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: 36 months

D6.4: Platform requirements documentation and platform design (T6.2)

Executive summary
The objectives of WP6 are to determine the needs and challenges of the news media industry and to address them with technologically advanced software services and user friendly tools. These services and tools, built throughout the project in WP1–WP5 will be offered through the EMBEDDIA Media Assistant Toolkit. Since Media Assistant Toolkit is built using the TEXTA Toolkit (TTK) REST API, this report gives an overview of the core and architecture of TTK as well as how it will be integrated with the EMBEDDIA Media Assistant Toolkit.

Partner in charge: TEXTA

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:	**D6.4**
Deliverable full title:	**Platform requirements documentation and platform design**
Deliverable short title:	**Media Assistant platform documentation**
Document identifier:	**EMBEDDIA-D64-MediaAssistantPlatformDocumentation-T62-submitted**
Lead partner short name:	**TEXTA**
Report version:	**submitted**
Report submission date:	**31/12/2019**
Dissemination level:	**PU**
Nature:	**R = Report**
Lead author(s):	**Raul Sirel (TEXTA)**
Co-author(s):	**Kristiina Vaik (TEXTA), Silver Traat (TEXTA)**
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
25/10/2019	v1.0	Raul Sirel (TEXTA)	First draft.
27/10/2019	v1.1	Kristiina Vaik (TEXTA)	Draft report template.
21/11/2019	v1.2	Raul Sirel (TEXTA)	Adding content in Section 3.
26/11/2019	v1.3	Kristiina Vaik (TEXTA)	Editing, adding content.
04/12/2019	v1.4	Matthew Purver (QMUL)	First internal review.
09/12/2019	v1.5	Saturnino Luz (UEDIN)	First internal review.
12/12/2019	v1.6	Kristiina Vaik (TEXTA)	Consolidation based on the internal reviews.
17/12/2019	v1.7	Matthew Purver (QMUL)	Second internal review.
12/12/2019	v1.8	Kristiina Vaik (TEXTA)	Consolidation based on the 2nd internal review.
20/12/2019	final	Nada Lavrač (JSI)	Quality control.
23/12/2019	submitted	Tina Anžič	Report submitted.

Table of Contents

List of abbreviations

API	Application Programming Interface
EMA	EMBEDDIA Media Assistant
ExM	Ekspress Meedia
GUI	Graphical user interface
NLP	Natural Language Processing
STT	Finnish News Agency, *Suomen Tietotoimisto*
STY	Styria Media Services
TTK	TEXTA Toolkit

1 Introduction

The overall objectives of WP6 are to determine the needs and challenges of the news media industry, and address them with technologically advanced software services and user-friendly tools. One of the objectives of WP6 is to define the end-user platform requirements, integrate the tools and resources and build the EMBEDDIA Media Assistant (EMA) Toolkit.

This present deliverable is one of the deliverables of Task T6.2 of WP6 describing the requirements and design of the platform which will integrate all the software services, tools and resources developed and gathered within WP1–WP5.

The report is organized as follows. Section 2 gives an overview of the architecture and core concepts of the TEXTA Toolkit, an existing platform which will be the foundation of the EMBEDDIA Media Assistant Toolkit. It is important to note that the work on the TEXTA Toolkit back-end started in M7 while this deliverable is presented in M12. As a consequence, most of what is described in Section 2.1 has already been implemented, while cross-lingual capabilities will be added once the models in WP1–WP5 are made available. Section 3 describes the interaction between the TEXTA Toolkit and the EMBEDDIA Media Assistant Toolkit. It introduces how the user needs will be supported through the functional needs of the Toolkit and reports the general architecture of the communication flow between the two platforms. Finally, Section 4 presents the conclusions and future work directions.

2 The architecture of the TEXTA Toolkit

This section introduces the TEXTA Toolkit which will be the backbone for the EMBEDDIA Media Assistant Toolkit. The first part of this section gives a general overview of the Toolkit's ecosystem. In the second part we introduce some core concepts and terms which need some elaboration to get a better understanding of the TEXTA universe.

