EMBEDDIA

Cross-Lingual Embeddings for Less-Represented Languages in European News Media

Research and Innovation Action
Call: H2020-ICT-2018-1
Call topic: ICT-29-2018 A multilingual Next generation Internet
Project start: 1 January 2019 Project duration: 39 months

D7.6: Reusable EMBEDDIA components available through the ClowdFlows web interface (T7.4)

Executive summary

ClowdFlows is an open source online data science platform for developing and sharing of data mining and machine learning workflows that operates in Web browsers, with no need for client-side installation. It allows visual programming from software components and is therefore suitable for demonstration purposes and offering solutions to users also outside computer science. An introduction to ClowdFlows and the descriptions of EMBEDDIA components that were integrated in the first two years of the project duration were presented in Deliverable D7.4. In this deliverable, we present the software components and the workflows developed for various use cases in the last year, together with the outcomes of platform assessment by the end-users.

Partner in charge: JSI

Project co-funded by the European Commission within Horizon 2020 Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	–
RE	Restricted to a group specified by the Consortium (including the Commission Services)	–
CO	Confidential, only for members of the Consortium (including the Commission Services)	–

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825153

Deliverable Information

Document administrative information	
Project acronym:	**EMBEDDIA**
Project number:	**825153**
Deliverable number:	**D7.6**
Deliverable full title:	**Reusable EMBEDDIA components available through the ClowdFlows web interface**
Deliverable short title:	**EMBEDDIA components in ClowdFlows**
Document identifier:	**EMBEDDIA-D76-EMBEDDIAComponentsInClowdFlows-T74-submitted**
Lead partner short name:	**JSI**
Report version:	**submitted**
Report submission date:	**28/02/2022**
Dissemination level:	**PU**
Nature:	**R = Report**
Lead author(s):	**Martin Žnidaršič (JSI), Vid Podpečan (JSI)**
Co-author(s):	**Janez Kranjc (JSI), Andraž Pelicon (JSI), Senja Pollak (JSI)**
Status:	**__ draft, __ final, _x_ submitted**

The EMBEDDIA Consortium partner responsible for this deliverable has addressed all comments received. Changes to this document are detailed in the change log table below.

Change log

Date	Version number	Author/Editor	Summary of changes made
13/07/2021	v0.1	Martin Žnidaršič (JSI)	initial draft
24/09/2021	v0.2	Vid Podpečan (JSI)	fastText training widgets descriptions
04/11/2021	v0.2	Vid Podpečan, Andraž Pelicon, Janez Kranjc (JSI)	services widgets descriptions
05/11/2021	v0.3	Martin Žnidaršič (JSI)	workflows draft text
12/11/2021	v0.4	Martin Žnidaršič (JSI)	widget descriptions update
30/01/2021	v0.5	Vid Podpečan (JSI)	additional descriptions
30/01/2022	v0.5	Martin Žnidaršič, Senja Pollak (JSI)	workshop description
31/01/2022	v0.5	Martin Žnidaršič (JSI)	corrections, conclusions
02/02/2022	v0.6	Marko Robnik-Šikonja (UL)	internal review
02/02/2022	v0.7	Marko Pranjić (TRI)	internal review
21/02/2022	v0.8	Vid Podpečan (JSI), Martin Žnidaršič (JSI)	final corrections and additions
26/02/2022	v0.9	Nada Lavrač (JSI)	Quality control
28/02/2022	final	Martin Žnidaršič, Senja Pollak (JSI)	Additons after quality control
28/02/2022	submitted	Tina Anžič	Report submitted

Table of Contents

List of abbreviations

CSV	Comma separated value
D	Deliverable
LSI	Latent Semantic Indexing
LR	Learning Rate
ML Bert	Multilingual BERT
T	Task
TEXTA MLP	TEXTA Multilingual Language Processor
TNT-KID	Transformer-based Neural Tagger for Keyword Identification
WP	Work Package

1 Introduction

Software that is developed in the scope of EMBEDDIA offers functionalities that are interesting and of value to various users, from professional users and software integrators, to researchers and practitioners in various disciplines of science. In order to provide access and ability of experimentation with our solutions also to the users that are not sufficiently proficient in programming to be able to exploit our libraries and APIs, we provide selected software components also in the form of software components of a visual programming platform named ClowdFlows (Kranjc et al., 2012; Kranjc, 2017). Availability of our software components in ClowdFlows is primarily intended to allow their use by practitioners outside computer science that work with natural language, for example by researchers in humanities. With its graphical user interface, modular component design and ease of sharing of developed solutions, ClowdFlows is well suited for experimentation, demonstration purposes and use in education.

The work presented in this document was performed the scope of Task T7.4 in EMBEDDIA's work-package (WP7) that is focused on disseminating and sharing the results of the project with various potential users. Task T7.4 is aimed at ensuring sustainability and re-usability of specific components and solutions also beyond the media context by making them available in ClowdFlows. The concepts and components considered in T7.4 originate from most of the other work-packages in the project and tasks that resulted in ready to use software, particularly T1.1, T2.2, T3.1, T3.2, T4.3 and T6.2.

The ClowdFlows platform introduction, the intermediate state of integration of EMBEDDIA's software components and initial workflows were described in Deliverable D7.4. In this deliverable we report on the final selection, implementation and integration of components and workflows, as well as on the feedback of the target users. Apart from listing the previously covered components, we do not describe the components presented in D7.4, and provide descriptions only for the ones developed afterwards. Therefore, for a complete overview of integration of EMBEDDIA components in ClowdFlows, both deliverables (D7.4 and this one D7.6) should be considered.

The rest of this deliverable is organized into four sections. In Section 2, we list previously existing widgets and describe the ones added since D7.4. Selected public workflows developed using these components are described in Section 3. Section 4 describes the workshop organized to demonstrate EMBEDDIA ClowdFlows components to target users and to collect their feedback. Conclusions are made in Section 5.

2 New EMBEDDIA components in ClowdFlows

In this section we present the widgets (software components that form units of graphical workflows) developed in the EMBEDDIA project. We first list the widgets that were already reported in Deliverable D7.4 and then focus on newly added widgets in the subsequent subsections.

In D7.4 we reported on the following widgets:

- BERT
- BERT Embeddia
- ML BERT for hate speech
- ML BERT for sentiment classification
- Doc2Vec
- LSI
- ELMo
- ELMo Embeddia
- Universal Sentence Encoder
- GloVe
- Word2Vec
- fastText
- fastText Croatian
- fastText Embeddia
- fastText Slovenian
- Punkt Sentence Tokenizer

- Regex Word Tokenizer
- Tok Tok Word Tokenizer
- Concatenate embeddings
- Create Dataset
- Create scikit bunch
- Export dataset
- Import dataset
- Language
- Load Corpus from CSV
- Token Filtering
- Load file
- Display String
- Filter data by label
- Join list of strings
- Word cloud

2.1 Support for training fastText models

In the scope of deliverable D7.4 we introduced a number of components that can provide pretrained text embeddings, for example ELMo, Word2Vec, and various pretrained BERT and fastText models. These general pretrained models are useful in many contexts. However, the models trained on specific corpora are also of great importance, particularly for very specific tasks. Training of new embedding models requires huge datasets, significant computational power, and takes large amounts of time. Large dataset transfers and processing time latencies do not fit well the interactive online scenarios in ClowdFlows. Nevertheless, to enable experimentation with custom embeddings models, we developed a component that builds new embedding models with the fastText algorithm, which is one of the most efficient and least computationally demanding methods of this kind. In order to ensure near real-time training of a fastText model it is recommended that the input corpus does not exceed 2M words or approximately 10MB of raw input text. While this seems tiny in comparison with typical corpora used in training embedding models, it turns out that typical ClowdFlows end users from the digital humanities domain often work with even smaller and very specific corpora such as poetry collections, specialized news articles, etc. (see Section 4). Therefore, the widget for training fastText models also contains hints how to adjust the default parameters when training on small corpora. In the following, we describe the software component for training such models and two additional components that extend the use of fastText embeddings beyond what was reported in D7.4.

2.1.1 Train fastText

The *train fastText* widget trains a new fastText model on the input corpus. The resulting model can be passed on to other widgets to perform specific tasks. The input to this widget is a text corpus, such as e.g., the results from the *Load Corpus from CSV* widget. The corpus can be tokenized, lemmatized or used without such preprocessing steps.

The settings available for the *train fastText* widget, as shown in Figure 1, are the following:

bucket sets the number of buckets (word and character ngram features are hashed into a fixed number of buckets);

epoch sets the number of training epochs;

lr sets the learning rate;

dimension sets the size of word vectors;

window sets the size of the context window;

model sets the type of the unsupervised fasttext model (cbow or skipgram), and

min_count sets the minimal number of word occurrences to take the word into account.

