User Modeling and User-Adapted Interaction - Special Issue manuscript No.
(will be inserted by the editor)

Generating Post-Hoc Review-based Natural Language Justifications for Recommender Systems

Cataldo Musto · Marco de Gemmis · Pasquale Lops · Giovanni Semeraro

Received: August 12, 2019 - First Revision: April 7, 2020

Abstract In this article, we present a framework to build *post-hoc natural language justifications* that supports the suggestions generated by a recommendation algorithm. Our methodology is based on the intuition that *reviews' excerpts* contain many relevant information that can be used to *justify* a recommendation, thus we propose a *black box* explanation strategy that takes as input a recommended item and a set of reviews, and builds as output a *post-hoc* natural language justification which is completely independent of the underlying recommendation model.

To validate our claims, we also introduce three different implementations of our conceptual framework: the first one uses natural language processing and sentiment analysis techniques to identify relevant and distinguishing aspects discussed in the reviews, and combines reviews' excerpts mentioning these aspects in a natural language justification which is presented to the target user. The second implementation extends the first one by introducing automatic aspect extraction and text summarization, which are exploited to generate a unique *synthesis* presenting the main characteristics of the item that is used as justification. Finally, the third implementation tackles the problem of generating a *context-aware justification*, that is to say, a justification that differs on varying of the different contextual situations, by automatically learning a lexicon for each contextual setting and by using such a lexicon to diversify the justifications.

In the experimental evaluation we carried out three user studies in different domains, and the results showed that our methodology is able to make the recommendation process more transparent, engaging and trustful for the users, thus confirming the validity of the intuitions behind this work.

Cataldo Musto
Department of Computer Science
University of Bari 'Aldo Moro'
Italy
E-mail: cataldo.musto@uniba.it

Keywords Recommender Systems · Explanation · Text Summarization · Natural Language Processing · Sentiment Analysis

1 Introduction

Adaptive and personalized systems play an increasingly important role in our daily lives, since we more and more rely on platforms that tailor their behavior on the ground of our preferences and needs and support us in a broad range of heterogeneous decision-making tasks. As an example, we are used to exploit Spotify to get the best music playlist for our workout, we ask Netflix to suggest us a movie to watch on a rainy night at home, and we use Amazon to get recommendations about items to buy.

The rise of such dynamics made technologies such as *personalized search engines* [42], *recommender systems* [25] and *intelligent personal assistants* [20] very popular and essential. However, as the importance of such technologies in our everyday lives grows, it is fundamental that the *internal mechanisms* that guide the behavior of the algorithms are as clear and transparent as possible, especially in domains where complex decisions are made, such as medicine, finance or electricity market [16].

This claim is supported by the recent *General Data Protection Regulation (GDPR)*[1], which emphasized and regulated the *users' right to explanation* [19] when people face machine learning-based or, generally speaking, intelligent systems. Such a need is even more felt for recommender systems (RSs) since, as shown by Cramer et al. [15], it exists a relationship between the transparency of a RS and users' acceptance of the recommendations. Similarly, Sinha et al. [46], showed that a higher *transparency* of the algorithm leads to a higher *trust* of the user.

Accordingly, recommender systems (and adaptive systems, in general) need to progressively evolve from simple *black boxes*, whose only goal is to provide effective suggestions, to systems able to encode some kind of *explanation* capability, that is to say, the ability to properly explain or justify their own behavior [49].

To this end, a huge research effort has been recently devoted to developing methodologies and algorithms able to *explain* the behavior of recommendation algorithms [41]. According to the taxonomy of explanation strategies proposed by Friedrich et al. [17], these methodologies range from *content-based* techniques based on the analysis of textual features that describe the recommendation (*e.g., "I suggest you this movie since you liked movies of the same director."*) to *collaborative* explanations [23], that support the recommendation by informing the user about the behavior of her neighbors (*"e.g., I suggest you this item since your neighbors have liked it.")*, and hybrid techniques that try to take the best from both the methods.

In this work, we follow the first research line, since we propose a content-based approach to build *post-hoc natural language justifications* that rely on

[1] http://ec.europa.eu/justice/data-protection/reform/files/regulation_oj_en.pdf

the information gathered from *users' reviews* that discuss the recommended item. The *post-hoc* nature of our justification is the first distinguishing trait of this work, since we generate *justifications* that are completely independent from the underlying recommendation models and completely separated from the step of generating the recommendations.

Our interest towards *review data* follows a very simple intuition. Indeed, this information is very interesting from both a *quantitative* and a *qualitative* point of view: on the one hand, they represent an important portion of the textual data available online, suffice it to say that more than 700 million reviews and opinions are available on TripAdvisor[2] and 26,380 reviews are posted on Yelp *every minute*[3]. On the other hand, the information held by these systems is very valuable, since it gives a clear picture of what people *like* and what people *think* about the places they visit and the experiences they have, and such an intuition is further confirmed by the findings emerging from several related research in the area of opinion mining [12] and recommender systems [3,30], which showed that the exploitation of reviews data can lead to more precise and effective recommendations [11,22,36].

However, differently from the above mentioned research, in this work we investigated the impact of the information conveyed by users' reviews in a different task: given that reviews usually contain a lot of evidences about the *aspects* of the item that impressed the users the most (see Figure 1, gathered from a popular community[4]), we tried to exploit such information to generate a *natural language justification* that supports the suggestions generated by a recommendation algorithm. Specifically, we processed and analyzed the reviews in order to obtain a set of characteristics that are often discussed in the reviews (with a *positive* sentiment, of course) and can induce the user in enjoying the recommended item.

To this end, we designed a general methodology to build natural language justifications based on users' reviews, and we introduced three different implementations of our method. The first one exploits natural language processing and sentiment analysis techniques to identify relevant and distinguishing aspects that are discussed in the review, and combines excerpts mentioning these aspects in a natural language justification of the recommended item. The second methodology extends the first one by introducing more sophisticated machine learning techniques, such as automatic aspect extraction and text summarization, and exploits such techniques to generate a unique *synthesis* that includes the main characteristics of the item. Finally, the third implementation tackles the problem of generating a *context-aware justification*, that it to say, a justification that differs on varying of the different contextual settings. This is done by automatically learning a lexicon for each context and by using such a lexicon to diversify the justifications on the ground of the different contextual situations.

[2] https://tripadvisor.mediaroom.com/us

[3] http://expandedramblings.com/index.php/yelp-statistics/3/

[4] https://www.rottentomatoes.com/m/ring

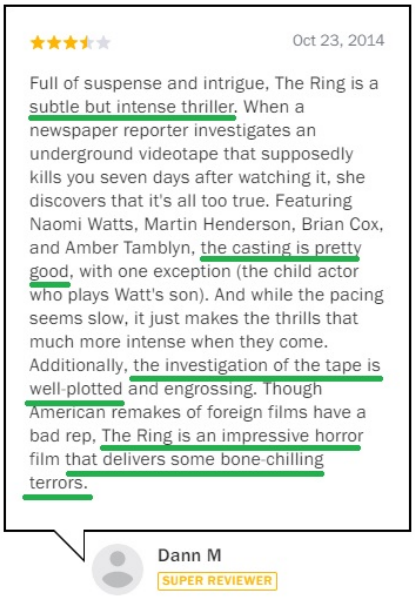

Fig. 1: Example of a review of the movie *The Ring*. Relevant aspects that impressed the user are underlined in green.

As previously stated, the most distinguishing aspect of our framework lies in *its black box nature*, since our methodology is completely independent of the underlying recommendation model: it only needs as input a set of reviews and a recommended item, and produces as output a natural language justification based on a subset of automatically extracted reviews excerpts. This makes our approach particularly flexible and solid. Indeed, it is immune to cold-start problems since neither a profile of the user nor a minimum amount of users and ratings is necessary to run the framework. In our opinion, this is strong point of the strategy that makes it particularly useful to provide users with transparent recommendations.

Moreover, we want to emphasize that in this article we used the term *justification* over *explanation*. Even if in most of the work they are used as synonyms, we followed the definition provided in [7], where it is stated that a *justification explains why a decision is a good one, without explaining exactly how it was made*, while an explanation is related to the concept of interpretability, that is to say, *if their operations can be understood by a human*.

Accordingly, the strategy we implemented is closer to the idea of justification, since we aim to identify the most relevant and distinguishing characteristics of the item that are automatically obtained by mining users' reviews without taking into account any information about the internal operations the recommendation algorithm carries out. Rather than explaining *how* the algorithm generated the recommendation, our natural language justification

is more devoted to describe *why* a user would be interested in the item, in order to provide her with a mean to make a more *informed decision* about consuming the item or not.

To summarize, the main goal of this article is to investigate the effectiveness of users' reviews as a source to build a post-hoc natural language justification supporting a recommendation. To this end:

- we propose a methodology based on *natural language processing* and *sentiment analysis* to process users' reviews in order to identify relevant and distinguishing aspects that characterize the item;
- we present three different implementations of the method, gradually more sophisticated, that also tackle the problem of generating a justification that can adapt on varying of the different contextual settings.
- we evaluate the effectiveness of the methodology and its impact on real users by carrying out a user study in three different domains, such as movies, books and restaurants. Throughout these experiments we compared our approach to other black-box explanation frameworks such as a classic *feature-based explanation algorithm* [37].

The rest of the paper is organized as follows: first, in Section 2 we provide an overview of relevant work in the area of explanation, while in Section 3 we describe the general organization of our pipeline and we present the different implementation of the method. Next, Section 4 focuses on the design of the user studies we carried out to evaluate our methodology and presents the results of the experiments. Finally, Section 5 discusses strengths and weaknesses of the approach, while conclusions and future work of the current research are provided in Section 6.

2 Related Work

The idea of improving interactive systems by introducing some kind of *explanation* mechanisms dates back to the early 90s, thanks to Johnson et al. [26]. Next, the first attempt towards the exploitation of such facilities in RSs was proposed in 2000 by Herlocker et al. [23], who presented a user study based on 21 different explanation interfaces and investigated how each interface had impact on the users' acceptance of the recommendations. Later, a similar analysis was proposed by Gedikli et al. [18], who compared a subset of Herlocker's techniques to an explanation based on a personalized tag cloud on the ground of different *explanation goals* that follow those proposed by Tintarev and Masthoff [50], namely: *transparency, scrutability, trust, effectiveness, persuasiveness, efficiency, satisfaction.*

The amount of research regarding the development of explanation and justification methods tremendously increased in the last few years, as confirmed by several studies that stressed the importance of transparency in intelligent systems [27]. The most recent discussion of the advances in the area has been presented by Nunes et al. [41], and we suggest referring to that overview to deepen the discussion.

According to the taxonomy of explanation strategies in RSs provided by Friedrich et al. [17], our technique can be classified as a *black box* methodology, since the model is not aware of the underlying recommendation model which is used to generate the suggestions. This means that our method is able to justify and explain the recommendations provided by every recommendation algorithm[5]. This is a very distinguishing aspects of this work, since most of the methodologies presented in the literature fall into the *white box* category. As an example, Bilgic et al. [6] used the information about the neighborhood to justify the recommendation generated by the system. Similarly, Coyle and Smyth [14] exploit the search histories of a community of online users as a source of text explanations.