2.1 An overview of the TEXTA Toolkit

The EMBEDDIA Media Assistant (EMA) Toolkit is built using the TEXTA Toolkit (TTK) 2.x REST API, which constitutes the backbone of the EMA Toolkit. It is the aim of the TTK back-end to bring together various text mining and NLP services and models into a single ecosystem. This is achieved by developing a common data model using Django.[1] The TTK API has been developed using the Django REST framework[2] which allows the users to perform various tasks via the API, including:

- train embeddings to compute word and phrase similarities;
- train binary and multi-class text classifiers for content tagging;
- train multi-label text classifiers for topic/tag prediction;
- create lexicons semi-automatically by finding similar words and phrases from trained embeddings;
- cluster words and phrases using their distributional similarity in embeddings;
- create projects and share trained resources with other users.

The EMA Toolkit, which is built on top of TTK to fully harness its data model and existing modules, is a collection of tools and services integrated into the TTK ecosystem and powered by a front-end developed in Angular.[3] TTK uses technologies including Elasticsearch, Redis, Apache TIKA, etc. for extracting and managing data and its pipelines. For statistical modelling, TTK applies Gensim, Scikit-learn, PyTorch, etc. A more detailed overview of the technologies involved is shown in Figure 1.

[1] https://docs.djangoproject.com/en/2.2/topics/db/models/
[2] https://www.django-rest-framework.org
[3] https://angular.io

Figure 1: Technology stack behind the TEXTA Toolkit.

2.2 Core concepts of the TEXTA Toolkit

Project is the main unit of access and management for datasets and resources (embeddings, text classifiers, etc.). A project is defined by its description, list of Elasticsearch indices related to the project (this is where the data is!), and a list of users who can access the project and its resources. All resources in TTK have to belong to a project and by adding or removing users, one can manage their access to the project.

For text processing TTK allows to build several different statistical models. The training process is initiated via TTK API, which results in creating the object in the TTK data model and also starting the asynchronous training task. **Tasks** are data objects for keeping track of the statistical models' training progress.

One of the most central components in TTK are **Searches**, which are used to define subsets from the data and use the search outcome to train text classification models and/or perform various aggregations over the data. Searches are managed via GUI and can contain one or more constraints on feature values, e.g., strings and dates. Documents matching the search criteria can be used in various actions / functionalities, e.g., extract relevant keywords from these documents, explore and summarize the data, or use these documents as training data for building a new classification model.

A core concept of the EMBEDDIA project are (cross-lingual) **word embeddings**, which by itself are representations of words in numerical form while preserving semantic relations between words and are used as input for machine learning or deep neural network models. Consequently, we have introduced this core concept also in TTK, similarly called **Embedding**. This concept is used to describe the distributional properties of words, which enables to compute the semantic similarity between different words and phrases. In TTK, word embeddings are used for extending the search results by finding synonymous keywords and by helping to build different lexicons of words similar to each other. Currently TTK only supports word2vec embeddings (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013), but for building different classification models the goal is to incorporate state-of-the-art embeddings, e.g., BERT (Devlin, Chang, Lee, & Toutanova, 2018), ELMo (Peters et al., 2018). Furthermore, TTK will also employ cross-lingual embedding models developed during the EMBEDDIA project to support multilingual

classification tasks.

In TEXTA terminology, **taggers** are monolingual binary text classifiers used to predict tags for documents, e.g., whether a user generated comment about Brexit is negative. Taggers are trained using a subset of documents defined by the *search* or a raw Elasticsearch query which essentially is a big JSON object. The subset of documents defines the *positive* set of examples (the documents about the same topic being tagged), whilst *negative* examples will be selected randomly and automatically so that these examples would not be duplicates of the positive examples. TTK taggers are trained using the *scikit-learn* pipelines including models like logistic regression, SVM etc. TTK automatically splits the training data into train and test data (80–20 ratio) and applies grid search combined with k-fold cross-validation to identify the best hyper-parameters. TTK also uses the SVM model for feature selection in the trained tagger in order to remove insignificant features. For features, taggers use both word-based and character-based n-grams.

Tagger Group is an extension to binary taggers for supporting monolingual multi-label classification. As its name suggests, a Tagger Group incorporates multiple (binary) taggers, which are executed in parallel to produce a list of tags / categories to the user. The Tagger Group has been successfully tested with over 6000 binary models while prediction times are usually less than 1 second. To achieve this, TTK employs a hybrid approach for multi-label classification. It uses unsupervised machine learning (document vectors as features) to limit the number of binary models used for the prediction. In such scenario, the input document will be compared to the training data in order to determine the most probable models to produce valid tags / categories for that input document. The two-stage (or hybrid) tagging approach has been developed and fine-tuned in TEXTA's previous commercial projects to predict topics and entities to newspaper articles and books. The algorithm and its performance have not yet been published.