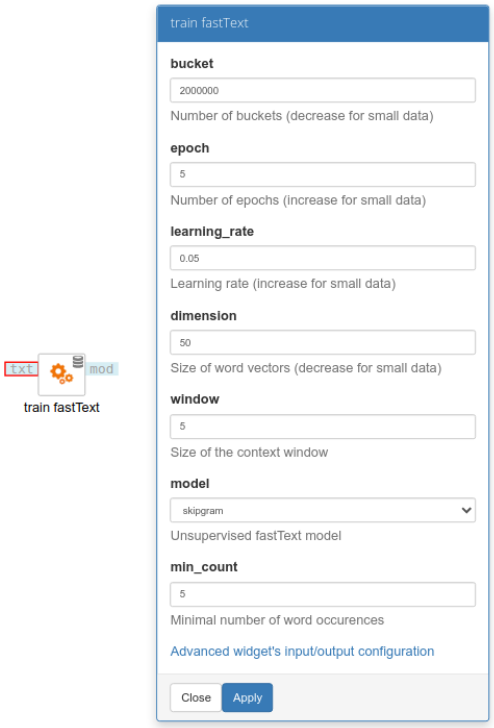

Figure 1: The *train fastText* widget and its settings.

Where applicable, a description of a parameters includes a hint whether it is advisable to increase or decrease the value when the training data is small.

There is an *Advanced widget's input/output configuration* link at the bottom of the settings window of each widget. This opens a form in which we can change the order of inputs or outputs to a widget or change whether we want an input value to be considered an input (like the *txt* input shown on the left side of the widget in Figure 1) or a parameter (to be entered in the settings window such as the one on the right side of Figure 1).

2.1.2 FastText neighboring words

This widget returns top k neighboring words for each input word, according to the given fastText model. This way we obtain the words that in the training corpus appear in similar contexts as the provided input words.

There are two parameters that a user can set for this widget (see Figure 2):

Number of neighbors sets the value of k of the returned neighbors,

Threshold is a parameter for the optional filtering of the neighbor list by edit distance. Namely, depending on the corpus, the preprocessing, and the used language, the neighbors of a given word can in some situations consist predominantly of its variations. If such neighbors are not of interest, we can set the *Threshold* parameter to a value lower than 1, which will cause filtering out words similar to the given word. Value of 1 causes no filtering, value of 0 is the strictest filtering and the values in between cause intermediate levels of filtering. The edit distance-based formula to compute the similarity between words is defined as $1 - \frac{d_{edit}(w,w')}{max(len(w),len(w'))}$. When the result of this expression exceeds given threshold the word is considered too similar and is not added to the list of neighbors.

Figure 2: The *fastText neighbouring words* widget and its settings.

2.1.3 Evaluate word expressions with fastText

As the fasttext python library does not offer the nearest neighboring functionality for word vectors, this widget implements it by obtaining vectors of all the words of the model and then applies vector and matrix operations (normalization and dot product) to compute the result.

This widget computes the result of an expression which uses numerical operations on words (internally represented as fastText embedding vectors), and returns a list of words that have embeddings which are the most similar to the numerical vector that represents the result of the expression.

It evaluates vectors of word expressions such as "king - man + woman" using the given fastText model. Two operators are currently supported: + and -. The results can be converted back to words and filtered using the specified threshold for the distance measure. As shown in Figure 3, the parameters are similar as in the case of the neighboring words widget:

Threshold is the edit distance filtering threshold (see explanation in Section 2.1.2).

Result size stands for the desired amount of terms that are the closest to the computed result.

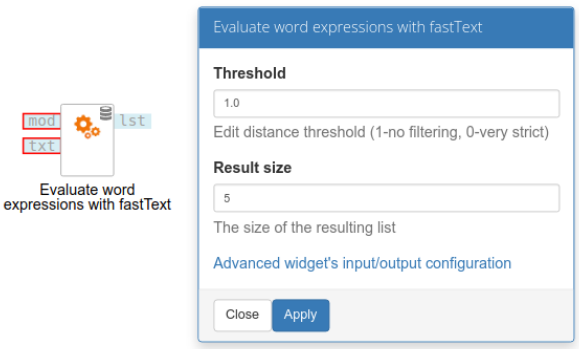

Figure 3: The *Evaluate word expressions with fastText* widget and its settings.

2.1.4 Apply trained fastText models

This widget applies a trained fastText model to a tokenized corpus and returns a matrix with document embedding vectors. Here, a document embedding is computed as an aggregation of embeddings of all tokens of the document. The only available setting is:

Aggregation method which defines how the token embeddings are to be combined. It supports aggregation by summation or by average.

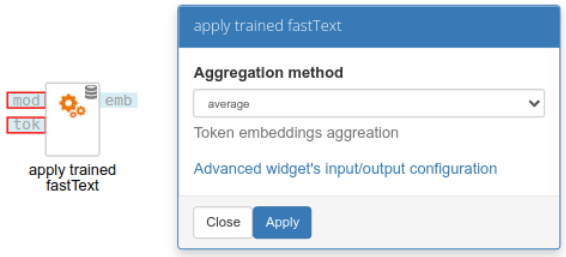

Figure 4: The *apply trained fastText* widget and its settings.

2.2 Selected Web service based components

The widgets described in this subsection offer a ClowdFlows interface to selected REST Web services that were developed in EMBEDDIA. These Web services are described on the EMBEDDIA Media Assistant Web page[1], and in Deliverable D6.9, and are intended for professional and research use. In ClowdFlows, they are offered as widgets for demonstration purposes and to make their application and re-use possible for users that are not proficient in programming and using Web service APIs.

As these widgets call external (to ClowdFlows) Web services, their operation depends on service availability, connectivity and access limitations.

2.2.1 Author profiling

This widget implements the interface to a service for English and Spanish author profiling that can determine whether a tweet or comment was written by a bot, male or female based on the system by Martinc et al. (2019). The widget expects a list of strings as input, even if there is only one string, so the input must be prepared accordingly. A simple, robust and flexible approach is to use the *Create List* widget as in the example shown in Figure 5. The widget does not have any parameters that could be defined by the user.

2.2.2 Comment analyzer

The *Comment analyzer* widget is a wrapper around the comment analyzer service developed in EMBEDDIA (Pelicon et al., 2021). It classifies the input texts as "OFFENSIVE" or "OK" (see example in Figure 6). The widget does not have any parameters.

2.2.3 Keyword Detection

This widget is a wrapper for a number of keyword detection services. Unlike the *Author profiling* and *Comment analyzer* widgets, this widget expects a simple string as input, as provided for example by the *Create String* widget. Particular services can be selected as a user defined parameter Analyzer. RaKUn is implementing the unsupervised system by Škrlj et al. (2019), while TNT-KID (Martinc et al., 2021) was trained for several languages, where the Estonian and Latvian versions implement the TNT-KID extension with tagset matching by Koloski et al. (2021).

[1] https://embeddia.texta.ee/

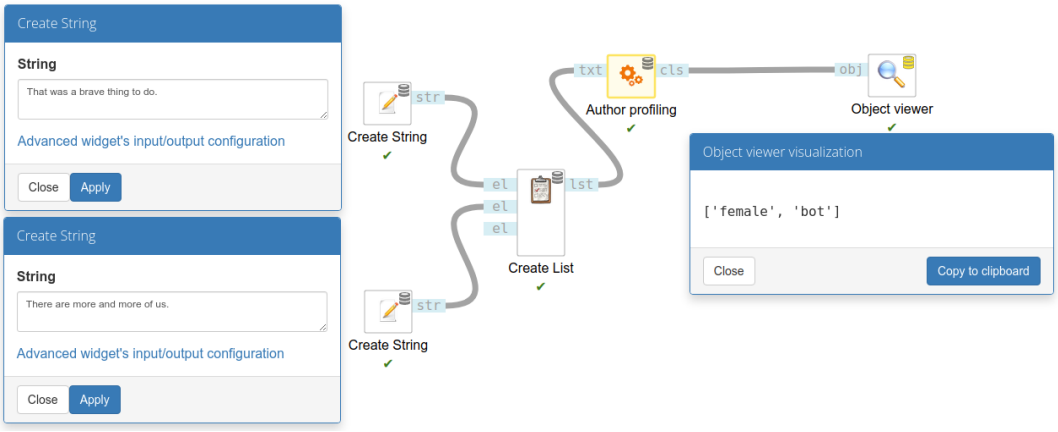

Figure 5: The *Author profiling* widget in an example of its use. The widget expects a list of strings even in case of a single string, which can be ensured by running the texts through the *Create List* widget as in this small workflow.