Moreover, we can also state that we proposed a *content-based* explanation strategy, since our justifications rely on descriptive features extracted from users' reviews. Other relevant content-based explanation methodologies have been proposed by Symeonidis et al. [48], Cramer et al. [15] and Vig et al. [51]. In the first two attempts, a very similar strategy based on the identification of the overlapping features that are in common between the profile of the user and the characteristics of the recommended item is used to generate the explanation. Such an approach is improved in [51], since the authors introduce a *tag-based explanation strategy* that relies on the combination of tag relevance, measuring how good is the tag in describing the recommended item, and tag preference, measuring how interested is the user in the topics related to the tag. Another relevant attempt in the area of content-based explanation algorithms is represented by ExpLOD [37], an explanation framework based on the information extracted from the Linked Open Data cloud [8]. As shown in following research discussing the methodology [38], the approach emerged as more effective than other explanation strategies and also resulted as *algorithmic-agnostic* and *domain-agnostic*. It is worth to note that ExpLOD shares a similar structure and a similar goal with our framework, since it is a black box explanation framework (as our methodology) based on the information available in DBpedia (instead of user reviews). Accordingly, we adopted ExpLOD as a baseline to evaluate the effectiveness of our framework.

With respect to the main literature in the area of content-based explanation strategies, one of the distinguishing aspects of the current work lies in the exploitation of *users' reviews* to generate the justifications. As previously introduced, the exploitation of these features is an established research line in the area of recommender systems. However, as shown in a recent overview [24], most of the approaches use the features extracted from users' reviews to enhance and improve the recommendation models, while we used these signals to generate a natural language justification that support the suggestions returned by a recommendation algorithm.

The exploitation of review-based features for explanation and justification purposes is a relatively newer research line. However, as shown in recent lit-

[5] It is worth to note that this aspect is neither investigated nor evaluated in the current research, and it is left for future work.

erature, most of the approaches that use review-based features focused on the generation of an *explainable recommendations* rather than explaining the behavior of the algorithm itself, as we do in our work. As an example, Baral et al. [4] propose an approach based on deep neural networks that quantifies the relationship between aspects and reviews and formulates user-aspect bipartite relation as a bipartite graph. Next, by using dense sub-graph extraction and ranking-based technique an *explainable recommendation* is returned. Similarly, He et al. [22] exploited tri-partite modeling of user-item-aspect tuples and used graph-based ranking to find the most relevant aspects of a user that match with relevant aspects of places. The use of neural networks is also investigated by Chen et al., who propose in [10] a Neural Attentional Regression model with Review-level Explanations (NARRE) for recommendation. The core of the approach is an attention mechanism that catches the usefulness of reviews. Such information is used to predict the interest of the user towards the item and the usefulness of each review, simultaneously. Therefore, the reviews labeled as highly-useful are exploited to provide the user with review-level explanations.

Even if these techniques may be figured out as closely related to our framework, there is a significant difference between our methodology and the above mentioned research: indeed, all these works are designed to generate *recommendations* that are *explainable*. Conversely, we did not generate any recommendation. We aimed to develop a general framework that builds natural language justifications that are *independent* of the underlying recommendation model. Due to this difference, we did not compare to any of this work in our experimental evaluation.

Another interesting work in the area has been recently proposed by Chen et al. [13]. In this case, the authors statically defined a set of relevant aspects and used sentiment analysis techniques to associate a sentiment score to each aspect. Next, they used such an information as explanation of the recommendation. Our work differs from [13] for two important elements: *(i)* we focused our attention on the algorithmic aspects of our methodology, thus we evaluated the effectiveness of the justifications rather than their impact on the users in terms of interface design; *(ii)* we did not bound on a fixed set of static aspects and we left the justification strategy deciding and identifying the most relevant characteristics of the item.

An approach that shares many aspects with our methodology has been presented by Muhammad et al. in [35]. In particular, they generate explanations from a set of aspects that are automatically obtained and ranked by analyzing user-generated reviews. However, even this approach significantly differs from our methodology since we proposed a more sophisticated and more satisfying generation phase. Indeed, Muhammad et al. [35] just focus on the identification of relevant aspects of the items (*e.g.*, bar, service, parking, etc.), without going into details of the *reasons* behind the choice of highlighting a specific characteristic, while our approach combines relevant excerpts gathered from the original reviews to build a more comprehensive natural language justification.

The generation of a natural language explanation is another distinguishing aspect of the work. In this research line, Chang et al. propose in [9] an approach

to generate crowd-based personalized natural language explanation that based on the work carried out on a crowd-sourcing platform. Differently from this approach, we exploited a fully automated pipeline exploiting natural language processing, sentiment analysis and text summarization techniques rather than relying on the annotations provided by a set of workers.

A final distinguishing aspect of this work regards the experimental design. Differently from most of the related literature, which focuses on the analysis of the interface-related aspects of the explanations (as [35] and [13]), we followed the same experimental protocol proposed in [49] and we evaluated the impact of the explanations on final users in terms of *transparency, engagement, trust, persuasiveness and effectiveness*.

To conclude, it is worth to note that two of the contributions provided by this work, that is to say, the exploitation of *text summarization* techniques to generate justifications and the generation of *context-aware justifications* can be considered as totally *novel* for the area. The only work that investigates the idea of adopting of features inspired by context (e.g., *"I suggest you this movie since you like this genre in rainy days"*) to explain recommendations has been proposed by Misztal et al. in [34]. However, this work significantly differs from our framework since they did not propose the idea of diversifying the justifications *of the same items* on varying of different contextual settings in which the item is consumed. Similarly, to the best of our knowledge, there is no literature investigating neither the effectiveness of text summarization techniques in this specific task.

3 Methodology

In this section, we present our methodology to generate *post-hoc natural language justifications*. As shown in Figure 2, from the users' point of view our approach takes as input a *set of reviews, a recommendation and (eventually) some contextual settings*, and returns as output a *natural language justification* supporting the recommendation.

As previously stated, we want to stress that we propose a *black-box* methodology. This means that our justifications are built regardless of the specific recommendation algorithm which is used to generate the recommendations, thus our system is also *algorithm-independent*[6].

Generally speaking, our workflow can be seen as a *conceptual model* that identifies the main components that are necessary to generate a natural language justification that is based on the information gathered from users' reviews. As shown in Figure 2, our methodology is based on three main building blocks, that is to say, an Aspect Extraction component, an Aspect Ranking component and a Generation component, whose goal is to extract the aspects discussed in the reviews, to rank the aspects in order to identify the

[6] As previously stated, we will investigate the algorithmic-independent nature of the methodology as a future work, by evaluating the effectiveness of the justifications on varying of the different algorithms used to generate the recommendations.

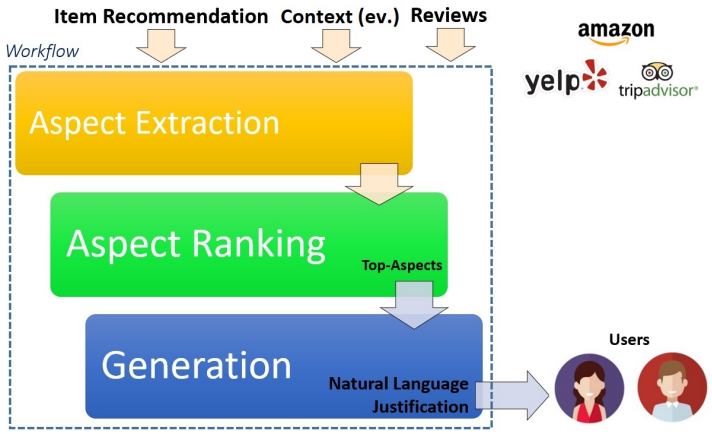

Fig. 2: Workflow carried out by our framework.

most relevant ones, and to generate a natural language justification based on these aspects, respectively.

Of course, each implementation of our conceptual model can be seen as an *instance* of such a generic pipeline. Accordingly, every time the workflow is instantiated, a different implementation of the single components will be proposed. As an example, each instance can implement a different strategy to rank the aspects in the Aspect Ranking phase or to generate the justifications through the Generation module.

Clearly, different implementations of the component will lead to the generation of different natural language justifications, thus one of the goals of this work will be to evaluate the effectiveness of the different implementations and to discuss strength and weaknesses of each variant of the pipeline.

As previously stated, this work introduces three different strategies to implement the conceptual model presented in Figure 2:

- Justifications based on Natural Language Processing and Sentiment Analysis techniques;
- Justifications based on Automatic Aspect Extraction and Text Summarization techniques;
- Context-aware Justifications based on Automatic Lexicon Generation.

For each implementation, we provide the details of the approach and we explain the design choices. For the sake of brevity, from now on we will refer to the three different implementations of the pipeline as NLP-Pipeline, TS-Pipeline and Context-Pipeline.

3.1 Justifications based on Natural Language Processing and Sentiment Analysis techniques (NLP-Pipeline)

The NLP-Pipeline exploits natural language processing and sentiment analysis techniques to build natural language justifications. This is a very *basic* implementation of the pipeline, which will be used as a *baseline* to assess and evaluate the effectiveness of more sophisticated implementations of the methodology.

In a nutshell, this implementation is based on the following steps: *(1)* the distinguishing *aspects* that characterize the item are extracted; *(2)* a ranking function is used to identify the most relevant ones; *(3)* excerpts of the reviews discussing such aspects are extracted and a *natural language template* is filled in through the aggregation of these sentences.

In the following sections, we separately discuss each step of the pipeline. Clearly, the names of the steps follow the names of the components of the conceptual workflow we presented in Figure 2.

3.1.1 Aspect Extraction

The goal of the Aspect Extraction module is to analyze the content conveyed by users' reviews and to extract a set of distinguishing aspects describing the recommended item.

Formally, our strategy takes as input a set of reviews $R = \{r_1, r_2 \dots r_n\}$ and produces a set of triples $\langle r_i, a_{ij}, rel(a_{ij}, r_i)\rangle$ representing the review r_i, the *j-th* aspect a_{ij} extracted from the review r_i and the relevance $rel(a_{ij}, r_i)$ of the aspect a_{ij} in the review r_i.

To this end, we adopted a pipeline of natural language processing techniques to process the content and identify such aspects. According to previous research [40], descriptive features of an item are usually represented using *nouns*, thus we run a Part-of-Speech (POS) tagging algorithm over the set of the reviews to obtain representative nouns. For the sake of simplicity, the goal of a POS-tagging algorithm is to assign a grammatical category to each word in an input text. In our case, the input text is represented by the reviews and the output is represented by a set of words whose grammatical category is related to *nouns* (NN, NNS, NNP, NNPS[7]).

To sum up, after this first step of the pipeline, the complete set of reviews R is transformed into a set of triples $\langle r_i, a_{ij}, rel(a_{ij}, r_i)\rangle$ that identifies the aspects that are usually used to describe and discuss the item i. As relevance score $rel(a_{ij}, r_i)$ the *counting* of the occurrences of the aspect a_{ij} in the reviews of the item i is used.

By referring to the example in Figure 1, nouns as *thriller, cast, investigation, tape, horror, terror* and so on would have been identified by the algorithm as potential *aspects* that are discussed in the reviews of the movie *The Ring*.

[7] A discussion about the available grammatical categories is out of scope to this paper. However, one of the most popular tagset, the Penn Treebank tagset, contains 36 categories for words. We suggest to refer to [32] for further reading about POS-tagging.

3.1.2 Aspect Ranking

The number of the available reviews discussing the item is often huge, thus the set of the triples returned by the ASPECT EXTRACTION module is usually very large as well. Accordingly, the goal of the ASPECT RANKING module is to filter the complete set of the aspects to identify the most relevant ones that best describe and characterize the item.

Given that the goal of the workflow is to provide the user with a precise and convincing *justification* that supports the recommendation, through this module we want to identify aspects that are *relevant* and *distinguishing*. In this specific scenario, we are interested in those aspects that are discussed in *many reviews* with a *positive sentiment score*.