While Taggers and Tagger Groups use classical machine learning to produce binary classification models, TTK can also handle deep neural models for binary and multi-label classification. As the models are all built by using PyTorch[4], in TTK the component is called **Torch Tagger**. It allows the user to use several state-of-the-art classification models, including fastText (Joulin, Grave, Bojanowski, & Mikolov, 2017), TextRNN (Zhou et al., 2016) using bi-directional LSTM networks, and RCNN (Lai, Xu, Liu, & Zhao, 2015) using recurrent convolutional neural nets. Torch Tagger models also have the option to use pre-trained vector models in the embedding layer. For creating data processing pipelines, Torch Tagger uses the torchtext package.[5] Torch Tagger has been validated on Ekspress Meedia (ExM) commentary dataset (approx. ~200 000 examples) in detecting monolingual toxic comments, achieving accuracy and F1-score of 96%.

In TTK, all deep neural models have been implemented using PyTorch following a common functional design pattern[6], which makes the introduction of new models to the TTK fairly straightforward. The current version of the Torch Tagger in TTK is deployed using the models and configurations described in the Github repository.

3 From TEXTA Toolkit to EMBEDDIA Media Assistant Toolkit

This section describes the interaction between the TKK and EMA Toolkits. The first part concentrates on the general architecture of the communication between the TKK and EMA Toolkits, mainly about the back-end / front-end and data management services. The second part describes how the user needs identified in deliverable D6.3 will be supported through the TTK basic functional requirements.

[4] `https://pytorch.org/`
[5] `https://torchtext.readthedocs.io/en/latest/`
[6] An example of the pattern used can be found here: `https://github.com/AnubhavGupta3377/Text-Classification-Models-Pytorch/`

3.1 General architecture

The general design of the EMA Toolkit is modular and uses a service-oriented style for architecture. All services in the EMA Toolkit use RESTful API to communicate with each other and are packaged as Docker images.