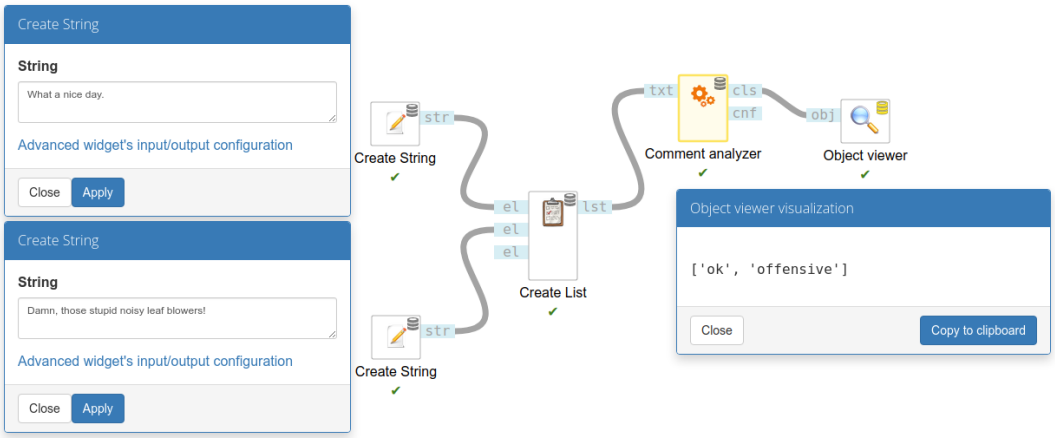

Figure 6: The *Comment analyzer* widget in an example of its use.

Analyzer provides the name of the algorithm for keyword detection. The choices are shown in Figure 7 and are the following:

- RaKUn (multilingual)
- TNT-KID (Croatian)
- TNT-KID (English)
- TNT-KID (Estonian)
- TNT-KID (Latvian)

2.2.4 Named Entity Recognition

This is a wrapper for named entity recognition services developed in EMBEDDIA. It expects raw string as input and outputs a list of tuples with the entity and its type, like in the example shown in Figure 8. The widget has only one parameter:

Analyzer which is the name of the algorithm for named entity recognition. The only choice currently is TEXTA MLP (Multilingual).

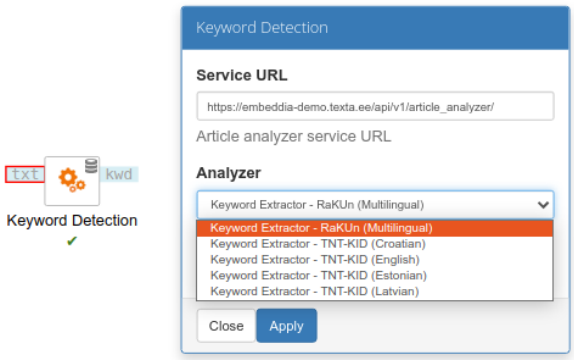

Figure 7: The *Keyword Detection* widget and its settings.

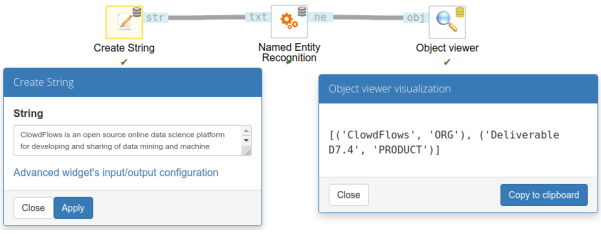

Figure 8: The *Named Entity Recognition* widget in an example of use. In this case, the Executive Summary of this deliverable is used as input text.

2.3 Support widgets

Since Deliverable D7.4, we created a few support features and widgets. A new feature in the results views of the *Display String* and *Object Viewer* widgets is the *Copy to clipboard* button, which allows the users to copy the entire contents of the presented text on to the clipboard. Copying of the resulting texts proved to be cumbersome in some situations, such as when a lengthy text is shown on the widget's canvas. The newly added widgets are described in the subsections below.

2.3.1 Load corpus from ZIP

This widget loads a corpus from the files in a given ZIP archive. This allows for loading from a multitude of files.

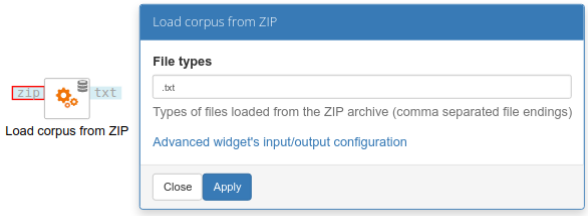

Figure 9: The *Load corpus from ZIP* widget and its settings.

The widget offers a setting:

File types that defines the types of files that are to be loaded from the provided ZIP archive. These are

specified as a comma separated list of file extensions (see Figure 9 for an example). The default recognized file type is .txt.

2.3.2 Token frequency

This widget outputs the tokens and their frequencies in the text of a given corpus. It does not have any parameters. The output is sorted by the descending frequency and can be viewed with *Display String* or *Object Viewer* widgets, as in the example shown in Figure 10.

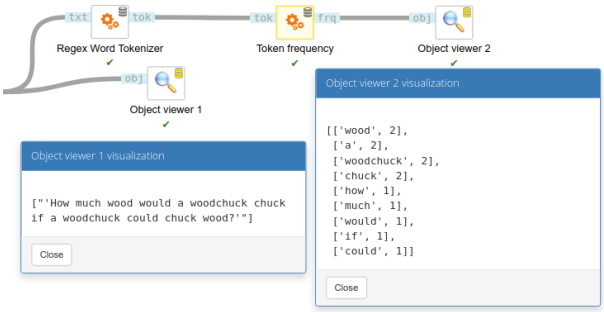

Figure 10: The *Token frequency widget* widget shown with an example of an input and output.

2.3.3 Detokenize

The *detokenize* widget joins lists of tokens into a string. This is useful when—after processing the text in a form of tokens—we need it in the form of a string, e.g., a widget might expect string (and not token) inputs. The need for such widget arises from the fact that for performance reasons ClowdFlows uses basic data structures which do not store meta information or collect intermediate results. Note that the widget cannot take into account the strategy used for tokenization and thus joins the input tokens with a single white-space character.

3 Selected EMBEDDIA workflows

In Deliverable D7.4, we described two initial workflows that were designed with EMBEDDIA components. In the following, we outline some of the public workflows that were developed since. Most of them were used in publications or at software demonstration events.

3.1 Token frequency analysis

There is a series of three new public workflows aimed at token frequency analysis, which we developed to serve as examples in a recently submitted paper about ClowdFlows and its natural language processing components that were developed in the scope of EMBEDDIA.

Figure 11 shows all three workflows. It shows the basic ClowdFlows text processing concepts, starting from loading data from a file with the *Load file* widget. In the example, this is a file with the first 1000 comments from April 2017 from the New York Times Comments dataset[2]. This is followed by the *Load Corpus from CSV* widget that skips any existing header and non-comment data columns. The resulting text is sent to a tokenizer, which produces the tokens. These are used further in three workflow branches.

[2] https://www.kaggle.com/aashita/nyt-comments

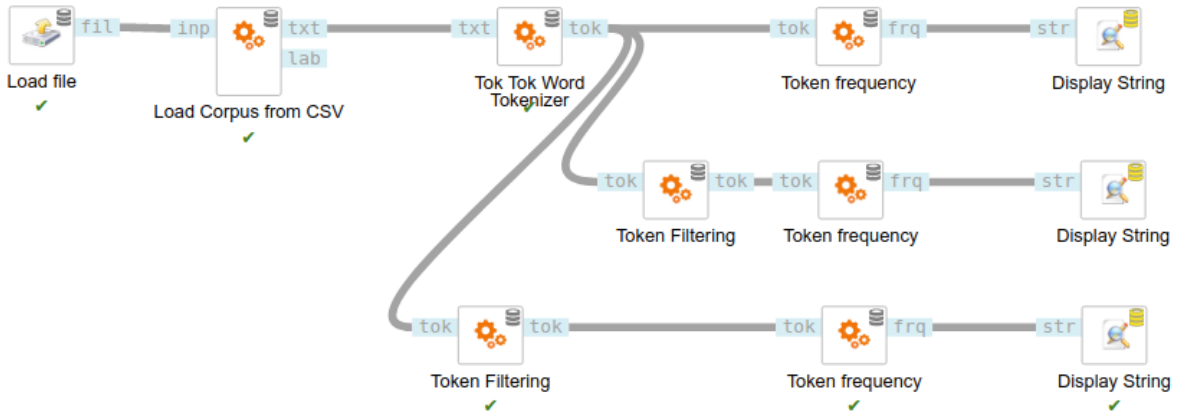

Figure 11: The token filtering workflow. Available at: `https://cf3.ijs.si/workflow/226`

In the first branch, the tokens directly enter the *Token frequency* widget, described in Section 2.3.2. The widget outputs the tokens and their frequencies. The output is sorted by the highest frequency and is visualized with the *Display String* widget. The first ten tokens and their frequencies for the given data are shown in Table 1.

Table 1: Most frequent tokens of the first branch of the workflow from Figure 11.

token	the	,	to	and	of	"	a	’	is	in
frequency	2814	2075	1419	1348	1244	1060	1011	950	834	797

As the results of the first branch contain tokens that are not words, a *Token Filtering* widget is added to the second branch to filter out the punctuation characters. However, its output still contains stopwords.

The third branch allows filtering of user-defined tokens that are added as parameters in the *Token Filtering*. In this case the results are much more useful, as shown in Table 2.