To this end, we used a combination of natural language processing and sentiment analysis techniques. As preliminary processing, we split each review in *sentences*: this choice is due to the fact that a review may include different (and maybe conflicting) opinions (*e.g., a user liked the cast of a movie, but he did not like the music*), thus we decided to calculate the *sentiment* conveyed by the review at a *sentence-based* level, in order to obtain a more precise identification of the opinion the user expressed in each single piece of the reviews. Next, for each aspect previously returned by the ASPECT EXTRACTION, we count the number of sentences in which the aspect appears and we calculate the average sentiment that emerges in the sentences discussing the aspect.

Formally, let R_i be the set of the reviews available for item i defined as previously introduced, and let S_{R_i} the set of the sentences we obtained by processing the original reviews, for each aspect a previously identified we calculate a score $score(a, R_i)$ as follows:

$$score(a, R_i) = (\alpha \frac{rel_{a,R_i}}{|S_{R_i}|} + \beta \frac{pos(a, R_i)}{pos(a, R_i) + neg(a, R_i)}) * IAF(a, R_i) \quad (1)$$

In the previous formula, rel_{a,R_i} is the number of sentences mentioning the aspect a in R_i, while $pos(a, R_i)$ and $neg(a, R_i)$ represent the number of sentences conveying a positive and negative opinion according to the output of sentiment analysis algorithm, respectively[8]. Thus, the second factor of the formula counts the ratio of sentences that associate a positive opinion to that aspect.

Next, $IAF(a, R_i)$ is the *inverse aspect frequency*, an adaptation of the classic IDF[9] that tries to smooth the score of very popular aspects. Finally, α and β are two parameters that weigh more the *popular aspects* or those that *positively polarized the opinion* of the users, respectively.

[8] Sentences conveying a neutral opinion were ignored. More details about the sentiment analysis algorithm used in the pipeline are provided in the next section.

[9] As the IDF calculates the number of documents that contain a term, the IAF counts the number of items in which at least one review discusses the aspect a. The lower the number, the higher the IAF score.

The rationale behind Formula (1) is to give a higher score to those aspects that are relevant and distinguishing. The relevance is given by the number of reviews that discuss a particular aspect, while if the community has a positively polarized opinion towards an aspect, it is labeled as distinguishing. At the end of this step, the aspects are ranked according to their descending score and the top-K are passed to the GENERATION module. The top-K aspects are labeled as *main aspects* of the item. By referring again to the example in Figure 1, it is likely that aspects as *cast* or *horror* would decrease their importance due to their popularity, at the expense of other aspects as *investigation, terror* and *tape* which are likely to be more distinguishing.

3.1.3 Generation

Once the most relevant aspects discussing the item have been obtained, the final step of the workflow is carried out by the GENERATION module. The goal of this component is to combine the aspects returned by the ASPECT RANKING module in a *natural language justification* that is provided to the user to support the recommendation she received.

In particular, we generate our justification by filling in a *template-based structure*. However, differently from similar work in the area, we used some excerpts extracted from the reviews rather than some descriptive properties gathered from some external knowledge source.

Intuitively, for each aspect a returned by the ASPECT RANKING module, we scanned through all the sentences in R_i by looking for an excerpt that discusses the aspect a. Next, such excerpts are merged and combined by adopting a template-based structure. One of the templates we adopted, including two relevant aspects[10] and presented by exploiting the Backus-Naur Form, follows: (BNF)[11]:

⟨*justification*⟩ ::= ⟨*intro*⟩ because ⟨*excerpts*⟩

⟨*intro*⟩ ::= `I suggest you` ⟨*item_name*⟩ | `I propose you` ⟨*item_name*⟩ | `I recommend you` ⟨*item_name*⟩

⟨*excerpts*⟩ ::= ⟨*first_static_phrase*⟩ ⟨*review_excerpt*⟩ `.` ⟨*adverb*⟩ `,` ⟨*second_static_phrase*⟩ ⟨*review_excerpt*⟩ `.`

⟨*first_static_phrase*⟩ ::= `people who liked this movie think that` | `people who watched the movie think that`

⟨*second_static_phrase*⟩ ::= `other people think that` | `people liked <item_name> since`

⟨*adverb*⟩ ::= `Furthermore` | `Moreover` | `Besides`

[10] In the experimental evaluation we compared two different templates including different number of aspects. For the sake of simplicity we just report one of them since they differ in a small portion of the grammar.

[11] https://en.wikipedia.org/wiki/Backus-Naur_form

⟨*item_ name*⟩ ::= name of the recommended item

⟨*review_ excerpt*⟩ ::= a compliant sentence

Clearly, not all the available excerpts can be used in such a template. In order to generate a compliant justification we need to look for excerpts that match specific criteria. Accordingly, we run a *sentence filtering step* whose goal is to filter out the sentences that are supposed to be not useful in the final justification we want to build.

Formally, we first split each review $r_i \in R$ in sentences $s_{i1} \dots s_{im}$. Next, we verify the compliancy of each sentence s_i and we maintain only the sentences matching the following criteria:

1. s_i contains a *main aspect* $a_1 \dots a_k$;
2. s_i is longer than 5 tokens;
3. s_i expresses a *positive* sentiment;
4. s_i does not contain first-person verbs and first-person personal or possessive pronouns.

The rationale behind these heuristics is straightforward: we want to include in the justification only the sentences that mention a *relevant aspect* and express a *positive sentiment* about the item. Moreover, we decided to filter out very short and non-informative sentences as well as those using the *first person*. In this case, the intuition is to prefer excerpts having a more *impersonal* style (e.g., *'The movie has a great cast'*) than those expressing personal opinions (e.g., *I liked the cast of the movie, this is my favourite director*). It is worth to note that in the current implementation we did not employ any strategy to filter out sentences and excerpts that do not reach a *minimum language quality*. This is left as future work.

To sum up, at the end of such a pipeline we can provide each recommended item with a natural language justification which is built by collecting and combining excerpts of the reviews that discuss relevant and distinguishing aspects of the movie. As an example, the justification generated by running the NLP-Pipeline on the two aspects of the movie *The Ring* follows:

"I recommend you The Ring because people who liked the movie think that it delivers some bone-chilling terror. Moreover, people liked The Ring since the the casting is pretty good."

In this case, the main aspects which are identified by mining users' reviews are *terror* and *cast*. For each aspect we looked for compliant excerpts and we combined them in the final justification that is returned to the user to support the recommendation she received.

As previously stated, such a simple technique provides the user with a simple (but effective) justification. In the following, we will try to extend and improve this pipeline by introducing more sophisticated techniques to extract the aspects and to generate the natural language justifications.

3.2 Justifications based on Automatic Aspect Extraction and Text Summarization techniques (TS-Pipeline)

The NLP-PIPELINE we previously introduced can provide the user with an effective natural language justification. However, these justifications are typically characterized by two main issues, that can harm the satisfaction of the user, especially when such justification facilities are used for a *long* time:

- The method to identify the aspects is rather *empirical*, since it is based on a simple *counting* of the aspects, and it only uses an adaptation of the IDF to lighten the weight of very popular aspects.
- The template-based structure used in the Generation phase is very *static* and the justifications differs one from each other just a little. Accordingly, they can result as very similar and poorly satisfying for the user.

In order to tackle both these issues, we introduce the TS-Pipeline, an implementation that extends the NLP-Pipeline by introducing more sophisticated techniques in the Aspect Extraction and Generation phases [39].

In particular, we propose the adoption of the Kullback-Leibler divergence [29] to automatically identify the most relevant aspects and we used automatic text summarization to build a *summary* of the reviews that makes the most relevant characteristics of the item emerge. In this case, the intuition behind the TS-Pipeline is to conceive the justification as a *summary* of the most relevant and distinguishing aspects of the item, which is presented to the user as justification of the recommendation she received.

In the following we will discuss the implementation of each component of the pipeline.

3.2.1 Aspect Extraction

As for the first implementation, also in this case we assume that an effective justification should include relevant and distinguishing traits of the items.

However, differently from the previous approach, we exploited a more solid and sound method to automatically extract *aspects* from reviews that is based on the *Kullback-Leibler divergence* (KL-divergence, referred to as δ), a non-symmetric measure of the difference between two distributions.

Formally, given two corpora c_a and c_b and a term t, point-wise KL-divergence is calculated as:

$$\delta_t(c_a||c_b) = p(t, c_a) log \frac{p(t, c_a)}{p(t, c_b)} \tag{2}$$

Where $p(t, c_a)$ is the number of occurrences of the term t in corpus c_a. In a nutshell, this measure relies on the idea that the use of language differs when talking about a specific domain with respect to a general topic, thus this method identifies the aspects whose distribution in a specific domain

(e.g., *movie reviews*) diverges from that in a general corpus (e.g., the British National Corpus BNC[12]).

In other terms, the measure identifies those aspects which are mentioned in the reviews *more often* than usual. Given such a formulation, our strategy to identify the main aspects mentioned in the reviews follows:

Require: review r_i, general corpus BNC, domain corpus d
Ensure: set of main aspects A and relevance scores
$A = \{\}$, $T = nouns(r_i)$
for all $t_k \in T$ **do**
if $\delta_{t_k}(d||BNC) > \epsilon$ **then**
$a_{ij} \leftarrow t_k$
$A = A \bigcup \{a_{ij}\}$
$rel(a_{ij}, r_i) \leftarrow \delta_{t_k}$
end if
end for

By following the formalization we provided in Section 3.1.1, given a set of reviews $R = \{r_1, r_2 \dots r_n\}$ the module produces as output a set of triples $\langle r_i, a_{ij}, rel(a_{ij}, r_i)\rangle$ identical to that produced by the NLP-Pipeline.

To obtain the set of the aspects discussed in the review r_i we first extract all the *nouns* by running a POS-tagging algorithm as we did in the NLP-Pipeline. Next, we calculate the KL-divergence for each noun by using as corpora our set of reviews and the BNC. Finally, the nouns having a KL-divergence greater than a threshold ϵ are labeled as *aspects* and the KL-divergence score is used as relevance score $rel(a_{ij}, r_i)$.

3.2.2 Aspect Ranking

As for the NLP-Pipeline, also in the TS-Pipeline the set of the aspects extracted is typically huge, thus it is necessary to rank the aspects in order to identify the most relevant ones that are worth to be included in the justification. In this implementation, the relevance score $score(a, R_i)$ for each aspect a is calculated as follows:

$$score(a, R_i) = (\alpha \frac{n_{a,R_i}}{|S_{R_i}|} + \beta \frac{pos(a, R_i)}{pos(a, R_i) + neg(a, R_i)}) * rel_{a,R_i} \quad (3)$$

By comparing Formula (3) to the Formula (1) we used in the NLP-Pipeline two main differences emerge. First, the simple counting n_{a,R_i} of the aspects is used in the first part of the formula. Second, the relevance score of the aspect rel_{a,R_i}, that is to say, the Kullback-Leibler divergence, is used instead of the adaptation of the IDF. In this case, the intuition is that the KL-divergence already encodes some information about the importance of the aspect, thus it can be used to emphasize the score of very distinguishing concepts.

However, regardless these differences, the formula shares the same rationale with the one we previously introduced, since it gives a higher score to the

[12] http://www.natcorp.ox.ac.uk/

Chinatown	The Ring	Titanic
cast	actor	story
ending	thriller	love
nicholson	effects	effects
performance	horror	picture
story	character	music

Table 1: Top-5 Aspects returned by the Aspect Ranking component for three different movies.

aspects that are often mentioned in the reviews with a *positive* sentiment. At the end of this step all the aspects are ranked and the *top-K* are labeled as *main aspects*.

