Table 1: Categories Definition

Category	Description	User Feedback Example
Information Giving	Sentences that inform or update users or developers about an aspect related to the app	`"This app runs so smoothly and I rarely have issues with it anymore"`
Information Seeking	Sentences related to attempts to obtain information or help from other users or developers	`"Is there a way of getting the last version back?"`
Feature Request	Sentences expressing ideas, suggestions or needs for improving or enhancing the app or its functionalities	`"'Please restore a way to open links in external browser or let us save photos"`
Problem Discovery	Sentences describing issues with the app or unexpected behaviours	`"App crashes when new power up notice pops up"`
Other	Sentences do not providing any useful feedback to developers	`"What a fun app"`

user reviews, that are useful for maintenance perspective, in five categories: *feature request, problem discovery, information seeking, information giving* and *other*. Table 1 shows, for each category: (i) the category name, (ii) the category description and (iii) an example sentence belonging to category. As described in [15], these categories emerged from a systematic mapping between the taxonomy of topics occurring in app reviews described by Pagano et al. [3] and the taxonomy of categories of sentences occurring in developers' discussions over development-specific communication means [16, 17]. Specifically, such taxonomy is defined to model feedback from user reviews that are important from a maintenance perspective.

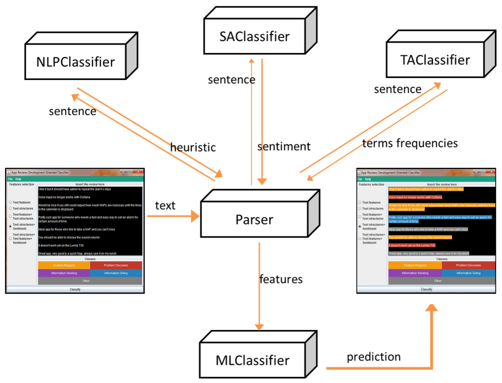

Figure 1: ARdoc's architecture overview

Figure 1 depicts `ARdoc`'s architecture. The main tool's module is represented by the `Parser`, which prepares the text for the analysis (i.e., text cleaning, sentence splitting, etc.). Our `Parser` exploits the functionalities provided by the Stanford CoreNLP API [18], which annotates the natural text with a set of meaningful tags. Specifically, it instantiates a pipeline with annotations for tokenization and sentences splitting. Once the text is divided into sentences ARdoc extracts from each of these sentences three kinds of features: (i) the lexicon (i.e., the words used in the sentence) through the TAClassifier, the structure (i.e., grammatical frame of the sentence) through the NLPClassifier, and (iii) the sentiment (i.e., a quantitative value assigned to the sentence expressing an affect or mood) through the SA Classifier. Finally, in the last step the `MLClassifier` uses the NLP, TA and SA information extracted in the previous phase of the approach to classify app reviews according to the taxonomy reported in Table 1 by exploiting a Machine Learning (ML) algorithm. We briefly describe, in Section 2.1, the information extracted by our tool from app reviews and, in Section 2.2, the classification techniques we adopted.

2.1 Features Extraction

The `NLPClassifier` implements a set of NLP heuristics to automatically detect recurrent linguistic patterns present in user reviews. Through a manual inspection of 500 reviews from different kinds of apps we identified 246 recurrent linguistic patterns[1] often occurring in app reviews, and for each of these patterns we implemented an NLP heuristic in order to automatically recognize it (more details about the process performed for the definition of the heuristics are available in our previous work [15]). The `NLP classifier` uses the Stanford Typed Dependencies (STD) parser [19], a natural language parser which represents dependencies between individual words contained in sentences and labels each dependency with a specific grammatical relation (e.g., subject or direct/indirect object).
Through the analysis of the typed dependencies, each NLP heuristic tries to detect the presence of a text structure that may be connected to one of the categories in Table 1, looking for the occurrences of specific keywords in precise grammatical roles and/or specific grammatical structures. For each sentence in input, the `NLPClassifier` returns the corresponding linguistic pattern. If the sentence does not match any of the patterns we defined, the classifier simply returns the label "No patterns found".

The `SAClassifier` analyzes the sentences trough the sentiment annotator provided by the Stanford CoreNLP [18] and for each sentence in input returns a sentiment value from 1 (strong negative) to 5 (strong positive). We use this sentiment prediction system because it is independent of hard-coded dictionaries – a drawback from lexical sentiment analysis techniques that have been previously used for the analysis of app reviews [12], [20], [14]. The `TAClassifier` exploits the functionalities provided by the Apache Lucene API[2] for analyzing text content in user reviews. Specifically, this classifier performs a stop-words removal (i.e., words not containing important information) through the `StopFilter` and normalizes the input sentences (i.e., reduces the inflected words in the root form) through the `EnglishStemmer` in combination with the `SnowballFilter` in order to extract a set of meaningful terms that are weighted using the *tf* (term frequency), which weights each word i in a review j as:

$$tf_{i,j} = \frac{rf_{i,j}}{\sum_{k=1}^{m} rf_{k,j}}$$

where $\mathbf{rf}_{i,j}$ is the raw frequency (number of occurrences) of word i in review j. We use the *tf* (term frequency) instead of *tf-idf* indexing because the use of the *idf* penalizes too much terms (as "fix", "problem", or "feature") appearing in many reviews [21]. Such terms may constitute interesting features for guiding ML techniques in classifying useful feedback.

2.2 Classification via ML Techniques

We used the NLP, TA and SA features extracted in the previous phase of the approach to train ML techniques and classify app reviews according to the taxonomy in Table 1. To integrate ML algorithms in our code, we used the Weka API [22]. The `MLClassifier` module provides a set of java methods for prediction, each of them exploits a different pre-trained ML model and uses a specific combination of the three kinds of extracted features: (i) text features

[1]http://www.ifi.uzh.ch/seal/people/panichella/Appendix.pdf
[2]http://lucene.apache.org