Table 2: Most frequent tokens of the third branch of the workflow from Figure 11.

token	trump	people	one	like	venezuela	would	get	us	time	could
frequency	408	191	138	126	123	114	97	94	81	79

3.2 Word neighbors in a trained fastText model

In Section 2.1.1 we presented the *train fastText* widget, which allows for training of a custom fastText model. This functionality, with various combinations of filtering and text processing (such as lemmatization) can be used for various analyses and studies. Normally, these are out of reach of the users that are not proficient in computer programming.

To demonstrate the functionality of the *train fastText* widget and to offer its functionality in a ready-to-use workflow, we created an exemplary public workflow that uses English texts from the BuzzFeed-Webis Fake News Corpus (Potthast et al., 2018a,b). The workflow is shown in Figure 12.

This workflow loads a corpus, learns a custom fastText model and then uses the model with widgets that offer the computation of a word neighborhood and evaluation of word expressions.

The corpus is loaded and parsed (column and cell separator selection) with the *Load Corpus from CSV* widget. This data is used in two branches - in the first, the language is detected with the *Detect language*

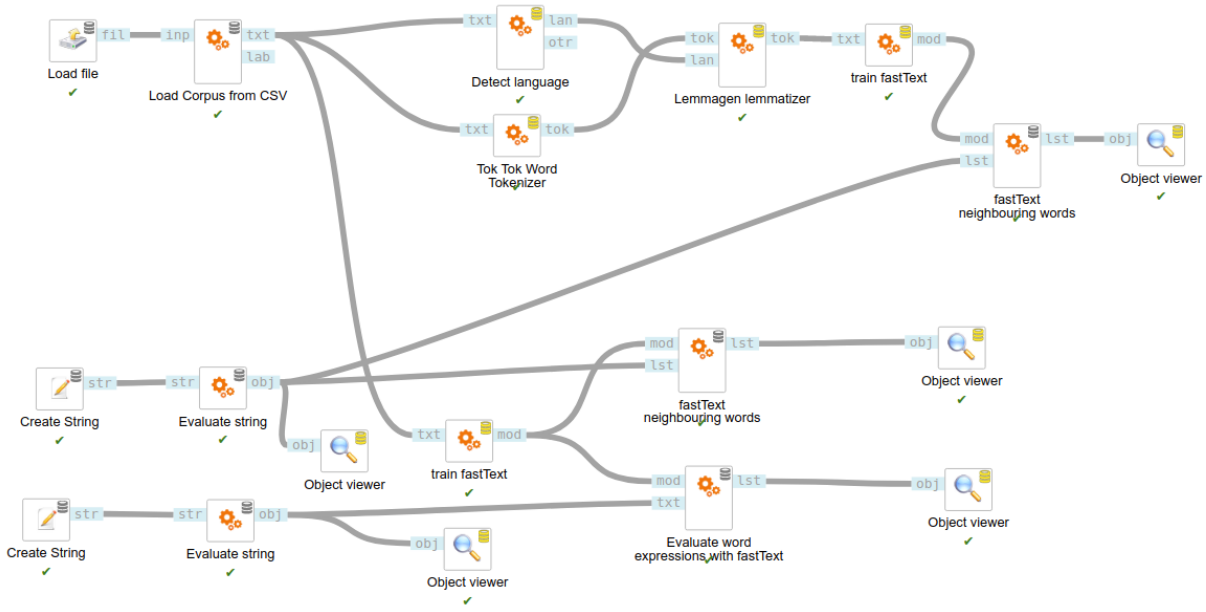

Figure 12: The "train fastText" workflow. Available at: `https://cf3.ijs.si/workflow/253`

widget (based on a language-detection library (Shuyo, 2010)) to activate a corresponding language model in the *Lemmagen lemmatizer* widget that implements Lemmagen lematizer (Juršič et al., 2010). The fastText model is trained using the results. The second branch skips lemmatization and trains the fastText model on non-lemmatized data. In both cases, the *fastText neighbouring words* widget obtains neighbors of the words from the input list. The branch in the lower part of the canvas demonstrates word expression evaluation with the *Evaluate word expressions with fastText* widget.

3.3 Extended embeddings experimentation

In Section 4.1 of Deliverable D7.4, we presented the Embeddings experimentation workflow demonstrating how ClowdFlows can be used to experiment with various machine learning algorithms and embeddings. With the availability of the *train fastText* widget, we could create an extended version that shows experiments with custom trained embeddings (see Section 2.1.4). An example of such an extension is shown in Figure 13.

The results of this single custom trained fastText model in the given setting are good, but still inferior to the results obtained with concatenation of embeddings from four other general embedding models. The evaluation on a Slovene news sentiment classification task with a stratified 75/25 split results in a 3% lower F1 in comparison with the concatenated model.

3.4 Application in the financial text analysis

The *train fastText* widget was demonstrated also in the domain of financial text analysis, which shows the exploitation potential for ClowdFlows out of the NLP domain.

It was used in a study of approaches for detection of forward-looking sentences in financial texts with word lists. The aim of this task is to analyze the given input text (a common example is an annual report of a company) and output the sentences that contain forward-looking expressions, like expectations, estimations or predictions.

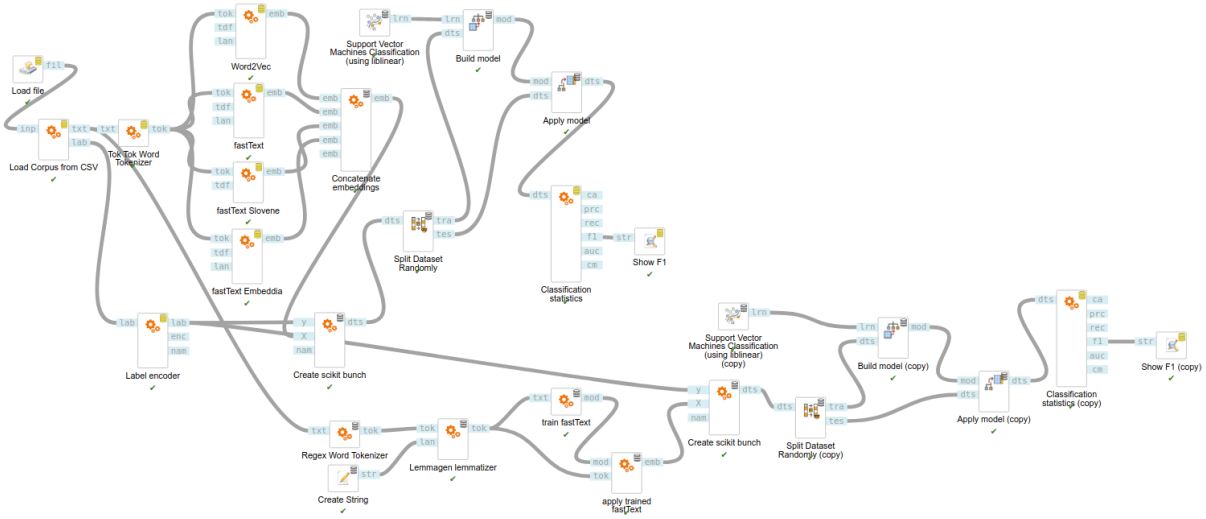

Figure 13: The extended experimentation workflow using embeddings. Available at: `https://cf3.ijs.si/workflow/186`

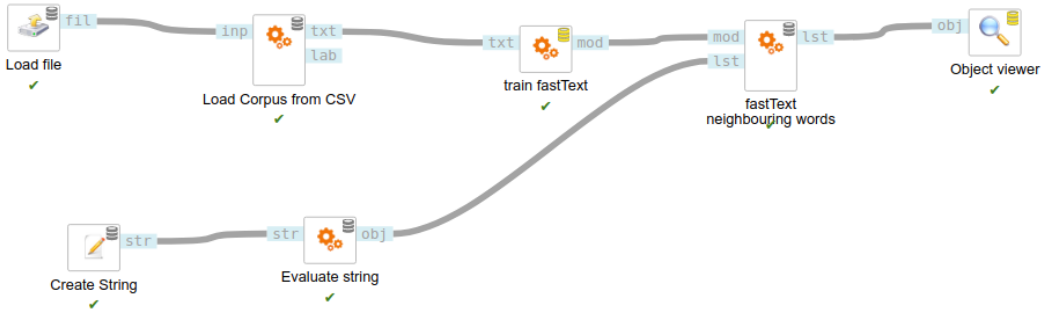

Figure 14: The workflow of the application in domain of analysis of financial texts. Available at: `https://cf3.ijs.si/workflow/223`

.

In the study we experimented also with extensions of word lists with the neighboring words according to a fastText model that was trained with a *train fastText* widget on a corpus of annual reports. The workflow (shown in Figure 14) was described in the paper by Štihec et al. (2021).

3.5 Author and comment analysis

To demonstrate the *Author profiling* and *Comment analyzer* EMBEDDIA Web services, we created a public workflow shown in Figure 15.

The workflow first implements the steps that are necessary for preparing the inputs for these two components, as it must be in a corpora representation format. In this particular case when individual inputs are used the input strings are concatenated into a list. Then, the resulting collection of data is sent to both services and the results are displayed.

