

TransFeatEx: a NLP pipeline for feature extraction

Agustí Gállego¹, Quim Motger¹, Xavier Franch¹ and Jordi Marco²

¹Department of Service and Information System Engineering, Universitat Politècnica de Catalunya

²Department of Computer Science, Universitat Politècnica de Catalunya

Abstract

Mobile app stores provide centralized access to a large data set of mobile app related natural language textual data, including developer's documentation (e.g., descriptions, changelogs) and user-generated data (e.g., user reviews). Motivated by this context, multiple studies have focused on data-driven elicitation processes for the automatic extraction of the set of features exposed by a catalogue of applications and the inferred, extended knowledge that can be derived from this information. Moreover, with the emerging and generalization of large language models, traditional linguistic-based approaches can be significantly improved by the potential of the knowledge embedded in this kind of models. In this paper, we present TransFeatEx, a NLP-based feature extraction pipeline that combines the use of a RoBERTa-based model with the application of consolidated syntactic and semantic techniques. The pipeline is designed as a customizable, standalone service to be used either as a playground, experimentation tool or as a software component to be embedded into a third-party software system for batch-processing large document corpora. An example of a demo plan is showcased here: https://youtu.be/gffyi_i_uTw.

Keywords

feature extraction, natural language processing, large language models, transformer models, mobile apps

1. Introduction


In natural language processing (NLP) and information retrieval, keyword extraction is defined as a text mining process which automatically identifies relevant terms to build condensed representations of text documents [1]. The definition of *relevance* is tailored by the purpose or application of the keyword extraction process, such as text classification [2], sentiment analysis [3] or document clustering [4]. Whether using statistical or machine/deep learning strategies to support these tasks, the availability of a large, representative document corpus is essential to support their design and implementation, as well as to assess their validity and overall quality.

In this sense, app stores have become a popular source of app metadata and app related natural language documents, which can be used as potential sources of descriptors for mobile software applications [5, 6]. As they have become a basic commodity for mobile users world-wide [7], they provide public access to a wide data set of documents, including official developer's documentation and user reviews. Consequently, they have been used as data sources and development context for multiple software components and tools based on keyword extraction tasks, including app classification [1] and app feature extraction [8]. Concerning the latter,

In: A. Ferrari, B. Penzenstadler, I. Hadar, S. Oyedeji, S. Abualhaija, A. Vogelsang, G. Deshpande, A. Rachmann, J. Gulden, A. Wohlgemuth, A. Hess, S. Fricker, R. Guizzardi, J. Horkoff, A. Perini, A. Susi, O. Karras, A. Moreira, F. Dalpiaz, P. Spoletini, D. Amyot. Joint Proceedings of REFSQ-2023 Workshops, Doctoral Symposium, Posters & Tools Track, and Journal Early Feedback Track. Co-located with REFSQ 2023. Barcelona, Catalunya, Spain, April 17, 2023.



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

a comprehensive, accurate representation of the set of features exposed by a set of apps is essential for multiple tasks, e.g. software evolution [5], or app recommender systems [9].

Feature extraction encompasses NLP techniques which include syntactic and/or semantic strategies [8, 9, 6]. However, the emerging of large, pre-trained language models, and more specifically, transformer models (e.g., GPT [10], BERT [11]), has outperformed the state-of-the-art performance of multiple NLP tasks, including keyword extraction. Moreover, these models offer great potential in terms of the linguistic (i.e., syntactic and semantic) knowledge they embed, which can be tailored and integrated into domain-specific feature extraction processes.

In this paper, we present **TransFeatEx**, a feature extraction tool which combines consolidated syntactic and semantic feature extraction techniques with the use of a RoBERTa-based pre-trained model [12] to automatically extract app features from app-related textual documents. The tool is designed as a customizable pipeline to provide researchers and developers with domain-specific tuning capabilities, and it is distributed as a decoupled, standalone web service.

2. Background

In this context, **feature extraction** is defined as a data-driven elicitation of functional requirements from the user perspective for a given software component [13], for which we focus on mobile applications. The vast amount of data items (i.e., natural language documents) generated by world-wide used mobile app repositories is a key motivational factor for mobile app oriented feature extraction research. While this category mainly includes app stores as the most popular source (e.g., Google Play), alternative platforms include external, non-proprietary repositories and search engines (e.g., AlternativeTo¹) indexing multi-source data.

These data sources give access to numerous natural language document from different types. Traditionally, feature extraction has been focused on descriptions and user reviews [8, 6, 5]. Concerning the latter, user-generated data introduces the challenge of processing polarized, biased knowledge, for which the addition of a sentiment analysis filtering component is suggested by multiple state-of-the-art proposals [14]. Beyond these, in the surveyed literature, additional document types like changelogs or summary/short descriptions are generally ignored.