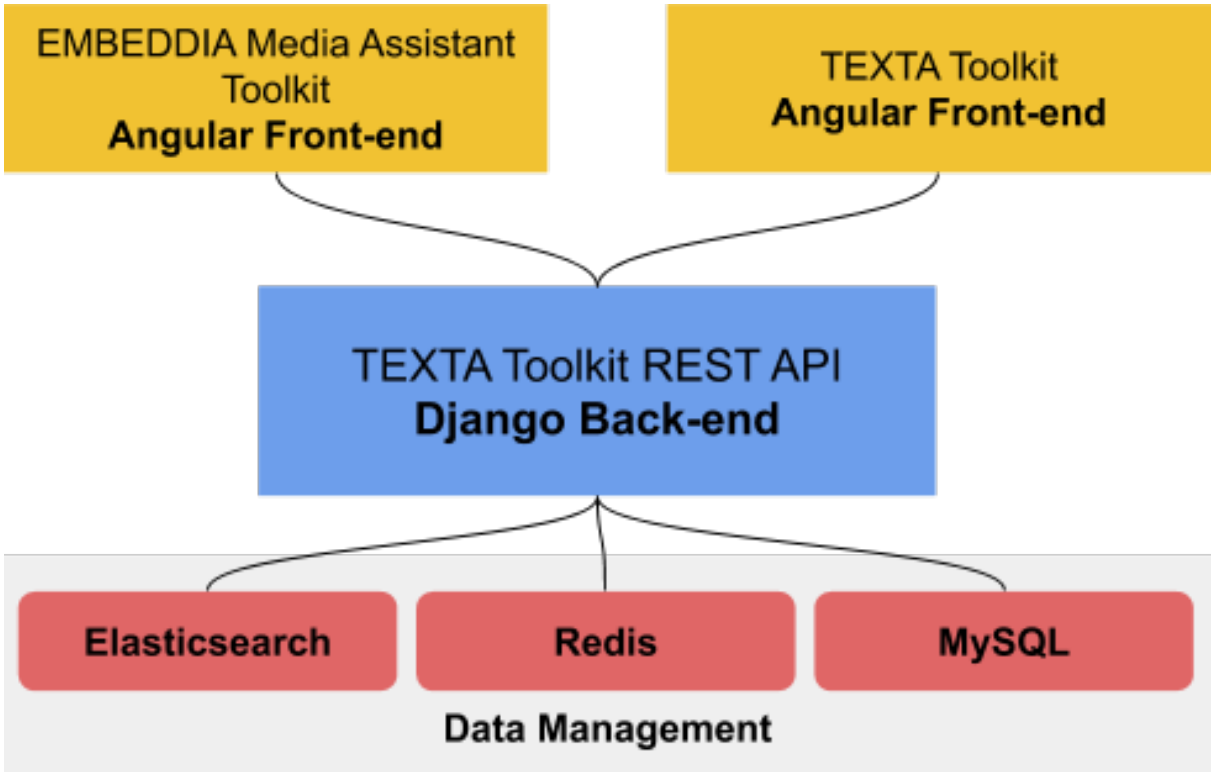

Figure 2: Services constituting the joint ecosystem between the TEXTA Toolkit and the EMBEDDIA Media Assistant Toolkit.

The EMA Toolkit will comprise of 3 main services (see Figure 2):

1. TTK REST API (**Django Back-end**) to support the management of mono- and cross-lingual taggers and embeddings (including the import and usage of pre-trained cross-lingual embeddings);
2. TTK front-end (**Angular Front-end**) for generating searches; importing, making subsets, exploring and aggregating data; training and validating mono- and cross-lingual classifiers and monolingual embeddings;
3. EMA front-end (**Angular Front-end**) summarising and visualising data for media partners and end-users in real-time.

Additionally, the EMA Toolkit will be supported by several data management services:

- **Elasticsearch** for storing the documents used for training and tagging; Elasticsearch API is also usable for importing documents to the Toolkit;
- **Redis** for message brokers in long-running data processing pipelines;
- **MySQL** for storing the data model containing information about the projects, tasks, embeddings and taggers etc.

The underlying logic for importing data, using embeddings and classification models built during the project in the EMA Toolkit is displayed in Figure 3. The EMA Toolkit and TTK both support importing

pre-trained models as well as training new custom models inside the Toolkit. All pre-trained models will be imported to the Toolkit by using REST APIs. Both imported as well as custom trained models can be used as an input in the embedding layers of Torch Tagger classification models. While monolingual embeddings provide faster learning rates (less epochs needed) and general knowledge about language with little training data, cross-lingual models support directly applying classifiers trained in one language to other target languages.

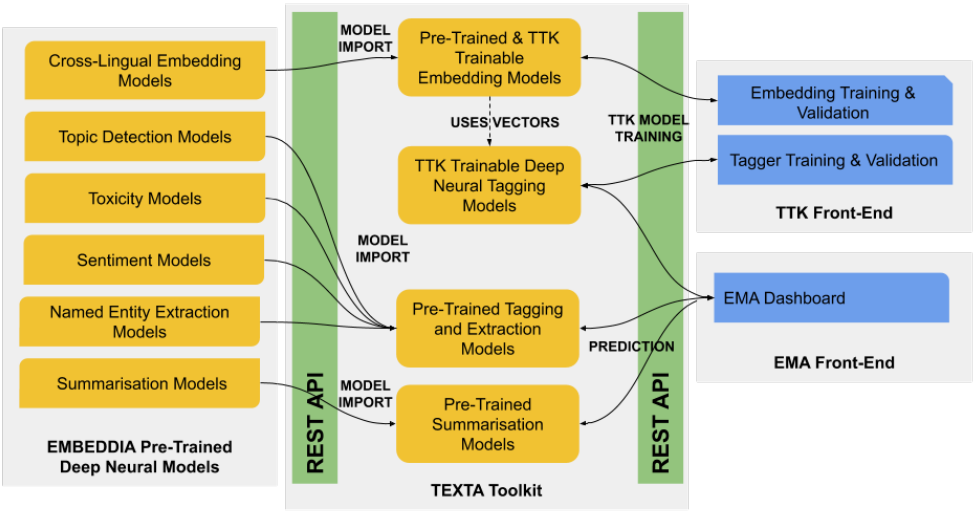

Figure 3: Visualisation of theb EMA Toolkit.