For example, when using the following inputs:

1. How dare you!
2. What a nice day!

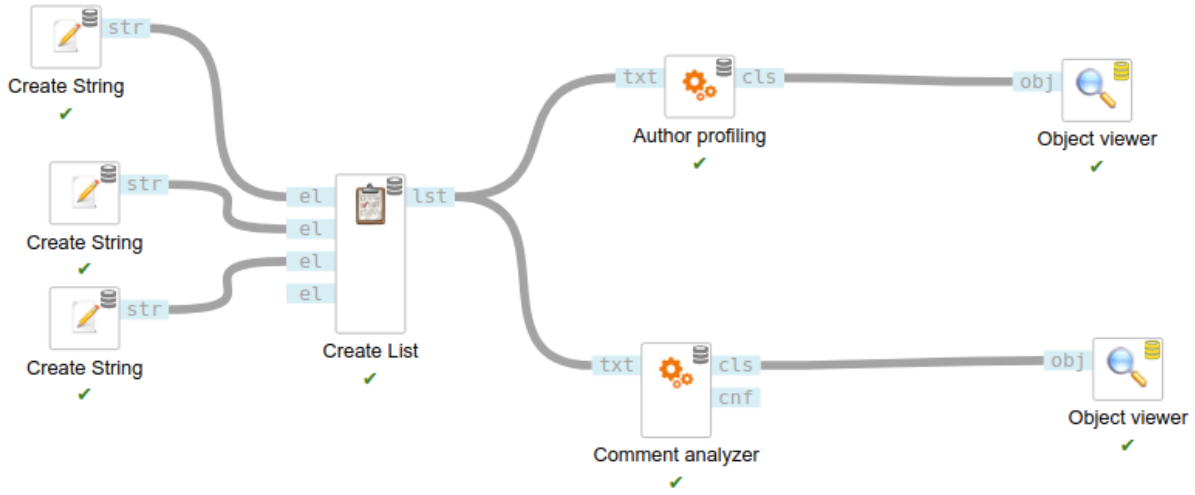

Figure 15: The workflow with EMBEDDIA services. Available at: `https://cf3.ijs.si/workflow/309`

.

3. Please log in to continue.

the workflow will display

1. ['female', 'male', 'bot']
2. ['offensive', 'ok', 'ok']

as the results returned by the two services. Note, however, that while such input is perfectly valid for both services, it is recommended to use longer text or a collection of texts by the same author for the *Author profiling* service in order to obtain reliable results.

3.6 Workflow for hands-on experimentation

Usability assessment of ClowdFlows and its software components provided by EMBEDDIA was conducted in a workshop with target end-users. Details about the workshop are provided in Section 4.

For the purposes of the workshop, we prepared a public workflow, which introduced ClowdFlows, the use of embeddings, and presented initial exploration of user-provided corpora. To cater to different formats of the input texts that the participants might want to experiment with, the workflow has two input branches (see Figure 16), one for texts that are loaded from files and another for directly entered texts. The rest of the workflow mostly corresponds to the one in Section 3.2.

4 User assessment

Assessment of ClowdFlows and the most important components provided by EMBEDDIA from the viewpoint of the targeted end-users was done in the scope of a one-day workshop, held online on 27th of January 2022.

The workshop was aimed at one of our primary target groups: researchers from various fields of humanities, with unknown proficiency in programming. The users were not expected to be proficient enough in programming to benefit from open APIs and programming libraries, but have the need for corpora analysis and many ideas for studies in which EMBEDDIA software components might be employed.

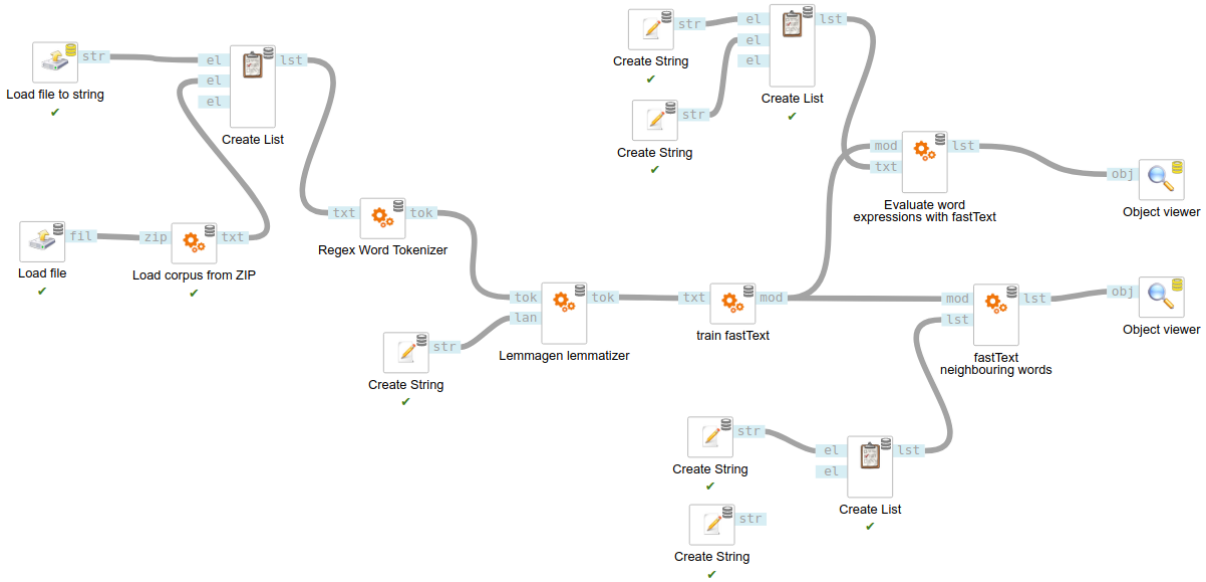

Figure 16: The workflow that was used in the workshop with end-users. Available at: `https://cf3.ijs.si/workflow/283`

.

Initial set-up of the workshop consisted of a common introductory presentation of ClowdFlows and the workflow from Section 3.6 , which was planned to take 20 minutes, followed by 8 individual 20 minute sessions in which each user would create its own instance of the workflow, load a corpus of interest and try some ClowdFlows components, hopefully reaching the ability to proceed independently. One individual session was dedicated to one user and the user's data of choice, but other users could be present as observers. The users in individual sessions were assisted, if they experienced difficulties with the use of the platform or with preparation of their input data. Participation in the workshop was by invitation. Participants that were invited to the workshop were all unrelated to EMBEDDIA or JSI directly, but their contacts were known to the JSI EMBEDDIA team from past collaborations, conferences or user initiated expressions of interest for language processing tools. None of the participants had previous experience with ClowdFlows. Due to high interest, the number of sessions was increased to 10.

Various kinds of corpora were used and the participants were interested in diverse aspects of text processing. In most cases other widgets were added to the initial workflow to resolve an issue or cater to specific interests. There were two specific technical issues uncovered during workshop: (I) errors occured in case of non-UTF input text with special characters, (II) some web services reported timeout.

An anonymous questionnaire was prepared for the participants and the link to the questionnaire was provided after the workshop. The questions and summarized responses are provided in Figures 17—21. Most of the participants found the demonstrated workflow very useful. A large majority of them (80%) never tried using embeddings before. The ClowdFlows user interface was mostly reported as easy to use, with only one finding it difficult. This, however, needs to be considered in the context of the participants responding shortly after using ClowdFlows with assistance. Without introduction and assistance, the responses might not be so beneficial. Most of the participants said that they would use ClowdFlows again, if provided with a workflow for their problem of interest and intends to continue using ClowdFlows.

Did you find the demonstrated workflow useful?

10 responses

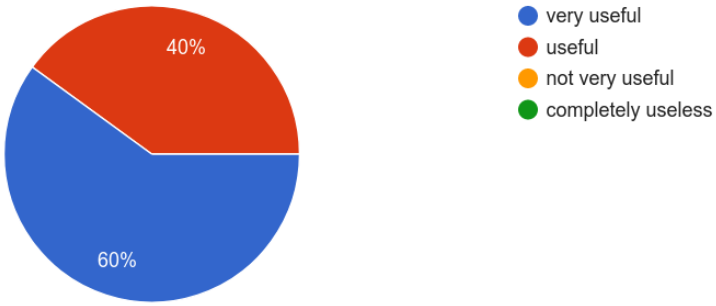

Figure 17

Did you ever perform these kinds of analyses (development and use of word embeddings) before?

10 responses

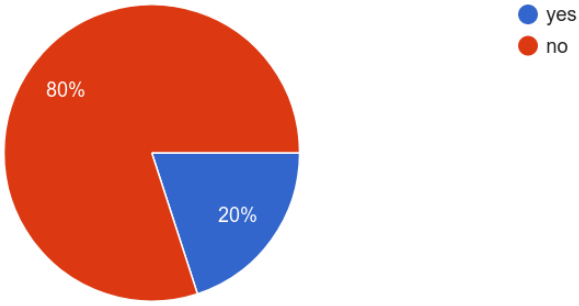

Figure 18

How do you find the ease of use of the ClowdFlows user interface?

