing, pp. 529–538.

[Walkowiak, 2018] Walkowiak, T. (2018). Language Processing Modelling Notation – orchestration of NLP microservices. In: *Advances in Dependability Engineering of Complex Systems, Springer International Publishing*, pp. 464-473.

[Wittenburg et al., 2010] Wittenburg, P., Bel, N., Borin, L., Budin, G., Calzolari, N., Hajicová, E, Koskenniemi, K., Lemnitzer, L., Maegaard, B., Piasecki, M., Pierrel, J., Piperidis, S., Skadina, I., Tufis, D., van Veenendaal, R., Váradi, T., and Wynne, M. (2010) Resource and Service Centres as the Backbone for a Sustainable Service Infrastructure. In Nicoletta Calzolari et al. (ed.) *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*, European Language Resources Association (ELRA), pp. 60--63.

[Zhao & Karypis, 2005)] Zhao, Y. and Karypis, G. (2005). Hierarchical Clustering Algorithms for Document Datasets. *Data Mining and Knowledge Discovery*, 10(2): 1.