To improve the understanding of the approach and the discussion, the *Top-5 Aspects* returned by Aspect Ranking component of the TS-Pipeline are reported in Table 1. In order to strengthen the findings of the approach, we report the output for three different movies.

As shown in the Table, the output emphasizes how our strategy can make relevant and interesting aspects of the items immediately emerge, since different and distinguishing elements of the movies are highlighted. As for the movie *The Ring*, that we used as a toy example in the previous section, we can note that some difference between the Top-5 aspects returned by the different implementations of the Aspect Extraction and Aspect Ranking modules actually exist. This difference is mainly due to the exploitation of the Kullback-Leibler divergence as relevance factor in the formula.

3.2.3 Generation

The implementation of the Generation phase represents the main difference that exists between the NLP-Pipeline and the TS-Pipeline.

As previously stated, in this implementation we aimed at avoiding the *static nature* of the template-based justifications that are used in the NLP-Pipeline, thus we have conceived the justification as a *summary* of the most relevant and distinguishing aspects of the item. To this end, we exploited *text summarization techniques* to automatically build a summary of the most relevant aspects discussed in the reviews, which is presented to the user as *justification* of the recommendation she received

In order to produce a suitable *summary*, the text summarization algorithm needs to be feed with a set of sentences. However, as we already pointed out in Section 3.1.3, some of the available sentences are not suitable for our scopes, that is to say, to generate a snapshot that highlights the main information encoded in the reviews.

Accordingly, also in the TS-Pipeline we run a sentence filtering phase whose goal is to filter out *non-compliant* sentences that are supposed to be not useful for the final justification we want to build. In particular, we used

the same criterion previously described to filter out non-compliant sentences from the original set.

At the end of the *sentence filtering* a set of potential *candidate sentences* is obtained. Such a set is used to feed a *text summarization* algorithm whose goal is to generate a unique summary to be used as justification of the recommendation. Such a summary is supposed to highlight the main contents discussed in the reviews of the item and to maximize both the *coverage* and *diversity* of the sentences to be included in the justifications. To this end, our strategy selects sentences which cover enough amount of the topics discussed in the original reviews and tries to avoid the redundancy as well.

To obtain a *summary* of the available reviews we adapted the method described in [44], which proved to be effective in a multi-document summarization task. This approach is very suitable for our goals, since each *review* can be easily considered as a *document*, thus the method can be used to summarize all the documents (that is to say, all the *reviews excerpts*) in a single *summary* that highlights the most salient features of the item.

Our approach combines centroid-based text summarization [43], which has the advantage of being unsupervised, with a pre-trained neural language model, such as Word2Vec [33], which is good in transferring information from web-scale textual corpora, and is based on two steps: first, all the information coming from the reviews of an item are condensed in a *centroid vector* which represents a pseudo-review. Next, the main idea is to project both the centroid and each sentence of the reviews in a vector space and to include in the summary only the sentences closer to the centroid.

Formally, given a set of reviews R and its vocabulary V with size $N = |V|$, we first define a matrix $M \in R^{N,k}$, so-called *lookup table*, where the i-th row is a word embedding of size k, $k << N$, of the i-th word in V. The values of the word embeddings matrix M are learned by using Word2Vec. When the lookup table is learned, the summarization method consists of three phases:

Centroid Vector Building. The centroid vector that represents the *meaning* of the reviews is built in two steps. First, the most meaningful words occurring in the reviews (that is to say, those having their $tf * idf$ weight greater than a topic threshold) are selected. Next, the embedding of the centroid is computed as the sum of the embeddings of the top ranked words in the reviews using the lookup table M.

$$C = \sum_{w \in R, tfidf(w) > t} M[idx(w)] \tag{4}$$

In the equation (4) we denote with C the centroid embedding related to the set of reviews R and with $idx(w)$ a function that returns the index of the word w in the vocabulary.

Sentence Scoring. Given the centroid vector, we need to identify the sentences to be included in the summary. For each sentence in the set of the reviews, we create an embedding representation by summing the vectors for

Movie	Justification
Chinatown	*This movie has a decent plot and great acting by Jack Nicholson. Unlike most noir films Chinatown has a great script with fantastic characters, a whirling, looping (but always believable) plot, and one of the best endings of any Hollywood film of any period.*
The Ring	*If you like or love the blood and gore kinds of films, this movie will certainly disappoint you as the focus is on character, story, mood and unique special effects. The Ring is a story about supernatural evil therefore, it is a horror film, done very much in the style of the psychological thriller.*
Titanic	*The highest grossing movie of all time, James Cameron's Titanic follows the love story of Jack and Rose set on the doomed 1912 maiden voyage of the Titanic. This is the greatest love story of our time and a very good drama with great special effects.*

Table 2: Example of justifications generated through the TS-Pipeline for the Movie domain.

each word in the sentence stored in the lookup table M.

$$S_j = \sum_{w \in S_j} M[idx(w)] \tag{5}$$

In the equation (5) we denote with S_j the j-th sentence in the set of reviews R. Then, the sentence score is computed as the cosine similarity between the embedding of the sentence S_j and that of the centroid C of the set R.

$$sim(C, S_j) = \frac{C^T \bullet S_j}{||C|| \cdot ||S_j||} \tag{6}$$

Sentence Selection. The sentences are sorted in descending order of their similarity scores. The top ranked sentences are iteratively selected and added to the summary until the limit, in terms of the number of words in the summary, is reached. In order to minimize the redundancy of the information included in the summary, during the iteration we compute the cosine similarity between the next sentence and each one already in the summary. We discard the incoming sentence if the similarity value is greater than a threshold.

At the end of this process a *summary* is produced. An example of the summaries generated for three movies is reported in Table 2. As previously stated, these summaries are presented to the users as justifications. By comparing the output obtained through the TS-Pipeline to that obtained through the NLP-Pipeline it immediately emerges that the structure of the justifications is more diverse and less static, thus our implementation successfully overcame one of the issues we noted in the justifications produced through the NLP-Pipeline. In the experimental session we will investigate whether such a more complex and diverse structure positively impacts on the users as well.

3.3 Context-aware Justifications based on Automatic Lexicon Generation (Context-Pipeline)

The Context-Pipeline implementation aims to tackle another limitation of the NLP-Pipeline[13], that it to say, the inability to diversify the justification on the ground of the different *contextual situations*.

This idea took its roots in the area of context-aware computing [45] and, in particular, of context-aware recommender systems [1]. According to the literature in the area, the context can be defined as a *set of factors that describe the current situation and can potentially influence the decision-making process*. As an example, in a restaurant recommendation scenario the contextual factors may be the *company* (family, partner, etc.), the *meal* (lunch, dinner, breakfast), the *day of the week* (weekday, weekend) and so on.

In a nutshell, our approach is based on a simple *intuition*: as the behavior of recommendation algorithms should vary on the ground of the *different contextual settings*, so a pipeline to generate justifications supporting the recommendations should do. As an example, a justification that aims to convince a user to enjoy a restaurant for a *romantic dinner* should contain different concepts and aspects with respect to a justification that wants to explain why a certain restaurant recommended for a *family lunch on Sunday* is a good pick.

In the following, we will show how we evolved the NLP-Pipeline in order to make it able to diversify its behavior on varying of the different contextual settings. In particular, our approach is based on the idea of automatically learning a *lexicon* of relevant aspects for each contextual situation, and to exploit such a lexicon to influence the ranking of the aspects and the generation of the justifications.

3.3.1 Aspect Extraction

As it already happened for both the NLP-Pipeline and TS-Pipeline, the goal of this module is to identify the aspects that are mentioned in the reviews of a particular item.

However, even if the goal of this module does not change, an important design choice here differentiates the strategy we implemented in this variant. Indeed, differently from the previous implementations where only *nouns* were extracted, all the *adjectives* that are mentioned in the reviews are identified and returned in this case.

Formally, given a set of reviews R, we run a POS-tagging algorithm over the reviews r_i that discuss the item i, and we return both the *nouns* and the *adjectives* detected in the reviews. As for the adjectives, we return all the tokens whose POS-tagging labels are equal to JJ, JJS and JJR, that correspond to the simple adjectives and to their comparatives and superlatives.

This design choice can be easily explained, since the *adjectives* can play a key role in the task of catching the characteristics of the different contextual

[13] Actually, this problem is common to almost every approach to generate explanations that has been presented in literature so far.

situations. Indeed, differently from simple nouns such as *service*, *meal*, *location*, the identification and the extraction of adjectives such as *romantic* or *quick* can be very useful to highlights specific aspects that are worth to be included to make a context-aware justification more convincing.

As an example, sentences such as *'a romantic location'* and *'very quick service'* both describe positive aspects of the item, thus they are worth to be included in a justification. However, they have a different importance depending on the specific context of usage, since the first one is very useful to induce the user in enjoying the restaurant for a dinner with her partner, while the second is particularly interesting for a user who is planning to have a dinner with her own family. Accordingly, the strategy we implemented was designed to put high in the rank nouns and adjectives that denote the aspects that are more suitable and interesting in a particular contextual situation.

3.3.2 Aspect Ranking

Once the *nouns* and the *adjectives* detected during the Aspect Extraction phase are returned, the Aspect Ranking component has to identify the most relevant aspects to be included in the justification.

In this case, the main difference between this implementation and the previous one lies in the notion of *contextual relevance*, since this implementation of the module also encodes the *relevance* of a certain aspect in a *particular contextual setting*, in a way that aspects that are particularly relevant in a certain context are provided with higher scores and are used to justify the recommendations.

In order to quantify the relevance of an aspect in a certain context, we automatically learned a *lexicon* for each *contextual setting*. To carry out this task, we went through the available reviews and we *manually annotated*, for each context, a substantial set of sentences that are worth to be used to justify a recommendation *in that specific setting*.

As an example, the previously mentioned sentence *'very romantic location'* is annotated with the contexts *company=partner* and *meal=dinner* while the sentence *'very quick service'* is annotated with the contexts *meal=lunch* and *company=colleagues*. In the first case, the intuition is that the concepts mentioned in the sentence can make a recommendation for a dinner with the partner more convincing, while in the second case the fact that the service is quick can be very useful to decide to enjoy a restaurant recommendation for a lunch with colleagues.

Formally, given a set of reviews R, we split each review r_i in sentences. Next, for each contextual setting c_j we manually annotated with the name of the context a subset of k sentences $s_1 \dots s_k$ that are relevant for that specific setting. Of course, each sentence s_j can be annotated with more than one context. Such an annotation can be exploited to automatically learn a *lexicon of aspects* relevant for that specific context, that is to say, aspects and concepts that are frequently used to discuss the item under a specific contextual setting. Next, this lexicon can be used in the Aspect Ranking phase to emphasize or

Company=Partner	**Company=Family**	**Company=Friends**
romantic	kids	cozy
music	playground	genuine
terrace	clean	lights
cozy	family	space
elegant	children	modern

Table 3: Top-5 Aspects returned by the Aspect Ranking component for the same item, on varying of different contextual settings (specifically, different *company*.

to lighten the weight of each aspect according to its importance for a specific context, thus influencing the generation of the justifications.