10 responses

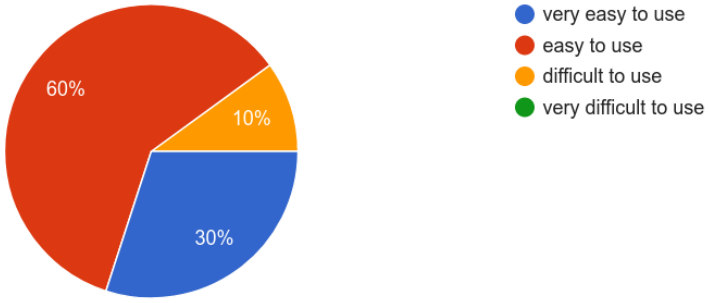

Figure 19

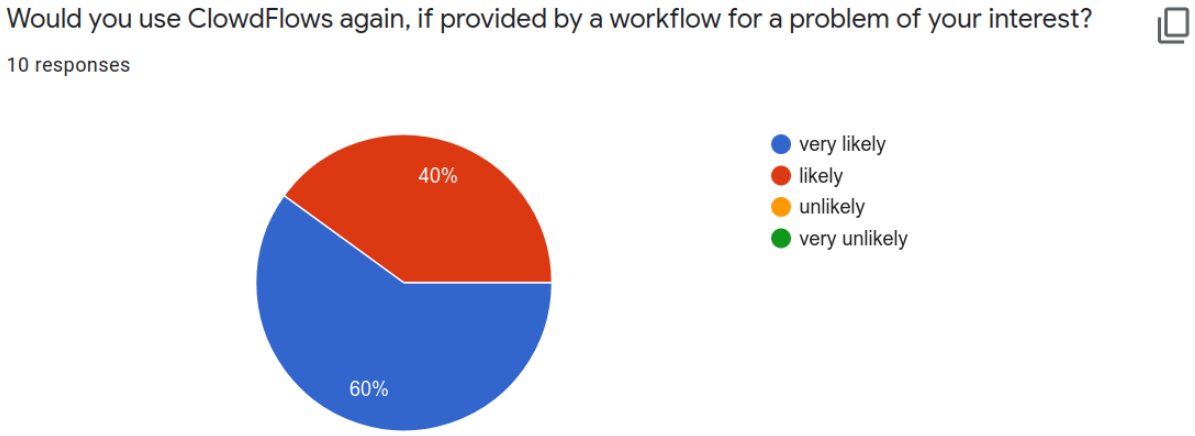

Figure 20

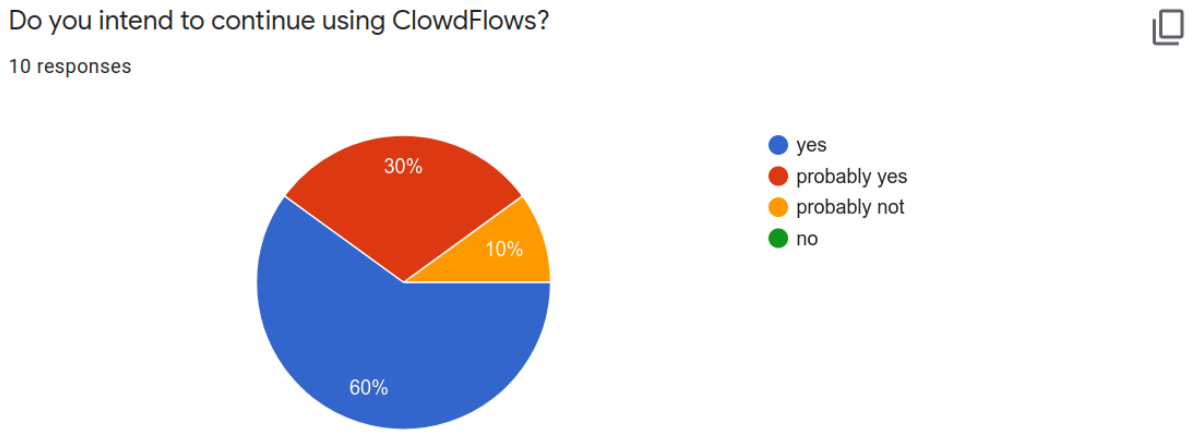

Figure 21

5 Conclusions

Many software components stemming from the work in EMBEDDIA were included in ClowdFlows, together with a number of support components that were developed to enable or facilitate their use. During the reported time, we focused less on the implementation of further analytic components, but rather on development of workflows that can demonstrate the work done in EMBEDDIA and offer it to users outside the computer science.

During the course of the project there were 42 widgets and 8 public workflows added to ClowdFlows. We encountered also various issues, bugs and glitches related with the incorporation of EMBEDDIA components into ClowdFlows. There are also some yet unmet wishes expressed by our users, such as more visualizations, clearer indications of requested input formats and further user guides. While the work in the first half of the project provided mostly analytical components, the work in the second half included more interactions with the target users and provided confirmations that the developed solutions are interesting, useful and can support our target users.

6 Associated outputs

The work described in this deliverable has resulted in the following resources:

Description	URL	Availability
ClowdFlows3 (online Web app.)	https://cf3.ijs.si/	Public
ClowdFlows3 (server-side code)		
backend	https://github.com/xflows/clowdflows-backend	Public (MIT)
frontend	https://github.com/xflows/clowdflows-webapp	Public (MIT)
Docker version	https://github.com/xflows/clowdflows-docker	Public (MIT)

Some of the work outlined in this deliverable is published in the following publication:

Citation	Status	Appendix
Štihec, J., Pollak, S., Žnidaršič, M. (2021). Preliminary experimentation with combinations and extensions of forward-looking sentence detection wordlists. In Proceedings of the 3rd financial narrative processing workshop(pp. 26–30).	Published	Appendix A

References

Juršič, M., Mozetič, I., Erjavec, T., & Lavrač, N. (2010). Lemmagen: Multilingual lemmatisation with induced ripple-down rules. *J. Univers. Comput. Sci.*, *16*, 1190-1214.

Koloski, B., Pollak, S., Škrlj, B., & Martinc, M. (2021, April). Extending neural keyword extraction with TF-IDF tagset matching. In *Proceedings of the eacl hackashop on news media content analysis and automated report generation* (pp. 22–29). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2021.hackashop-1.4

Kranjc, J. (2017). *Web workflows for data mining in the cloud: Doctoral dissertation* (Unpublished doctoral dissertation). Jožef Stefan International Postgraduate School.

Kranjc, J., Podpečan, V., & Lavrač, N. (2012). ClowdFlows: A cloud based scientific workflow platform. In P. A. Flach, T. Bie, & N. Cristianini (Eds.), *Machine learning and knowledge discovery in databases* (Vol. 7524, pp. 816–819). Springer Berlin Heidelberg.

Martinc, M., Škrlj, B., & Pollak, S. (2019). Fake or not: Distinguishing between bots, males and females. In L. Cappellato, N. Ferro, D. E. Losada, & H. Müller (Eds.), *Working notes of CLEF 2019 - conference and labs of the evaluation forum, lugano, switzerland, september 9-12, 2019* (Vol. 2380). CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-2380/paper_204.pdf

Martinc, M., Škrlj, B., & Pollak, S. (2021). Tnt-kid: Transformer-based neural tagger for keyword identification. *Natural Language Engineering*, 1–40. doi: 10.1017/S1351324921000127

Pelicon, A., Shekhar, R., Martinc, M., Škrlj, B., Purver, M., & Pollak, S. (2021, April). Zero-shot cross-lingual content filtering: Offensive language and hate speech detection. In *Proceedings of the eacl hackashop on news media content analysis and automated report generation* (pp. 30–34). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2021.hackashop-1.5

Potthast, M., Kiesel, J., Reinartz, K., Bevendorff, J., & Stein, B. (2018a, February). *Buzzfeed-webis fake news corpus 2016.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.1239675 doi: 10.5281/zenodo.1239675

Potthast, M., Kiesel, J., Reinartz, K., Bevendorff, J., & Stein, B. (2018b, July). A Stylometric Inquiry into Hyperpartisan and Fake News. In I. Gurevych & Y. Miyao (Eds.), *56th annual meeting of the association for computational linguistics (acl 2018)* (p. 231-240). Retrieved from https://www.aclweb.org/anthology/P18-1022

Shuyo, N. (2010). *Language detection library for java.* Retrieved from http://code.google.com/p/language-detection/

Škrlj, B., Repar, A., & Pollak, S. (2019). Rakun: Rank-based keyword extraction via unsupervised learning and meta vertex aggregation. In C. Martín-Vide, M. Purver, & S. Pollak (Eds.), *Statistical language and speech processing* (pp. 311–323). Cham: Springer International Publishing.