In contrast to currently existing solution of training simple embeddings (word2vec) in TTK, **cross-lingual embeddings will not be trained inside the TTK**. Instead (cross-lingual embedding) models from WP1 will run as separate services: they will be stored in Docker, which makes it easy to update and run them on the TTK server and communicate with the EMA Toolkit via APIs (see left side of Figure 3). Similar to cross-lingual embeddings, **EMBEDDIA classifier models from WP2–WP5 are also not trained inside the TTK**, instead these pre-trained models will also run as separate services as mentioned above (also see left side of Figure 3). These pre-trained cross-lingual deep learning models will be used in the EMA Toolkit via TTK API for different cross-lingual classification tasks (e.g., named entity extraction, topic and sentiment detection, comment moderation). However, a user can also train a **custom cross-lingual classification model inside the TTK** (see right side of Figure 3) by using the imported cross-lingual embeddings. The architecture of TTK treats cross-lingual embeddings as any other pre-trained embeddings which can be used in combination with Toolkit's different classification models.

3.2 Functional requirements

The main purpose and essence of the EMA Toolkit is all about the end-users and media partners. In order to develop methods and tools which will be implemented within the EMA Toolkit we need to derive and prioritise actual user needs of the news industry for the EMBEDDIA project. These actual user needs were explored in a project workshop which were summarized in *D6.3 User needs and challenges for news media industry (T6.1)*. The workshop involved discussing what kind of digital tools are necessary in newsrooms which culminated with discussing the promising applications and features per each task within every work package. In this section, we define functional requirements for the EMA Toolkit to ensure that it supports the main user needs identified.

Work package 3 (Cross-lingual Technologies for User-Generated Content) concentrates on different aspects of user comments (context & opinion analysis, comment filtering, report generation from multilingual comments). Media partners in general are interested in automating tasks which until this project have been done manually. They are especially interested in detecting comments which should be blocked or referred to human moderators. This also comes with a need to: a) give reasons of why a certain comment was blocked; b) who are the likely users to generate comments needing blocking or moderation; c) mine opinion, e.g. which comments were positive, constructive; d) mine topics and finally e) visualization. In order to support these needs, for data storage EMA Toolkit will use Elasticsearch via TTK which is also suitable for browsing, exploring and aggregating over the data. Applying cross-lingual multi-label classifiers on these comments via API will tag comments based on content / sentiment / topic. Hence, with the help of these tags EMA Toolkit will aggregate over the data, make generalized reports and in the end make the data alive and visualize it on the EMA Toolkit's Dashboard in a mind map style.

Work package 4 (Cross Lingual Content Analysis) concentrates on developing tools for analysis of news content across languages (multilingual news linking, news summarisation & visualisation; identifying viewpoints & sentiment in news reporting). Media partners were interested in linking news with similar topics serving as background or new (recommended) knowledge for the reader. So far they have been doing this news linking by topic manually. Media partners stated that news summarisation and visualisation is not one of their priorities. However, when connecting summarisation & visualisation with news linking it would be beneficial to see all related articles with a certain topic in different languages. Also, it was stated that news archives would be useful for writing new news articles. Media partners were also interested in a *neutrality barometer* to see whether the article is neutral and not biased. A very important aspect the media partners mentioned was to monitor neighbouring countries see what kind of news emerge there. When it comes to the end-user needs these needs are already supported by the TTK functional requirements discussed above. As mentioned above Elasticsearch is very fit to function as a diachronic archive where a journalist can browse old articles, search certain topics / keywords etc and aggregate the search outcome over different data fields (e.g. journalist seeks from the archive to know how has the European migrant crisis being portrayed over the years). This also applies for linking different news articles within the the country and between countries. Knowing the topic(s) or categories of the news articles (in different countries) and synchronously analysing the most emerging topics within a country with the help of Elasticsearch it is possible to visualize different emerging topics from different countries on the EMA Toolkit's Dashboard. This also applies for not even knowing the article categories or topics, in this case categorization classifiers will be applied before the synchronous analysis. The same principle also applies for connecting a certain article to background articles. Since this work package's tasks are building tools and models for detecting viewpoints and sentiment, it very realistic to build a tool for detecting whether a news story is biased or not, neutral or not. To meet this user need we will apply cross-lingual sentiment or viewpoint multi-label classifiers via API on article(s) and the end-user will see the visualisation of it on the EMA Toolkit's Dashboard.

Work package 5 (Multilingual Text Generation) concentrates on designing and developing news automation systems that are transferable across languages and domains (text generation from structured data, multilingual storytelling and content generation, creative language use for multilingual news and headline generation). The media partners (especially important to STT) are very keen on generating texts

styled for different genres. This workpackage will not be serviced from the TEXTA Toolkit's back-end side, instead the EMA Toolkit will communicate directly via REST API with the service provided by the project partner University of Helsinki.