Traditional syntactic and semantic strategies supporting app feature extraction are built upon deterministic (rule-based) or probabilistic/statistic (ML-based) techniques [13]. Syntactic strategies are focused on syntactic Part-of-Speech (POS) and dependency tree patterns recognition [8], while semantic strategies focus on lexical dictionaries and pre-trained models [9]. With the appearance of **transformer models**, proposals like BERT rapidly outdated the performance of previous approaches on traditional NLP linguistic tasks (e.g., POS tagging, dependency parsing, Named Entity Recognition) [11]. Among the variants that followed BERT, one of the most popular is RoBERTa, which outperformed BERT by limiting the scope of pre-training tasks and using a larger data-set [12], among other specifications.

¹<https://alternativeto.net/>

3. Tool description

The main goal of the TransFeatEx tool is to support researchers and developers in the development of projects, tools and processes which require from structured data concerning the set of features exposed by a catalogue of mobile applications. Without the need of annotated data, TransFeatEx leverages the accuracy of pre-trained transformer models to enrich the natural language texts received as input (such as descriptions or user-generated reviews) with linguistic annotations, and then uses such annotations to apply consolidated syntactic and semantic techniques in order to extract features from said texts. TransFeatEx includes customization capabilities to fine-tune feature detection tasks (e.g., POS patterns, syntactic dependency patterns) to allow for domain and context adaptation of the feature extraction process.

3.1. Tool architecture

The TransFeatEx tool has been designed as a standalone, decoupled pipeline deployed in the form of a web service accessible through an API with two basic user requests. The goal of this design is to allow easy integration of the pipeline with third-party software systems, as well as to facilitate playground and experimentation. It consists of a REST controller responsible for receiving requests, and a core NLP layer concerned with the actual text processing and feature extraction. The NLP layer is composed of two sub-pipelines:

- **Transformer-based pipeline:** the tool uses a RoBERTa-based pre-trained instance [12] to process the text and annotate its different elements with syntactic features.
- **Semantic and syntactic analyser:** the enriched textual data resulting from the previous step is analysed to identify potential features based on syntactic and semantic cues.

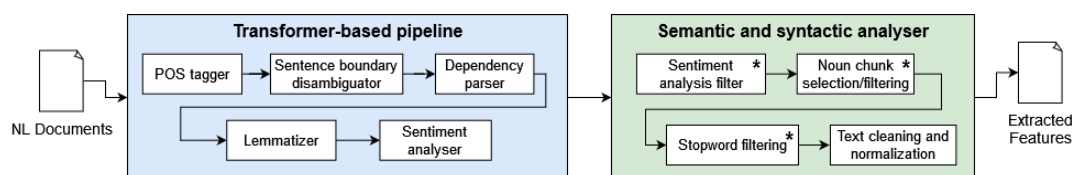


Figure 1: TransFeatEx inner workings. Steps marked with an asterisk (*) are customizable

3.2. Inner workings

When the tool receives a request for processing, the textual data goes through the two sub-pipelines in the NLP layer (Figure 1). The **transformer-based pipeline** consists of several pipes that annotate, process and enrich the textual data with additional syntactic and semantic information. The relevant components are the following:

1. **RoBERTa-based model:** the language model's core component powering subsequent ML-based pipes. We use this model as a pre-trained checkpoint with state-of-the-art accuracy evaluation metrics in syntactic tasks (e.g., POS tagging, dependency parsing)².

²Model's accuracy scores: https://spacy.io/models/en#en_core_web_trf.

2. **PoS Tagger:** it labels each word with its corresponding grammatical category.
3. **Sentence boundary disambiguator:** custom pipe that splits the text in sentences.
4. **Dependency parser:** the model infers the syntactic dependencies between the different words and includes this information in the data.
5. **Lemmatizer:** each word is annotated with its lemma (that is, the base form of the word).
6. **Sentiment analyser:** a sentiment analysis component that uses a simple algorithm which assigns a polarity and subjectivity weight based on predefined word scores.