Štihec, J., Pollak, S., & Žnidaršič, M. (2021). Preliminary experimentation with combinations and extensions of forward-looking sentence detection wordlists. In *Proceedings of the 3rd financial narrative processing workshop* (pp. 26–30). Retrieved from https://aclanthology.org/2021.fnp-1.4

Appendix A: Preliminary experimentation with combinations and extensions of forward-looking sentence detection wordlists

Preliminary experimentation with combinations and extensions of forward-looking sentence detection wordlists

Jan Štihec
University of Ljubljana
Ljubljana, Slovenia
stihec.jan@gmail.com

Senja Pollak
Jožef Stefan Institute
Ljubljana, Slovenia
senja.pollak@ijs.si

Martin Žnidaršič
Jožef Stefan Institute
Ljubljana, Slovenia
martin.znidarsic@ijs.si

Abstract

Forward-looking sentences are often a subject of studies of financial texts. Detection of such sentences is usually performed with wordlists of inclusive and exclusive keywords that are used as indicators of the forward-looking nature of the sentences at hand. In this paper we describe our assessment of potential improvements of forward-looking sentence detection wordlists by combining them together and by extending them with neighboring words in word-vector representations. Our current results indicate that simple combinations and straightforward extensions of wordlists with vector-space representation neighbors might not be suitable for FLS detection without further methodological improvements.

1 Introduction

Many studies of financial texts focus specifically on the contents of the forward-looking sentences (FLS). Detection of such sentences is then either a part of the methodological approach of a study or even one of the main aims of research.

Approaches to detection of these sentences usually employ lists of keywords, which are used as indicators whether a given sentence tends to be forward-looking or not. Keyword lists usually consist of: words that imply the future (e.g. "future"), future years numbers, conjugations of verbs that imply the future (e.g. "we intend") and combinations of certain adjectives and time indicators (e.g. "next year"). Some approaches also use lists of exclusive words, which are used to exclude a sentence that contains them from the forward-looking sentences identification process. Exclusive keywords are not always correlated with a nature of the sentence not being forward-looking, but might only indicate that a sentence containing them should not be analyzed in a specific study. It might for example contain keywords that are indicative for the parts of text, which aren't relevant for the study at hand.

The aim of our work is to study various wordlists for forward-looking sentence detection that appear in the literature, assess their combined use and experiment with wordlist extensions that are based on vector representation distances (similar to related work in terminology extraction, see e.g. (Pollak et al., 2019; Vintar et al., 2020)).

In this paper we report preliminary results of four wordlists, their combination and one wordlist's vector-space based extension on two manually labeled datasets. The current results indicate that the addition of exclusive wordlists might not always improve the results, which stands also for merging of the wordlists. The extension of the wordlists with word vector neighbors increases the amount of detected forward-looking-sentences, but it also increases the amount of sentences that are wrongly classified as FLS. With the current approach, this issue could not be alleviated with a similar extension of the corresponding exclusive wordlist.

2 Related work

Future-oriented information is recognized as very relevant to investors and is the subject of varius studies (Mio et al., 2020). Some studies rely on manual collection and analysis of FLS, while others employ automatic procedures that are mostly based on a number of widely used FLS wordlists. Each of these two approaches has its benefits and drawbacks, but we are interested in the latter one and the impact of the wordlists that are used for such purposes.

We identified four wordlists which are proposed in works that are commonly cited with regards to FLS identicifation and provide complete wordlists. Chronologically ordered the first one is the work by Li (2010), which is focused on information content and tone of FLS sentences. Next is the one by Athanasakou and Hussainey (2014) which is aimed at the assessment of the frequency of such statements and its relations with financial indicators.

The work of Muslu et al. (2015) studies the relation among FLS quantity and the firms' information environments. It suggests also use of word combination patterns, so the FLS wordlist that corresponds to this approach is relatively extensive. Tao et al. (2018) study the relationships among FLS features and IPO valuation. They use a wordlist based FLS detection approach (similar to the one by Muslu et al. (2015), but with additional consideration of sentence structure) in the stage of data preparation for machine-learning of a neural network based FLS classifier. All the listed studies provide a list of FLS inclusive keywords and all with the exception of Athanasakou and Hussainey (2014) also provide a list of FLS exclusive keywords.

3 Methodology

The approach that we used for the study described in this paper consists of: (I) selection of relevant wordlist-based approaches to FLS detection, (II) preparation of the data for testing and learning, and (III) design and running of the experiments.

We selected wordlists from four works (Li, 2010; Athanasakou and Hussainey, 2014; Muslu et al., 2015; Tao et al., 2018), which are often cited with regards to FLS detection and also provide the wordlists and detailed explanation of their FLS detection processes. We denote these wordlists as wl-Li, wl-At, wl-Mu and wl-Ta respectively. The data that was used for assessments and for learning the vector representations (also referred to as embeddings) in our experiments is described in detail in Section 3.1. For efficient experimentation with the selected wordlists we implemented a general wordlist-based labeling tool in python. Section 3.2 is dedicated to description of the methodological details of experiments.

3.1 Data

For the assessments of FLS detection approaches we used the sentences that were selected at random from recent (since 2017) annual reports of randomly selected FTSE 350 index constituents and were annotated as forward-looking/non-forward-looking by two human annotators. As the data was annotated by two annotators who worked on separate (not overlapping) groups of sentences, we treat this data as two datasets of 467 and 459 annotated sentences respectively and we denote them as D_1 and D_2. There are 260 FLS and 207 non-FLS sentences in D_1, while D_2 contains 122 FLS and 337 non-FLS sentences.

Table 1: Size of the used wordlists in terms of the amount of keywords.

	inclusive	exclusive
wl-Li	17	31
wl-At	45	/
wl-Mu	332	6
wl-Ta	373	6

Data was necessary also in the approach for extending wordlists, where it was used for learning vector space representations of words. Annotations are not needed for this purpose, but the data should be from the same domain as the task in which the vector representations are to be employed. We used a corpus of 604 periodic (10-Q and 10-K) reports. Specifically, it consisted of the 2018 Q4 reports from the Stage One 10-X Parse Data collection (from file `10-X_C_2016-2018.zip`) of the well known Notre Dame Software Repository for Accounting and Finance that was established by Loughran and McDonald (2016).

3.2 Experimental setup

In our experiments we used each individual selected wordlist and a merged wordlist that is denoted as wl-all and contains a set of all the words appearing in any of the wordlists. The wordlists were used for labelling the sentences as FLS or non-FLS. The results were calcualted separately for each of the two datasets.

With the exception of the approach by Athanasakou and Hussainey (2014), all the selected approaches provide an inclusive and an exclusive wordlist. First, we used only inclusive wordlists with a straighforward classification approach: the sentences that contained any word from a given inclusive list were classified as FLS. In the next series of experiments we used also all the corresponding exclusive wordlists in the sense that any sentence classified as FLS was re-classified into non-FLS, if it contained any word from the given list of exclusive words.

Note that our use of the wordlists is not completely comparable with most of the related works, from which the wordlists originate, as they were focused on specific sections of financial reports and some of the FLS detection approaches additionally considered numeric indications of future years or, in case of the approach by Tao et al. (2018), the

Table 2: Accuracy (acc) and recall (rec) of FLS classification with inclusive wordlists only.

	acc D_1	rec D_1	acc D_2	rec D_2
wl-Li	0.67	0.60	0.68	0.70
wl-At	0.68	0.66	0.62	0.74
wl-Mu	0.64	0.45	0.71	0.59
wl-Ta	0.64	0.45	0.71	0.59
wl-all	0.71	0.78	0.60	0.87

use of wordlists represented only a part of the FLS detection approach.

The last series of experiments, assessment of the effect of embeddings-based extensions of wordlists, was done only with one original wordlist - the one proposed by Li (2010). Again, both only the inclusive and the inclusive/exclusive options were experimented with. The word vector representations were learned with the fastText approach (Bojanowski et al., 2016) in the ClowdFlows3[1] prototype online tool for data analysis (parameters for learning the fastText model and neighbors selection: *vector size*=20, *context window size*=5, *minimal word occurences*=5, *distance threshold*=0.9). For each of the words in the original wordlist, the original word and five of the neighboring words from the vector space were included in the extended wordlist. The word neighbors were post-processed as follows: (I) any punctuation character at the start or the end of the word was removed, (II) any words that are considered English stop-words by the NLTK language toolkit[2] were removed.

The exclusive wordlist from Li (2010) includes also some bi-grams that are combinations of words: 'expected','anticipated','forecasted','projected','believed' that are preceded with each of the following auxiliary verbs: 'was' , 'were', 'had' and 'had been'. To obtain the corresponding embedding-based neighbors of these terms, we first calculated the neighbors of the words without the auxiliary verbs and then added all the combinations with auxiliary verbs to all the resulting word neighbors.

The resulting extended wordlists are provided in Appendix A.

4 Results and findings

Results of the assessment for inclusive wordlists are presented in Table 2 in terms of accuracy and recall of the FLS class. The recall might be more of interest if the aim of FLS detection is to analyse FLS contents or pre-filtering. For estimation of the amount of FLS the more relevant measure is accuracy, but it needs to be considered carefully in case of unbalanced datasets such as D_1 and D_2. From Table 2 we can see that on D_1 the best individual wordlist results are obtained with wl-At and that the merged wordlist yields better results as any of the individual approaches in terms of both performance measures. This is not the case on D_2, which has more non-FLS sentences. On D_2 these two approaches are better in terms of recall, but worse than others in terms of accuracy.