In summary, in order to support the user needs identified in deliverable D6.3, the functional requirements for the EMA Toolkit are as follows:

- import data into Elasticsearch using an API;
- use pre-trained cross-lingual word embeddings in classification tasks;
- train and validate cross-lingual classification models;
- apply cross-lingual classifiers for tag / category prediction via API (e.g. topics, comment moderation, sentiment);
- browse, explore, and aggregate data (news articles, commentaries etc.);
- a GUI for real-time data summarisation and visualisation.

4 Conclusions and further work

The aim of this report is to give an overview of the EMBEDDIA Media Assistant Toolkit. In summary it will be built upon the TEXTA Toolkit back-end that unites various data mining, other NLP services and models into one single ecosystem. This ecosystem is achieved by developing a common data model using Django. Since TEXTA Toolkit API has been developed using Django RESTful framework, allowing users to perform tasks via API, therefore the EMBEDDIA Media Assistant Toolkit will also be designed to be modular using service-oriented style. All services in EMBEDDIA Media Assistant Toolkit will use RESTful APIs to communicate with each other and these services are packaged as Docker images.

EMBEDDIA Media Assistant Toolkit will comprise of 3 main services:

1. TEXTA Toolkit REST API to support the management of mono- and cross-lingual taggers and embeddings (including import and usage of pre-trained cross-lingual embeddings);
2. TEXTA Toolkit front-end for performing searches, importing, subsetting, exploring and aggregating data, training and validating mono- and cross-lingual tagging models and monolingual embeddings;
3. EMBEDDIA Media Assistant Toolkit front-end summarising and visualising real-time data for media partners and end-users.

All these services, tools and resources built throughout the project in WP1–WP5 will be made available through the EMBEDDIA Media Assistant Toolkit.

Further work must be done by introducing cross-lingual word embeddings to the TEXTA Toolkit. With cross-lingual word embeddings it is possible to train models for different arbitrary downstream tasks, e.g., text classification, topic / sentiment / hate speech detection, named entity recognition (NER). In order to support requirements set by the EMBEDDIA Media Assistant Toolkit for cross-lingual tasks, the TEXTA Toolkit will be further upgraded to import pre-trained cross-lingual embeddings and classifier models; use pre-trained cross-lingual embeddings developed throughout the EMBEDDIA project for training custom cross-lingual classification models and lastly, use pre-trained PyTorch models for different downstream tasks, e.g., topic and hate speech detection, sentiment prediction.

In addition, further work needs to be done by establishing some additional functional requirements among end-users and media partners (ExM, STY, STT). At the moment TEXTA has done a lot of work on meeting the project needs from the back-end perpective and the next step will be to start discussing the front-end of the EMBEDDIA Media Assistant Toolkit. The EMA Toolkit's Dashboard will be built, once models from WP1 to WP5 are made available.

This task will be conjointly working with Task 6.3 whose objective is to evaluate user experience in using the produced tools and the Media Assistant Toolkit. Both back-end and front-end development will rest on the actual news media needs, therefore the development will follow the agile principles (Beck et al., 2001; Paetsch, Eberlein, & Maurer, 2003) in which the end-users will be involved in the development from the beginning. TEXTA will ensure that not only the developed solutions and applications are based on the actual news media needs, but are constantly tested and evaluated by the industrial end-users as they are iteratively integrated into the EMBEDDIA Media Assistant.

References

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... Thomas, D. (2001). *Manifesto for agile software development.* Retrieved from http://www.agilemanifesto.org/

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Retrieved from http://dx.doi.org/10.18653/v1/e17-2068 doi: 10.18653/v1/e17-2068

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Proceedings of the twenty-ninth aaai conference on artificial intelligence* (pp. 2267–2273). AAAI Press. Retrieved from http://dl.acm.org/citation.cfm?id=2886521.2886636

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Paetsch, F., Eberlein, A., & Maurer, F. (2003). Requirements engineering and agile software development. In *Proceedings of the twelfth international workshop on enabling technologies: Infrastructure for collaborative enterprises* (pp. 308–). Washington, DC, USA: IEEE Computer Society. Retrieved from http://dl.acm.org/citation.cfm?id=938984.939792

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., & Xu, B. (2016, December). Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers* (pp. 3485–3495). Osaka, Japan: The COLING 2016 Organizing Committee. Retrieved from https://www.aclweb.org/anthology/C16-1329