Results serve as input for the **semantic and syntactic analyser**, where the tool uses the annotated syntactic features of the input text to identify, extract and clean the noun phrases that could potentially be mobile app features. This sub-pipeline is composed of the following:

1. **Sentiment analysis filter:** retrieve the subjectivity and polarity scores of the current textual data and exclude it from processing if said scores don't fall between certain customizable threshold values (default values do not exclude any document).
2. **Noun phrase selection:** select all noun phrases. The tool focuses on the noun phrases³ as a general pattern matching most frequent feature patterns identified in literature [8]. In this step, the algorithm currently omits those chunks whose main word is a pronoun to avoid ambiguous interpretations (see Section 7 for future work suggestions).
3. **Noun phrase filtering:** exclude all feature candidates not matching a list of relevant syntactic dependencies regarding the root element of the phrase. This step is customizable, so requests may include a list of dependencies for the tool to consider. If none is provided, based on preliminary results, the tool uses a default set, namely direct objects (*dobj* in spaCy), adverbial clauses (*advcl*), noun phrases in apposition (*appos*), and subjects (*ROOT*).
4. **Stopword filtering:** exclude the data items that depend on one of a specific set of irrelevant verbs. This set of verbs can be passed along the request for processing. By default, no data item is excluded in this phase.
5. **Cleaning:** clean all the resulting noun phrases. In this process, the tool removes any stopwords present in the potential feature (i.e. any tokens whose PoS is possessive pronouns, determiners or punctuation).
6. **Normalizing:** normalize the data. The remaining words in the cleaned noun phrases are reverted to its basic, lemmatized form. This is done to avoid duplicated extracted features.
7. **Feature building:** append the lemmatized root (if any) to the cleaned phrases.

4. Demo

TransFeatEx⁴ exposes two main use cases:

- **Batch-processing:** it can be used to process in bulk a data-set of NL documents. It supports text identifiers to easily match a source text to the extracted features and, subsequently, to a mobile app.

³The Transformer-based pipeline used identifies this syntactic construct and offers it through the "noun chunk" structure, which includes additional information, such as the head of the phrase.

⁴Examples available at: https://github.com/gessi-chatbots/NLP_pipeline. Check README for demo details.

1. Deploy TransFeatEx as a standalone web service (see “*How to install*”).
 2. Send a request to `/extract-features` with the required payload format (see “*How to use*”), which includes the set of text documents and customization options.
 3. If sentiment analysis filtering is required, send a POST request to `/review-extraction` with the relevant payload and the subjectivity/polarity thresholds.
 4. The textual data goes through the processes explained in Section 3. When processing reviews, the data is filtered out based on the subjectivity threshold specified.
 5. Receive the response with the text-linked extracted features (see “*How to use*”).
- **Playground:** the tool also offers a playground feature to visually explore the results for a given text document with different configurations.
 1. Run the script `feature-visualization.py` with the two required parameters:
 - a) `-f`: text file to be processed (i.e., summary, description, changelog or user review).
 - b) `-c`: custom configuration file, including: (1) the list of relevant syntactic patterns for dependency parsing; and (2) the custom stopword list for stopwords filtering.
 2. After processing is done, the visualization will be available at `http://localhost:5000`.

Record your own routes **feature** on a map, **add waypoints feature**, **take pictures feature** along the itinerary and upload them to your Wikiloc account from your phone.

Enjoy free offline topographic maps **feature** from all over the world for you to use without coverage or data. Great when you're out in the mountains or traveling without Internet connection.

Want to **take your outdoor experiences feature** even further? With Wikiloc Premium:

- Turn your mobile **feature** into a GPS navigator. Your smartphone will guide you with a heading indicator and sound alerts to warn you if you go off-track during navigation.
- Live Tracking **feature** . Share your position **feature** in real time with your family & friends while **doing the route feature** .

Figure 2: Sample text with highlighted features (from the description of Wikiloc’s trailing app)

5. Planned evaluation

Data set preparation. We collected from Google Play and AlternativeTo a data set of Android mobile apps in the field of trail tracking, sports activities and other related support apps by using a set of related keywords from the given domain through its app search engine. For each app, we collected (1) metadata fields (e.g., app name), (2) proprietary documents (i.e., summary, description, changelog) and (3) user-generated documents (i.e., user reviews).

Evaluation metrics. We expect to use a combination of annotated data items (i.e., feature annotations from real users available in AlternativeTo) with a manual annotation process of the extracted features to compute evaluation metrics. While we plan to measure accuracy, precision, recall and f-measure, we also plan to differentiate two evaluation scenarios. The first scenario aims at evaluating feature extraction of multiple documents for a single app, for which we plan to focus on recall (a more permissive approach can be used to only report matching features

among different documents). The second scenario aims at evaluating feature extraction of a single document, for which we plan to focus on precision (a more restrictive approach will reduce the amount of noisy results, i.e., false positives).

Experiment set up. We plan to conduct multiple experiments using the customizable layers of the tool pipeline as experimentation variables: sentiment analysis filtering, noun phrase selection and noun phrase filtering. For each of these filters, we will define multiple configuration settings, from more permissive to more restrictive set of criteria (as depicted above). We will use a cross-validation strategy to determine the optimal tool configuration.