Table 3: Accuracy (acc) and recall (rec) of FLS classification with inclusive and exclusive wordlists.

	acc D_1	rec D_1	acc D_2	rec D_2
wl-Li	0.67	0.60	0.68	0.70
wl-At	0.68	0.66	0.62	0.74
wl-Mu	0.63	0.41	0.71	0.51
wl-Ta	0.63	0.40	0.71	0.51
wl-all	0.65	0.58	0.63	0.65

Table 4: Accuracy of FLS detection with embeddings-based extensions of the wordlists by Li (2010). Use of extension is denoted by e(), in stands for the use of the inclusive and ex for the use of the exclusive list.

	acc D_1	rec D_1	acc D_2	rec D_2
in	0.67	0.60	0.68	0.70
e(in)	0.68	0.69	0.59	0.78
e(in) ex	0.68	0.69	0.59	0.78
e(in) e(ex)	0.63	0.59	0.60	0.64

Addition of excluding wordlists into consideration slightly reduced all the recalls, with profound effect mostly in case of the merged wordlist. In such a setting, the combined wordlist did not outperform individual ones on any of the two datasets as it for example performs worse than wl-Li with respect to both measures on both datasets.

What we can draw from the first two experiments is that a combination of individual wordlists is not necessarily beneficial, particularly not for the case of considering also exclusive keywords.

Experimental assessment of the embeddings-based extensions of a wordlist are presented in Table 4. The extended inclusive wordlist improves recall, but in D_2 at the expense of accuracy. In comparison with the wordlist extension approach of merging the wordlist with other proposed ones, the

[1] ClowdFlows3 homepage: https://cf3.ijs.si/ The used workflow is available at: https://cf3.ijs.si/workflow/223

[2] https://www.nltk.org/

		Predicted	
		FLS	NON FLS
Actual	FLS	157	103
	NON FLS	50	157

		Predicted	
		FLS	NON FLS
Actual	FLS	86	36
	NON FLS	113	224

Figure 1: Contingency matrix for original inclusive wl-Li on D_1 (left) and D_2 (right).

extension with embeddings performs worse than the merged wordlist for both measures on both datasets.

		Predicted	
		FLS	NON FLS
Actual	FLS	180	80
	NON FLS	70	137

		Predicted	
		FLS	NON FLS
Actual	FLS	95	27
	NON FLS	163	174

Figure 2: Contingency matrix for extended inclusive wl-Li on D_1 (left) and D_2 (right).

Consideration of exclusive keywords was expected to compensate for some of the accuracy lost on D_2 due to potentially too wide reach of the inclusive keyword extensions, but consideration of the original exclusive keywords did not have an effect on results (a single sentence was classified differently in D_1), while use of an embedding-extended exclusive wordlist caused more non-FLS sentences to be correctly classified and vice-versa for the FLS (for details see Figures 1 to 3). This caused slight changes in accuracy in line with the class distributions of the two datasets. Most importantly, overall the approach with both the extended inclusive and extended exclusive wordlist in all aspects performed worse than the approach with original state of these two wordlists (for comparison see Table 3).

Our study is preliminary and we intend to conduct more experiments on larger datasets, but the current results indicate that straightforward extensions of wordlists with vector-space representation neighbors might not be suitable for FLS detection. In most experimental settings this holds also for extensions of wordlists by merging them together, although by a lesser extent.

This does not mean that such approaches cannot improve FLS detection, but it indicates that it might be necessary to go beyond a simple automated word vector neighbor extension and that such methodological improvements would be sensible already before further experimentation.

		Predicted	
		FLS	NON FLS
Actual	FLS	153	107
	NON FLS	65	142

		Predicted	
		FLS	NON FLS
Actual	FLS	78	44
	NON FLS	140	197

Figure 3: Contingency matrix for extended inclusive and extended exclusive wl-Li on D_1 (left) and D_2 (right).

Acknowledgements

This paper is supported by the project Quantitative and qualitative analysis of the unregulated corporate financial reporting (No. J5-2554), which was financially supported by the Slovenian Research Agency. The paper was supported also by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 825153, project EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media). The authors acknowledge also the financial support from the Slovenian Research Agency for research core funding for the programme Knowledge Technologies (No. P2-0103). For access to the dataset of labeled forward looking sentences we thank the Faculty of Economics of the University of Ljubljana.

References

Vasiliki Athanasakou and Khaled Hussainey. 2014. The perceived credibility of forward-looking performance disclosures. *Accounting and business research*, 44(3):227–259.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Feng Li. 2010. The information content of forward-looking statements in corporate filings—a naïve bayesian machine learning approach. *Journal of Accounting Research*, 48(5):1049–1102.

Tim Loughran and Bill McDonald. 2016. Textual analysis in accounting and finance: A survey. *Journal of Accounting Research*, 54(4):1187–1230.

Chiara Mio, Pier Luigi Marchini, and Alice Medioli. 2020. Forward-looking information in integrated reports: Insights from "best in class". *Corporate Social Responsibility and Environmental Management*, 27(5):2212–2224.

Volkan Muslu, Suresh Radhakrishnan, KR Subramanyam, and Dongkuk Lim. 2015. Forward-looking md&a disclosures and the information environment. *Management Science*, 61(5):931–948.

Senja Pollak, Andraž Repar, Matej Martinc, and Podpečan Vid. 2019. Karst exploration: extracting terms and definitions from karst domain corpus. In *Proceedings of eLex19*, pages 934–956, Sintra, Portugal.

Jie Tao, Amit V Deokar, and Ashutosh Deshmukh. 2018. Analysing forward-looking statements in initial public offering prospectuses: a text analytics approach. *Journal of Business Analytics*, 1(1):54–70.

Špela Vintar, Larisa Grčić Simeunović, Matej Martinc, Senja Pollak, and Uroš Stepišnik. 2020. Mining semantic relations from comparable corpora through intersections of word embeddings. In *Proceedings of the 13th Workshop on Building and Using Comparable Corpora*, pages 29–34, Marseille, France. European Language Resources Association.

A Embedding-based extensions

Extension of the inclusive part of wl-Li. The words from the original wordlist are in bold, followed by up to five extensions (less, if removed as stop-words or duplicates with extensions of preceeding original words):
will accordingly furthermore **should** relied regarded ultimate context **can** frequently unreliable predicate producibility problem **could** harm harmed adverse **may** even substantial us **might** materialize pursued occur difficult **expect** expand effectively continue expansion **anticipate** profitable **believe** proactively history believes regularly **plan** plans sponsors sponsor **hope** hopes success perspectives identify teamwork **intend** intends **seek** seeking stop decide **project** progress feasibility projects predevelopment **forecast** quarter-to-quarter profitability forecasting forecasts **objective** objectively objectivity maximize **goal** toward targeting driving striving excellence

Extension of the exclusive part of wl-Li. The words from the original wordlist are in bold, followed by up to five extensions (less, if removed as stop-words or duplicates with extensions of preceeding original words):
undersigned, undersigned's, duly, thereunto, countersigned, **herein**, reference, referenced, **hereinafter**, hereinabove, mean, indicated, **hereof**, TAA, **hereon**, henceforth, **hereto**, confirms, **theretofore**, grantor, asserted, party, deemed, obligated, **therein**, documents, **thereof**, therefor, **thereon**, **expected**, differences, future, reversals, different, **was expected**, was differences, was future, was reversals, was different, **were expected**, were differences, were future, were reversals, were different, **had expected**, had differences, had future, had reversals, had different, **had been expected**, had been differences, had been future, had been reversals, had been different, **anticipated**, negative, forecast, unanticipated, results, **was anticipated**, was negative, was forecast, was Unanticipated, was results, **were anticipated**, were negative, were forecast, were Unanticipated, were results, **had anticipated**, had negative, had forecast, had Unanticipated, had results, **had been anticipated**, had been negative, had been forecast, had been Unanticipated, had been results, **forecasted**, magnified, imbalance, movements, variability, fluctuation, **was forecasted**, was magnified, was imbalance, was movements, was variability, was fluctuation, **were forecasted**, were magnified, were imbalance, were movements, were variability, were fluctuation, **had forecasted**, had magnified, had imbalance, had movements, had variability, had fluctuation, **had been forecasted**, had been magnified, had been imbalance, had been movements, had been variability, had been fluctuation, **projected**, projecting, **was projected**, was projecting, **were projected**, were projecting, **had projected**, had projecting, **had been projected**, had been projecting, **believed**, likelihood, verified, mistaken, livelihood, **was believed**, was likelihood, was verified, was mistaken, was livelihood, **were believed**, were likelihood, were verified, were mistaken, were livelihood, **had believed**, had likelihood, had verified, had mistaken, had livelihood, **had been believed**, had been likelihood, had been verified, had been mistaken, had been livelihood, **shall**, hereunder,