Formally, for each aspect a detected in the Aspect Extraction phase and for each contextual setting c, we calculate the *context-aware score* of the aspect as follows:

$$score(a, c, R_i) = (\alpha \frac{rel_{a,R_i}}{|S_{R_i}|} + \beta \frac{pos(a, R_i)}{pos(a, R_i) + neg(a, R_i)} + \gamma \frac{n_{a,c}}{|S_c|}) * IAF(a, R_i) \quad (7)$$

Equation 7 follows the notation already presented in Section 3.1.2, where we discussed the basic NLP-Pipeline. However, this formula introduces a third factor which represents the ratio between $n_{a,c}$, that is to say, the number of sentences mentioning the aspect a annotated with the contextual setting c and $|S_c|$, that is to say, the overall number of sentences annotated for that specific context. Intuitively, the factor is high if the aspect a is typically useful to justify the recommendation enjoyed in the context c.

The output of the Aspect Ranking for the same item over three different contextual settings (specifically, different *company*) is presented in Table 3. As shown in the Table, very different aspects are highlighted for each context. Some concepts (e.g., *cozy*) may be common to different settings, but each contextual situation is characterized by a very specific vocabulary of terms that have to be used to justify the recommendation. At first glance, Table 3 confirms the effectiveness of our methodology since the main aspects of each context correctly catch and emphasize concepts that are particularly relevant in that specific context.

3.3.3 Generation

Once the main aspects that are relevant for a certain contextual setting have been returned, the Generation phase can start. As for the other implementations, the goal of this step is to identify compliant excerpts of the reviews that mention the *main aspects* identified in the Aspect Ranking step.

In this case, we used the template-based structure we implemented in the NLP-Pipeline as a reference model to generate our natural language justifications. Such a choice is due to the fact that our goal is to evaluate the

Restaurant	Justification
Company=Partner	*You should visit restaurant 'Grotta Palazzese'. It is very suitable for a dinner with your partner thanks to its romantic view and its elegant atmosphere.*
Company=Family	*You should visit restaurant 'Grotta Palazzese'. It is very suitable for a dinner with family since it has a nice playground for kids and the ambience is very clean and cozy.*
Company=Friends	*You should visit restaurant 'Grotta Palazzese'. It is perfect for a dinner with your friends thanks to its lights that create a very nice atmosphere. Moreover, food is very nice and genuine.*

Table 4: Example of justifications generated through the CONTEXT-PIPELINE for the RESTAURANT domain.

validity of the intuition of diversifying the justifications on varying of the different contextual settings, rather than evaluating the effectiveness of a more sophisticated strategy to generate the justification[14]

Accordingly, we run the sentence filtering phase to filter out non-compliant excerpts as we did in Section 3.1.3, and we filled in the template with the main aspects and the excerpts. The main difference between this generation phase and the one previously introduced in the NLP-PIPELINE lies in the fact that k different justifications, that is to say, one for each contextual setting, are generated for the same item through this pipeline. An example of the output of the pipeline in the restaurant domain, for three different contextual settings, is reported in Table 4.

As shown by the output reported in the Table, our strategy to generate context-aware justifications successfully carries out the task, since each contextual setting led to a different justification that highlights different aspects that characterize the item, and this is supposed to be helpful for the target user who is going to evaluate the goodness of the recommendation she received.

4 Experimental Evaluation

In order to evaluate the effectiveness of our conceptual model and of the different pipelines we introduced we carried out three user studies in different domains, such as *movies*, *books* and *restaurant* where real users were asked to enjoy the suggestions generated by a recommendation algorithm and to evaluate the justifications generated by the different instances of the methodology.

[14] Actually, one of the experiments discussed in the next section evaluates the impact of more complex generation strategies, that is to say, text summarization techniques. In this case, we decided to use a basic template-based structure instead of a more sophisticated generation in order to have a higher control and understanding on the outcomes of the experiments. Rather than changing two modules, that is to say, Aspect Extraction and Generation, we preferred to maintain the same generation phase of the NLP-PIPELINE and we just changed the strategy for Aspect Extraction and Ranking by making it *context-aware*.

In particular, our experiments aimed to answer to the following research questions:

- **RQ1:** How do reviews-based justifications perform with respect to classical features-based explanations? (User Study 1)
- **RQ2:** Does automatic aspect extraction and automatic text summarization improve the quality of the resulting justifications? (User Study 2)
- **RQ3:** How does context-aware justifications perform with respect to simple non-contextual review-based justifications? (User Study 3)

By referring to the names of the different pipelines we previously introduced, in User Study 1 we compared the NLP-Pipeline to an explanation strategy that did not exploit the information conveyed by users' reviews, in User Study 2 we compared the NLP-Pipeline to the TS-Pipeline and in User Study 3 we compared the NLP-Pipeline to the Context-Pipeline.

In the following sections, we separately discuss the experimental design of each user study, we will provide the results of the experiment, and we will discuss the outcomes emerging from each session.

4.1 User Study 1: reviews-based justifications vs. feature-based explanations

In order to answer to **RQ1**, we designed a user study involving 286 subjects (male=76.3%, degree or PhD=48.4%, already used a RS=91.0%) in two different domains, that is to say *movies* and *books*.

Interest in *movies* and *books* was indicated as *medium* or *high* by 89.46% and 62.78% of the sample, respectively. Our sample was obtained through the *availability sampling* strategy, and it includes students, researchers in the area and people not skilled with computer science and recommender systems. As in [50], we evaluated the following metrics: *transparency*, *persuasiveness*, *engagement*, *trust* and *effectiveness*.

4.1.1 Experimental Protocol

To run the experiment, we deployed a web application[15] implementing the NLP-Pipeline we introduced in Section 3.1

Experimental Design. Before comparing our NLP-Pipeline to a feature-based explanation based we run a *preliminary experiment* aiming at identifying the best-performing configuration of the NLP-Pipeline, on varying of different configuration of the components.

Such an experiment was run in a *between-subject* fashion, that is to say, each user was randomly assigned to a configuration of our pipeline, and she evaluated the justifications for both the domains. The order the domains were presented to the user was randomized. Clearly, the user was not aware of the specific configuration he was interacting with.

[15] https://tinyurl.com/review-based-justifications

	Movies	Books
#items	307	333
#reviews	153,566	52,560
avg. reviews/item	500.21	157.80
avg. words/review	138.38	126.55

Table 5: Statistics about the Movie and Books datasets

Next, we compared the best-performing configuration to the baseline to a feature-based explanation that does not exploit the information conveyed by a users' reviews through a *within-subject* experiment, that is to say, all the users were provided with two different explanation styles (*i.e.*, review-based justifications and the explanation built through ExpLOD [37]), and we asked the users to evaluate both of them. It is worth to note that we did not evaluate any other baseline since in our previous work [38] we already showed that our framework perform better than other explanation styles.

Data Acquisition and Data Mapping. In order to run our NLP-Pipeline, we carried out the following three steps: (1) First, we needed a dataset of items to be recommended, thus we gathered a subset of the well-known MovieLens and BookCrossing datasets for the movie and book domain, respectively[16]. Such a subset contains only the items mapped to DBpedia [2]. It is worth to note that we limited our data to the items available in DBpedia since the baseline we compared to relied on the descriptive features available in the Linked Open Data cloud. (2) Next, we needed some textual reviews discussing the items, thus we collected a set of Amazon reviews about movies and books[17]. (3) Finally, we carried out a *manual mapping* of the movies and books available in the review dataset with those available in the recommendation datasets.

At the end of the mapping process, run by matching the name of the items with the ASIN number[18] available in the review database, we obtained a set of *307 movies* and *333 books* that we used to run the experiments. Some statistics about the datasets is provided in Table 5. To guarantee the reproducibility of the experiments, we made available the mapping of the data and the reviews we used in the experiments[19].

Parameters of the Experiment. In order to evaluate different configurations of our NLP-pipeline, we compared four different alternatives of the implementation, obtained by varying *Justification Vocabulary* and *Justification Length*.

As regards the length, we compared *long* and *short* justifications, depending on the number of aspects that were returned by the Aspect Ranking

[16] http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/

[17] http://jmcauley.ucsd.edu/data/amazon/links.html - Only the reviews available in the 'Movies and TV' and 'Books' categories were downloaded.

[18] Acronym for Amazon Standard Identification Number - https://en.wikipedia.org/wiki/Amazon_Standard_Identification_Number

[19] https://tinyurl.com/review-data-uniba

component. Short justification contained *two* aspects while long justifications contained *four* aspects. These values were set in an empirical fashion. We also took into consideration having a higher number of aspects, but an empirical and preliminary analysis showed that a higher number of aspects returned very long and unsatisfying justifications, thus we did not evaluate further configurations. As regards the vocabulary, we evaluated both *static* and *complete* vocabularies, depending on whether a fixed and predefined list of aspects was used or not. Static lists were based on 50 static aspects, that were defined by using domain knowledge and by exploiting design choices of previous research in the area. It is worth to note that when configurations based on fixed aspects were exploited, we did not need to run the Aspect Extraction. Conversely, we only executed the Aspect Ranking by using as input the pre-defined set of aspects and we used the top-K aspects to generate the justification. Parameters α and β in Formula (1) were set to 0.5 after a rough tuning. To sum up, the combination of the values for *justification vocabulary* and *justification length* led to the four configurations we compared in the experiments.

Implementation details. Recommendations were generated by using Personalized PageRank (PR) [21] as recommendation algorithm, as in our previous work [37]. To implement the recommendation algorithms we exploited Jung framework[20] and PR was run by using the default settings (80% of the weight distributed on the nodes the user liked). For the Aspect Extraction and the Aspect Ranking modules, we exploited the algorithms available in CoreNLP[21]. To identify the sentiment conveyed by the single sentences, we used the Stanford Sentiment Analysis algorithm[22]. As regards the Generation phase, we only took into account sentences whose sentiment was labeled as *positive* or *very positive*.

In a nutshell, each user involved in the experiment carried out the following steps:

(1) Collection of Demographic Data. First, we asked the users to provide some basic demographic data and to indicate their interest in *movies* and *books* domains.

(2) Preference Elicitation, Generation of the Recommendation and Generation of the Justification. To gather user preferences, we asked users to explicitly rate at least three items for each domain, chosen among a randomly generated subset of 20 movies and 20 books extracted from the datasets. Once the profiles were built, recommendations were generated by running the recommendation algorithm. We used the preferences of the users and the top-1 recommendation to feed our framework.

(3) Between-subject Evaluation through Questionnaires. Next, we asked the users to fill in a questionnaire to evaluate the quality of the justification. Each user was asked to evaluate the previously presented *metrics* through a five-point scale (1=strongly disagree, 5=strongly agree) and to evaluate how

[20] http://jung.sourceforge.net/
[21] https://stanfordnlp.github.io/CoreNLP/
[22] https://nlp.stanford.edu/sentiment/

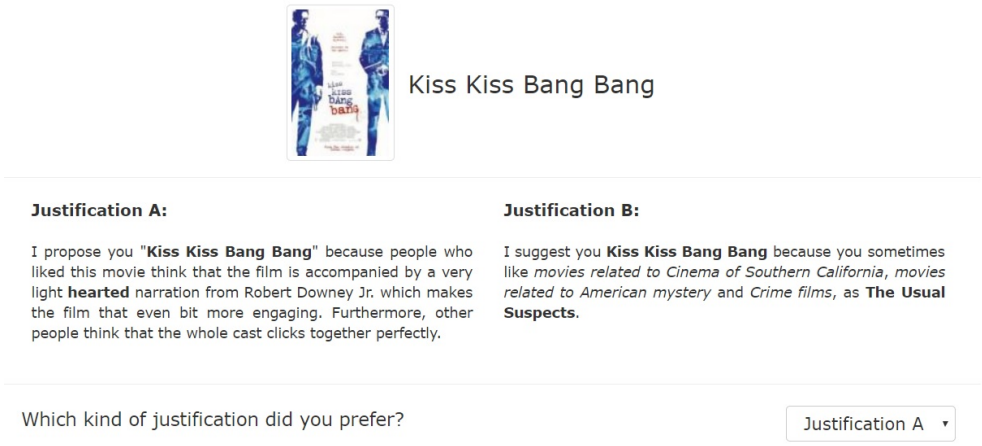

Fig. 3: Screenshot of the platform during the within-subject part of the experiment

much she liked that suggestion. The questions the users had to answer are presented in Table 6. Finally, in the last part of the experiment, each user had to enjoy a trailer of the movie and an excerpt of the book, and had to evaluate again the movie and the book after watching the trailer and reading the excerpt.