6. Related work

The SAFE approach is one of the most popular contributions to the RE community in the field of mobile app feature extraction [8]. They mainly focus on a POS-based pattern analysis inferred from a data-set of mobile app descriptions and reviews. They highlighted as the most common patterns (1) those composed by a noun chunk (e.g., *Noun+Noun*, *Verb+Noun*), and (2) those composed by two words. However, they did not consider syntactic dependency parsing, and neither applied any kind of sentiment analysis layer to user generated data. Concerning syntactic analysis, related literature mainly focuses on POS and syntactic-based patterns [8], TFIDF-based keyword extraction [15] or topic modelling approaches [14]. Concerning sentiment analysis, most approaches either use syntactic, rule-based approaches like VADER or pattern analysers [14], supervised machine learning approaches [14, 15] and, more recently, deep learning approaches [16] Finally, there are few solutions based on using large language models. KEFE is a BERT-based approach using a deep learning classifier to filter out non-relevant syntactically extracted features [17]. RE-BERT, similarly to KEFE, focuses on supervised classification with contextual word embeddings using BERT as a pre-trained language model [18].

7. Conclusions

TransFeatEx provides a software-based technical solution to test and evaluate a customizable feature extraction process based on the combination of state-of-the-art large language models and consolidated syntactic and semantic linguistic analysis. The tool is expected to lay the groundwork for future research and application on domain-specific scenarios, while facilitating the process of its use both for analytical and playground purposes. As future work, we envisage three main immediate research action points: (1) to conduct the planned evaluation depicted in Section 5; (2) to extend the scope of syntactic structures covered in the syntactic pipeline; (3) to explore the fine-tuning process of a large language model to specifically fit the feature extraction task by exploring the linguistic knowledge embedded into the model; and (4) to enhance the SA filter using more advanced techniques.

Acknowledgments

With the support from the Secretariat for Universities and Research of the Ministry of Business and Knowledge of the Government of Catalonia and the European Social Fund. This paper has

been funded by the Spanish Ministerio de Ciencia e Innovación under project / funding scheme PID2020-117191RB-I00 / AEI/10.13039/501100011033.

References

- [1] A. Onan, et al., Ensemble of keyword extraction methods and classifiers in text classification, *Expert Systems with Applications* (2016).
- [2] K. Kowsari, et al., Text classification algorithms: A survey, *Information (Switzerland)* (2019).
- [3] O. Araque, et al., Enhancing deep learning sentiment analysis with ensemble techniques in social applications, *Expert Systems with Applications* (2017).
- [4] J. Park, et al., ADC: Advanced document clustering using contextualized representations, *Expert Systems with Applications* (2019).
- [5] S. Panichella, et al., How can i improve my app? Classifying user reviews for software maintenance and evolution, in: *31st International Conference on Software Maintenance and Evolution*, 2015.
- [6] W. Maalej, et al., On the automatic classification of app reviews, *Requirements Engineering* (2016).
- [7] ACMNL, Market study into mobile app stores, 2022. URL: <https://www.acm.nl/en/publications/acm-launches-market-study-mobile-app-stores>, Accessed 22 November 2022.
- [8] T. Johann, et al., SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews, in: *25th International Requirements Engineering Conference (RE)*, 2017.
- [9] M. K. Uddin, et al., Comparison of Text-Based and Feature-Based Semantic Similarity Between Android Apps, 2020, p. 530–545.
- [10] A. Radford, K. Narasimhan, Improving Language Understanding by Generative Pre-Training, 2018.
- [11] J. Devlin, et al., BERT: Pre-training of deep bidirectional transformers for language understanding, in: *NAACL HLT 2019 - 2019*, 2019.
- [12] Y. Liu, et al., RoBERTa: A Robustly Optimized BERT Pretraining Approach, 2019. URL: <https://arxiv.org/abs/1907.11692>.
- [13] S. Lim, et al., Data-Driven Requirements Elicitation: A Systematic Literature Review, *SN Computer Science* (2021).
- [14] S. Kumari, Z. A. Memon, Extracting feature requests from online reviews of travel industry, *Acta Scientiarum - Technology* 44 (2022).
- [15] M. Kasri, et al., A Comparison of Features Extraction Methods for Arabic Sentiment Analysis, in: *4th International Conference on Big Data and Internet of Things*, 2020.
- [16] M. S. Akhtar, et al., How Intense Are You? Predicting Intensities of Emotions and Sentiments using Stacked Ensemble, *IEEE Computational Intelligence Magazine* (2020).
- [17] H. Wu, et al., Identifying key features from app user reviews, in: *International Conference on Software Engineering*, 2021.
- [18] A. F. de Araújo, R. M. Marcacini, RE-BERT: Automatic Extraction of Software Requirements from App Reviews Using BERT Language Model, in: *36th Annual ACM Symposium on Applied Computing*, 2021.