(4) Within-subject Evaluation through Questionnaires. Finally, we asked each user to evaluate the explanation style they preferred between our review-based methodology and a feature-based explanation built through ExpLOD. As shown in Figure 3, we provided the user with both the explanations in the same screen, and we asked them to select the best one in terms of transparency, persuasion, engagement, trust and effectiveness. The whole experiment took less than 5 minutes[23].

Evaluation Metrics. We evaluated *transparency*, *persuasiveness*, *engagement* and *trust* of the recommendation as the average score collected through the user questionnaires, while the *effectiveness* was calculated as the normalized difference between the pre- and post-trailer ratings and pre- and post-excerpt ratings the user provided for the recommendation.

It is worth to note that we asked the user to evaluate the *quality of the recommendation* by just considering the information available in the justification. In this case, we followed the evaluation protocol proposed by Bilgic et al. [6], whose insight is that an *effective* explanation may help the user to evaluate to what extent she would like the recommendation, even before enjoying it.

For each configuration of the pipeline more than 70 observations were collected. In our setting, an *observation* is the interaction of a user with the system. Given that in [28] the minimum acceptable sample size for each exper-

[23] The platform is available online and the experimental protocol can be still run.

aim	**question**
transparency (TRA)	"I understood why this item was recommended to me"
persuasion (PER)	"The justification made the recommendation more convincing"
*engagement (*ENG*)*	"The justification helped me discover new information about this item"
*trust (*TRU*)*	"The justification increased my trust in the recommender system"
*effectiveness (*EFF*)*	"I like this recommendation"

Table 6: Details of the Questionnaire

Configuration		**Metrics**				
Aspects	*Length*	TRA	PER	ENG	TRU	EFF
Complete	Short	**3.91**	3.60	3.25	3.70	0.53
Complete	Long	3.74	3.48	3.35	3.46	0.59
Static	Short	3.40	3.13	3.09	3.23	0.64
Static	Long	3.77	**3.68**	**3.55**	**3.73**	**0.55**

Table 7: Comparison of different variant of the NLP-Pipeline for the Movie domain. The best-performing configuration for each metric is reported in **bold**.

Configuration		**Metrics**				
Aspects	*Length*	TRA	PER	ENG	TRU	EFF
Complete	Short	**3.80**	3.64	3.40	3.56	**0.42**
Complete	Long	3.60	3.37	3.31	3.32	0.58
Static	Short	3.57	3.41	3.07	3.32	0.76
Static	Long	3.64	**3.82**	**3.45**	**3.71**	0.45

Table 8: Comparison of different variant of the NLP-Pipeline for the Book domain. The best-performing configuration for each metric is reported in **bold**.

imental condition was set as 73, we can state that our experiment guaranteed the significance of the results.

4.1.2 Discussions of the Results

The results we obtained for the preliminary experiment comparing the different configurations of the NLP-Pipeline are reported in Table 7 and Table 8 for the Movie and Book domains, respectively. Each line of the tables represents a different configuration of the pipeline, while the values represent the average scores given by the users for that specific metric. The higher the better, except the effectiveness where the best configuration is the one closer to zero.

The first result emerging from the experiment is that *longer justifications based on static list of aspects* obtained the best results for most of the metrics. However, in many comparisons, short justifications based on the complete list of aspects are the second best configuration and obtained similar results,

with a very tiny and often not significant gap. This is an interesting finding, that leads to the main outcome of this first experiment: *when we generate our justifications by exploiting only fixed list of (relevant) aspects, users tended to prefer long justifications to short ones.* This is confirmed for all the metrics we took into account. Thus, it is likely that giving more information to the user, established that the aspects provided in the justifications are bounded to relevant characteristics, leads to a more transparent, convincing and persuading output.

Conversely, when the justifications include *all* the available aspects the algorithm is able to discover, shorter justifications tend to be preferred. In this case, our conjecture is that the absence of any filter leads to a strategy that may introduce some noise, especially when a longer list of aspects is returned. This can make the output less convincing and satisfying.

However, it is worth to note that the *Complete* configuration obtained the best results for the TRANSPARENCY. In this case, it is likely that the inclusion of a larger set of characteristics can provide the user with a wider and clear justifications of the behavior of the recommender systems.

The same findings also hold for the BOOK domain, since *shorter* justifications are preferred when the *complete* set of aspects is taken into account, while *longer* justifications obtained the best results with the *static* list of relevant aspects. It is worth to note that also for the BOOK domain the most effective justifications are those obtained without limiting the vocabulary of the aspects. Overall, we can state that the consistency of the outcomes in both the domains is an interesting and important results, that further confirms the strength and the soundness of our methodology.

Finally, it is important to state that these results also confirmed the general effectiveness of our pipeline. Indeed, scores higher than 3.00 (equivalent to *'partially agree'* as answer) were obtained for all the metrics. Even if this may emerge as a secondary outcome, it is a very interesting and important result since the idea of combining reviews excerpts in a natural language justification supporting a recommendation through a fully automated pipeline was a poorly investigated research line, thus the overall effectiveness of the justifications that are generated is not a trivial outcome and it is worth to be underlined.

Next, we compared the best-performing configurations emerged from the preliminary experiment to the explanations generated through ExpLOD [37] in a *within-subject* experiment. Specifically, we used short justifications based on complete aspects for *transparency* and *effectiveness*, and long justifications based on static aspects for *persuasion, engagement and trust.*

As shown in Table 9 and Table 10, most of the users indicated that they preferred our review-based methodology for building natural language justifications. This finding is confirmed for both the domains and for all the metrics we took into account.

It is worth to note that the best results for the MOVIE domain were obtained for the *engagement.* This means that the users asserted that the exploitation of reviews data can help discover new information about the recommended item. This was an expected outcome, since feature-based explanations

Movies	NLP-Pipeline	ExpLOD	Indifferent
Transparency	**47.4%**	38.6%	14.0%
Persuasion	**51.7%**	43.3%	5.0%
Engagement	**66.7%**	25.0%	8.3%
Trust	**53.3%**	35.5%	11.7%
Effectiveness	**57.9%**	35.0%	7.1%

Table 9: Comparison of the NLP-Pipeline to a feature-based baseline in the Books domain. The configuration preferred by the higher percentage of users is reported in **bold**.

Books	Review-based	ExpLOD	Indifferent
Transparency	**58.1%**	36.0%	5.9%
Persuasion	**61.8%**	29.0%	9.2%
Engagement	**54.6%**	27.3%	18.1%
Trust	**58.2%**	27.2%	14.6%
Effectiveness	**59.9%**	31.1%	10.0%

Table 10: Comparison of the NLP-Pipeline to a feature-based baseline in the Movie domain. The configuration preferred by the higher percentage of users is reported in bold.

usually provide effective justifications but they typically rely on very popular and well-known characteristics of the movie, as the actors or the director. Conversely, through the automatic analysis of users' reviews more particular and specific aspects of the movie that impressed the user can emerge, and this can let the users discover new information and lead them to a more informed decision about the quality of the recommendations. Similar outcomes were noted for the Book domain, where the review-based justifications were preferred again by the users for all the metrics, with a huge gap with content-based explanation for all the choices.

These outcomes definitely confirmed the effectiveness of our intuitions and showed that the exploitation of the information conveyed by users' reviews through the extraction of relevant and distinguishing aspects of the items can provide users with effective and transparent justifications.

4.2 User Study 2: reviews-based justifications vs. automatic text summarization

The second user study was designed to answer to **RQ2**, that is to say, to evaluate to what extent the exploitation of more sophisticated methodologies for aspect extraction and generation of the justification, as those we implemented in the TS-Pipeline, can improve the effectiveness of the basic NLP-Pipeline. To this end, we arranged a second user study involving 141 subjects (male=70.3%, degree or PhD=38.4%, already used a RS=89.0%) in the same domains, that is to say, *movies* and *books*.

As for the first experiment, the sample was obtained through the *availability sampling* strategy and the interest in *movies* and *books* was indicated as *medium* or *high* by a significant portion of the sample (82.11% and 61.32% respectively).

4.2.1 Experimental Protocol

To run the experiment, we deployed another web application[24] that implements the TS-Pipeline described in Section 3.2. Given that the user study was run in the same domains, we used the same data about *movies* and *books* we described in Section 4.1.

As for User Study 1, we first run a preliminary experiment aiming at identifying the best-performing configuration of the TS-Pipeline, then we compared the best configuration emerged from the preliminary experiment to the NLP-Pipeline we previously discussed. Also the experimental design followed the organization of User Study 1 , since the first part of the experiment was run in a *between-subject* fashion, while to compare TS-Pipeline to NLP-Pipeline we run a *within-subject* experiment.

Parameters of the Experiment. Throughout the second user study we compared four different alternatives of the TS-Pipeline, obtained by varying *number of aspects* and *justification length*. As for the length, we compared *long* and *short* justifications, depending on the overall length of the summaries we generated (50 words for short justifications, 100 words for long justifications). These values were set in an empirical fashion. As for the number of aspects, we compared the justifications built by including all the available aspects, the *top-10* aspects and the *top-30 aspects*. This means that the *text summarized* is feed with all the sentences that: *(i)* match the criterion we introduced in Section 3.2.3; *(ii)* mention one of the first k aspects ($k = 10, 30$, depending on the setting). When *all* the aspects are taken into account, the summarizer is fed with all the compliant sentences. As previously stated, the text summarizer we exploited in this experiment is based on the algorithm discussed in [44]. The parameters of the centroid-based summarization strategy were tuned by exploiting the results presented in the original article. In particular, we set topic threshold as 0.6 and similarity threshold as 0.95. Topic threshold represents the minimum *tf-idf* weight which is required to include a particular word in the calculation of the centroid vector of the item, while similarity threshold represents the minimum cosine similarity score which is required for a candidate sentence. These values were set after a grid search over candidate values. Word embeddings pre-trained on Google News (*size=300*) and based on skip-gram architecture were used as input of the whole pipeline.

Implementation details. As for the first user study, recommendations were generated by using Personalized PageRank (using the default settings) as recommendation algorithm. Similarly, we used the algorithms available in

[24] https://tinyurl.com/uniba-expl-summarizer

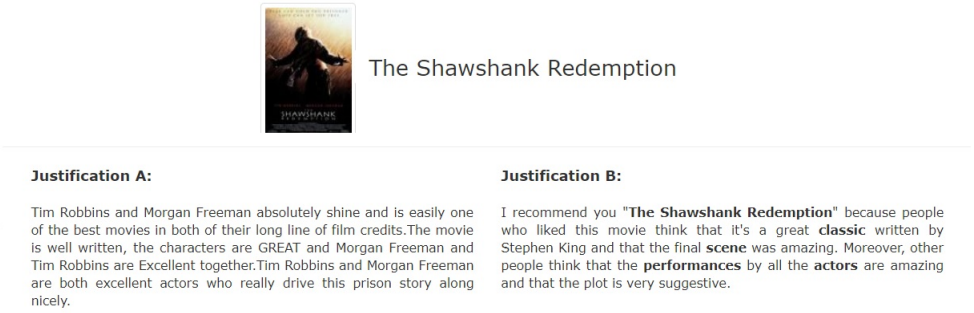

Fig. 4: Screenshot of the platform during the within-subject part of the experiment

CoreNLP and Stanford Sentiment Analysis for Aspect Extraction and Aspect Ranking. As regards the Text Summarizer, pre-trained word embeddings learned through Word2Vec were used. The source code of the summarizer is available online[25].

The experiment protocol followed by each user was the same described for User Study 1, and it is based on: *(i)* a preference elicitation (followed by the generation of the justifications) phase; *(ii)* a between-subject evaluation through questionnaires, identical to that arranged in the first user study and; *(iii)* a within-subject evaluation through questionnaires, aiming at comparing the NLP-Pipeline to the TS-Pipeline. A screenshot of the within-subject portion of the user study is provided in Figure 4.

Finally, as evaluation metrics we used the same ones we exploited in User Study 1, that is to say, *transparency*, *persuasiveness*, *engagement* and *trust* and *effectiveness*.

4.2.2 Discussion of the Results

The results we obtained for the preliminary experiment comparing the different configurations of the TS-Pipeline are reported in Table 11 and Table 12.

As for the Movie domain, the first result emerging from the experiment is that *longer justifications* tend to beat their shorter counterparts. This means that very short justifications cannot convey the information which is needed by the final users to better understand the quality of the suggestions they received. Such a low quality is also confirmed by analyzing the scores we obtained for *short justifications*: they are often below 3.00 (equivalent to *'partially agree'* as answer), which is considered as the minimum score that makes a justification as *acceptable*. Conversely, when longer justifications are produced by our technique, the results we obtained are generally higher. This means that the addition of more sentences can make the justifications and the recommenda-

[25] https://github.com/gaetangate/text-summarizer

Configuration		Metrics				
Aspects	*Length*	TRA	PER	ENG	TRU	EFF
All	Short	2.89	2.74	3.26	2.93	0.48
Top-10	Short	2.83	3.06	3.06	2.83	0.89
Top-30	Short	3.16	3.06	2.69	3.19	0.94
All	Long	3.58	3.46	3.29	3.38	0.45
Top-10	Long	**3.95**	**3.64**	**3.37**	**3.55**	0.55
Top-30	Long	3.24	3.18	3.12	3.22	**0.38**

Table 11: Evaluation of different configuration of the TS-Pipeline in the Movie domain. The best-performing configuration for each metric is reported in **bold**.

Configuration		Metrics				
Aspects	*Length*	TRA	PER	ENG	TRU	EFF
All	Short	3.02	3.01	3.09	2.74	0.93
Top-10	Short	3.04	3.26	2.99	2.94	0.82
Top-30	Short	3.23	3.31	2.79	3.22	0.84
All	Long	3.38	3.44	3.44	3.31	0.75
Top-10	Long	3.49	**3.74**	**3.51**	**3.67**	0.65
Top-30	Long	**3.61**	3.33	3.23	3.40	**0.41**

Table 12: Evaluation of different configuration of the TS-Pipeline in the Books domain. The best-performing configuration for each metric is reported in **bold**.

tion process more satisfying, engaging, transparent and trustful for the target users.

As regards the number of *aspects* to be included, the experiment showed that the best results are obtained by exploiting the *top-10 aspects* identified by our aspect ranking module. In the Movie domain, this is confirmed for all the metrics except of the effectiveness. This means that by selecting a subset of relevant aspects we can provide the user with a summary containing the most relevant and useful information to evaluate the quality of the suggestion. Conversely, when a higher number of aspects is exploited (or even when *all* the aspects are taken into account), some noise is probably introduced in the summarization process, thus non-relevant or non-interesting sentences are put in the final justification, and this leads to a decrease in the overall results.

The outcomes we obtained from the Books domain are in line with those we just discussed, since *longer* summaries based on the *top-10* aspects returned by the Aspect Ranking module led to the best justifications. The only exception is represented by the transparency and effectiveness metrics, where the configuration that exploit the *top-30* aspect resulted as the best-performing. In this case, we can state that the injection of more information can make the justification more transparent and allow the user to quickly perceive to what extent she would like the item.

Movies	TS-Pipeline	NLP-Pipeline	Indifferent
Transparency	**54.55%**	40.91%	4.55%
Persuasion	**77.27%**	13.64%	9.09%
Engagement	**63.63%**	27.27%	9.09%
Trust	**68.18%**	4.55%	27.27%
Effectiveness	**62.30%**	8.77%	28.93%

Table 13: Comparison between TS-Pipeline and NLP-Pipeline in the *movie* domain. The configuration preferred by the higher percentage of users is reported in **bold**.

Books	TS-Pipeline	NLP-Pipeline	Indifferent
Transparency	**55.32%**	41.47%	3.21%
Persuasion	**73.55%**	20.21%	6.24%
Engagement	**68.51%**	22.57%	8.92%
Trust	**64.11%**	24.87%	11.02%
Effectiveness	**66.34%**	21.55%	12.11%

Table 14: Comparison between TS-Pipeline and NLP-Pipeline in the *books* domain. The configuration preferred by the higher percentage of users is reported in **bold**.

Next, we compared the best-performing configuration of our TS-Pipeline, that is to say, *long justifications based on top-10 aspects* to the justifications generated by exploiting the basic NLP-Pipeline in a *within-subject* experiment. Results are reported in Table 13 and 14.

As shown in both the Tables, most of the users indicated that they preferred our methodology based on *automatic text summarization*. It is worth to note that we obtained the higher results for *persuasion* and *engagement* in both the domains. This is a very encouraging outcome that confirmed the intuition behind this approach, since the exploitation of text summarization can help to make interesting information about the recommended item emerge, and this can allow the user to discover new information about the item and can persuade her in enjoying the recommendation.

Conversely, the smallest gap between the different configurations was noted for the *transparency*. This is an expected outcome as well, since it is likely that a simple template-based structure can make easier to identify and immediately understand the relevant and distinguishing aspects of the recommended item. However, even for this metric the TS-Pipeline overcame the NLP-Pipeline and this further confirmed the validity of the approach.

Overall, we can state that the results emerging from these experiments confirmed that the exploitation of more sophisticated methodologies for generating the recommendation and automatically select aspects can improve the overall impact of the justifications on the final users.

Restaurants	**City=Bari**	**City=Turin**
#PLACES	171	757
#REVIEWS	19,978	84,228
AVG. REVIEWS/ITEM	116,83	111,27
AVG. WORDS/REVIEW	64.38	64.13

Table 15: Statistics about the RESTAURANT dataset for the cities of Bari and Turin.

4.3 USER STUDY 3: non-contextual reviews-based justifications vs. context-aware reviews-based justifications

Finally, the goal of the third user study was to compare the novel CONTEXT-PIPELINE to the basic NLP-PIPELINE that does not exploit contextual information.

In this case, we carried out a user study in a different scenario, that is to say, RESTAURANT recommendation. This choice was due to the fact that the identification of the contextual settings in the restaurant domain (*e.g.*, company, kind of meal, etc.) is easier and more straightforward. Moreover, we were also interested in evaluating the effectiveness of the pipeline in a scenario different than movies or book recommendation[26].

Our user study involved 132 users, selected through the *availability sampling strategy*, as usual. The characteristics of the sample were comparable to those we noted for USER STUDY 1 and USER STUDY 2 (male=61.4%, degree or PhD=31.7%, already used a RS=77.1%, interest in food and tourism=92.11%).

4.3.1 Experimental Protocol

To run the experiment, we deployed a web application[27] implementing the CONTEXT-PIPELINE we described in Section 3.3.

Experimental Design. In this user study, we compared our CONTEXT-PIPELINE to the basic NLP-PIPELINE in a *within-subject* fashion, that it to say, we provide the users with both a context-aware justification as well as its non-contextual counterpart, and we asked the users which one they preferred for the different explanation aims.

Data Acquisition and Data Mapping. In order to gather data about restaurants, we exploited the official Yelp APIs[28]. In particular, we decided to focus on two different cities of the Italian territory, that is to say, *Bari* and *Turin*, since most of our experimental sample was living in one of the two cities.

[26] Of course, as future work we will plan to evaluate the effectiveness of context-aware justifications for MOVIE and BOOKS recommendation, as well

[27] https://itfra.github.io/ - available only in Italian

[28] https://www.yelp.com/fusion

Contextual Situation	Contextual Setting
Company	Partner
	Friends
	Family
Meal	Breakfast
	Lunch
	Dinner
Mood	Good
	Bad
Health	Healthy
	Health Problems
Day	Weekday
	Weekend

Table 16: List of the contextual settings used in User Study 3.

Next, in order to build the justifications, a set of reviews discussing each item was needed. To this end, we exploited both Yelp and TripAdvisor API[29]. We decided to also use TripAdvisor in order to increase the number of reviews for each item. As for Yelp, we used the ID of the place we previously obtained to retrieve all the available reviews. Conversely, given that TripAdvisor IDs were not mapped with Yelp IDs, we used the name of the place and the city to carry out a basic *data mapping* that allowed us to retrieve reviews from TripAdvisor, as well. As previously stated, given that most of the sample was italian-speaking, we only extracted reviews written in Italian.

At the end of the process we extracted 19,978 reviews for 171 restaurants located in *Bari* and 84,228 reviews for 757 restaurants located in *Turin*. More statistics about the datasets are available online. In order to make the experiments reproducible, the dataset we used in User Study 3 was made available online[30].

Parameters of the Experiment. The only parameter of the user study is represented by the set of *contextual settings* we adopted in the experiment. In particular, we defined five different contextual situations, that is to say, *company, type of meal, mood of the user, health requirements, day of the week* and 13 contextual settings in total. Overall, the set of the contexts we managed is reported in Table 16.

As we will show in the following, the Context-Pipeline will be able to generate a context-aware justification for each of *single* contextual settings (*e.g.,* Meal=Dinner) and for each *combination* of contextual settings as well (*e.g.,* Meal=Dinner, Company=Friends, Mood=Good).

Implementation Details. Given that the reviews were written in Italian, we could not use CoreNLP as we did in User Study 1 and User Study 2. Conversely, this user study exploits LinguA[31], a state-of-the-art linguistic annotation pipeline which combines rule-based and machine learning algorithms

[29] https://developer-tripadvisor.com/content-api/

[30] https://tinyurl.com/restaurants-review-data-uniba

[31] http://www.italianlp.it/demo/linguistic-annotation-tool/

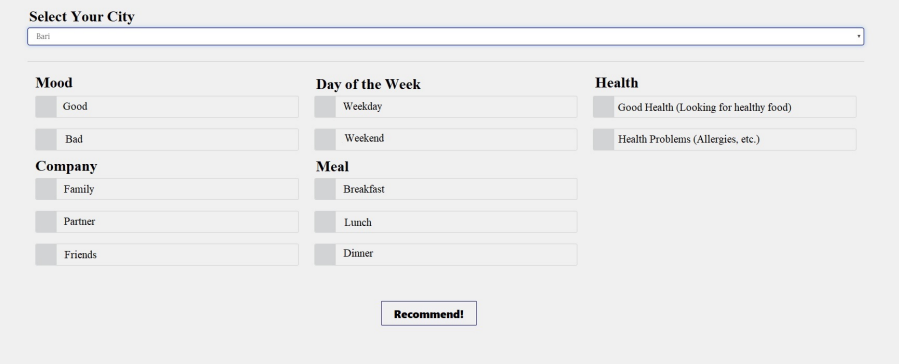

Fig. 5: Selection of the contextual settings for restaurant recommendation. For the sake of readability, the figure reports an English translation of the interface.

that implements tokenization and POS-tagging in Italian. To associate the sentiment to each sentence contained in the reviews, we used SENTIPOLC [5], a tool specifically designed for sentiment analysis in Italian.

As we stated in the previous section, in order to generate our *context-aware lexicons* we manually annotated between 100 and 200 sentences for each contextual setting. Then, the *nouns* and the *adjectives* most frequently used in each contextual setting were used to drive the generation process. Finally, to provide users with recommendations we implemented a content-based strategy based on the descriptive features of the restaurants.

In a nutshell, each user who joined the experiment carried out the following steps:

(1) Collection of Demographic Data. First, we asked the users to provide some basic demographic data and to indicate their interest in the *restaurant* domain through an online questionnaire separated from the platform.

(2) Elicitation of the Contextual Setting, Generation of the Recommendation and Generation of the Justification. As shown in Figure 5, in order to receive restaurant recommendations each user had to select the desired city (*Bari*, in the Figure) and a subset of the available contextual settings. Next, recommendations were generated by looking for the best-matching restaurant for the specific contextual setting. Finally, once the top-1 recommendation was identified, both the *context-aware* justification and the basic non-contextual justification were generated and returned to the target user .

(3) Within-subject Evaluation through Questionnaires. Once the justifications were generated, we asked each user to evaluate the justification style they preferred between the Context-Pipeline and the NLP-Pipeline. A screenshot of this portion of the experiment is shown in Figure 6. As shown in the Figure, the user was not aware of which one was the context-aware justification and which one was built through the NLP-Pipeline.

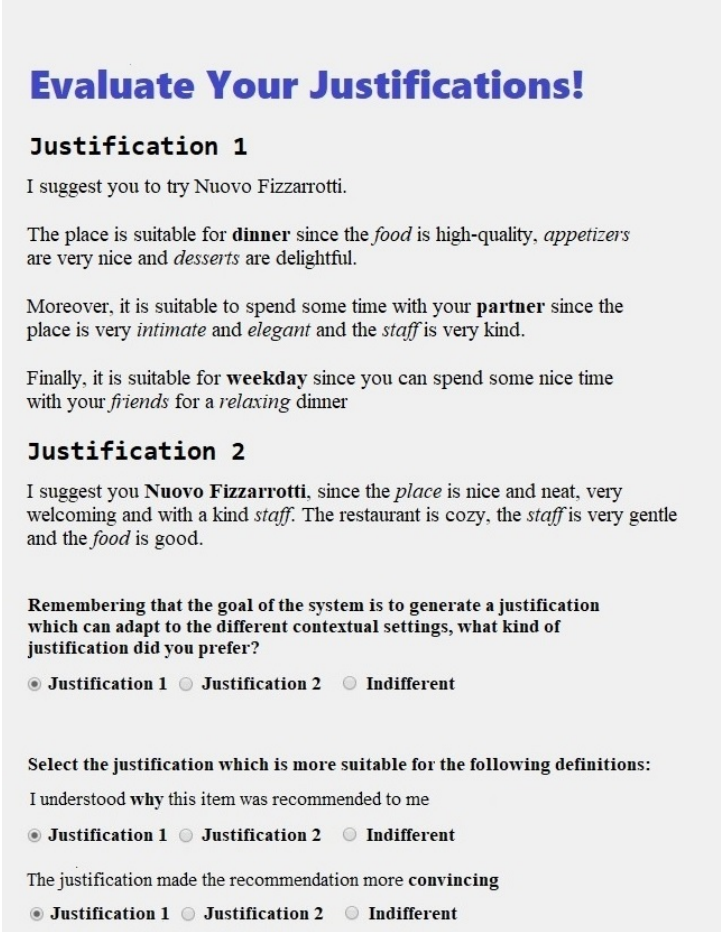

Fig. 6: Screenshot of the platform during the within-subject part of the experiment. For the sake of readability, the figure reports a translation of the interface in English.

As evaluation metrics, we used the same metrics we adopted for User Study 1 and User Study 2, that is to say, *transparency*, *persuasiveness*, *engagement*, *trust* and *effectiveness*. Each metric was evaluated through one specific question formulated in the questionnaire.

4.3.2 Discussion of the Results

The comparison between our Context-Pipeline and the basic NLP-Pipeline is reported in Table 17 and 18. In order to improve the discussion, we split the results on the ground of the number of contextual settings chosen by the users during the user study: specifically, Table 17 reports the results obtained

Contexts=1	Context-Pipeline	NLP-Pipeline	Indifferent
Transparency	27.38%	26.86%	**45.76%**
Persuasion	30.94%	33.69%	**35.37%**
Engagement	23.48%	27.55%	**48,97%**
Trust	19,23%	34.95%	**45,82%**
Effectiveness	26.42%	**53,21%**	20.37%

Table 17: Comparison between Context-Pipeline and NLP-Pipeline when *one* contextual setting is selected. The configuration preferred by the higher percentage of users is reported in **bold**.

when the users selected only *one* contextual setting (*e.g.*, Meal=Breakfast), while Table 18 reports the results when *more than one* context was selected (*e.g.,* Meal=Breakfast and Company=Partner).

Our expectation is that the results should differ *little* when just one context is selected by the user. Indeed, given that the Generation phase is implemented by following a static template-based structure for both the NLP-Pipeline and Context-Pipeline, it is likely that the resulting justifications may be similar and the user does not perceive any significant difference between the output returned by the pipelines.

Conversely, when more than one context is selected, we expect that an *higher gap* between the configurations emerge since the Context-Pipeline should be able to catch the characteristics of the different contextual settings and should generate a richer and more comprehensive justification accordingly.

Such an intuition was confirmed by the results we obtained, since the Context-Pipeline was preferred by the majority of the sample *especially* when *more than one context* was selected. Indeed, as shown in Table 17, under simple contextual settings the resulting justifications are perceived as almost equivalent by the target user (*'indifferent' is the most selected option, with the exception of the Effectiveness).*

Conversely, as the number of contexts increases, our *context-aware justifications* can provide the users with a more detailed justification that emphasize the aspects that are relevant and distinguishing *for each specific contextual setting.* Such a richer and more satisfying structure led the majority of the user select our context-aware justifications. As shown in Table 18, our Context-Pipeline was selected by a large percentage of the sample, with an increase of around 50 percentage point for each justification aim.

These results were particularly interesting for our goals, since they proved that our intuition of diversifying and extending the justifications by identifying the aspects that are most interesting for each specific contextual setting can lead to very satisfying justifications, and this paves the way for further investigations in the area of *context-aware justifications.*

Contexts>1	Context-Pipeline	NLP-Pipeline	Indifferent
Transparency	**82.76%** (+55.38)	11.35%	5.89 %
Persuasion	**81.12%** (+50.18)	12.84%	6.04%
Engagement	**73.52%** (+50.04%)	12.32%	14.16%
Trust	**74.60%** (+55.37%)	11.57%	13,83%
Effectiveness	**86.19%** (+59.77)	7.32%	6,49%

Table 18: Comparison between Context-Pipeline and NLP-Pipeline when *more than one* contextual setting is selected. The configuration preferred by the higher percentage of users is reported in **bold**. The improvement we gain when a larger number of contexts is selected is reported between parentheses.

Configuration	**Strengths**	**Weaknesses**
NLP-Pipeline	- Easy to run; - Basic but effective.	- Justifications can result as too static.
TS-Pipeline	- Diversifies the justifications; - More solid extraction of the aspects.	- Decrease in transparency; - Effort in sentence filtering.
Context-Pipeline	- Adaptation to the context; - Introduces adjectives as aspects.	- Requires manual annotation; - Generation still simple.

Table 19: Strengths and Weaknesses of the different pipelines.

5 Discussion and Limitations

In this article, we have presented three different implementations of our general methodology to generate natural language justifications. The implementations were gradually more sophisticated, and evolved the basic NLP-Pipeline by introducing a more complex strategy to generate the justifications that exploited automatic text summarization (TS-Pipeline) and by taking into account the different contextual situations in which a recommendation can be consumed to generate justifications (Context-Pipeline).

As shown in the experiments we previously presented, all the implementations satisfied the users since all the different justifications generated through our methods had a positive impact. However, the research is still far from being considered as *complete*, and there is a lot of room for improvements since all the approaches we introduced have specific weaknesses that need to be addressed in future research. The main strengths and weaknesses of each implementation are reported in Table 19.

As we stated throughout the article, we NLP-Pipeline provides a good compromise between effectiveness and simplicity since it is very easy to run and can generate good justifications, as proved by the first user study where the output of this pipeline was preferred to the classical content-based explanation that rely on descriptive features. However, the extraction and generation strategies are too static, since they are based on a very fixed template that can poorly diversify the resulting justifications. In a long-term usage of the methodology, this can result as poorly satisfying for the user.

The TS-Pipeline successfully tackles this aspect, since it diversifies the generation phase by introducing automatic text summarization techniques. However, such a choice leads to a decrease in the overall transparency of the generation process since it is carried out by an external component, namely, an automatic text summarizer, whose behavior can be sometimes somewhat *obscure* and poorly interpretable. Moreover, when this pipeline is selected, it is necessary a huge effort to filter out non compliant sentences. Indeed, by feeding the generation module with sentences what are not compliant it is very likely that the resulting justification can be poorly understandable and badly structured.

Finally, the Context-Pipeline has the great advantage of taking into account the notion of *context*, which is a completely novel intuition in the area of explanation. However, such a strategy has two relevant weaknesses: first, it inherits the simplicity of the template-based structure of the NLP-Pipeline. Next, it requires some manual effort to annotate the compliant sentences that are necessary to generate the contextual lexicons.

Our methodology has the main advantage of being completely independent of the underlying recommendation model. In our opinion, to have an algorithm-independent framework that builds a justification which is just based on users' reviews provides several advantages. Indeed, it is immune to cold-start problems since neither a profile of the user nor a minimum amount of users and ratings is necessary to run the framework. In our opinion, this is strong point of the strategy that makes it particularly useful to provide users with transparent recommendations.

Moreover, another interesting aspect that follows the development of this strategy lies in the fact that *negative arguments* can be also generated for the same item. Indeed, by following the same pipeline, reviews can be processed to make *negative* aspects emerge, as well. Such negative aspects may be combined to *positive* and distinguishing aspects of the item in order to provide *fairness* to the whole recommendation pipeline, since a more objective and less biased justification that combines *both* positive and negative aspects is presented. This leaves space for future extensions and investigations on the framework, since new user studies that allow to evaluate the impact of less biased and more fair justification on the user may be easily designed.

However, the flexibility and the soundness of the methodology also conceals some interesting drawback: as an example, an unethical company may use such an approach to promote items she wishes to sell by just picking positive aspects discussed in the reviews. Currently, there is no strategy to effectively handle this problem. This (potential) issue is worth to be investigated in a future experiment, where we can select items from the tail of the recommendation list and we can check whether such a justification can induce a user in enjoying or selecting an item even if it is not the most suitable one for her.

To conclude, the experiments showed the all the pipelines can be exploited to support the users with natural language justifications. The preference towards a particular pipeline is a precise design choice: as an example, if the context matters and it is particularly relevant to influence the decision-making

process, the CONTEXT-PIPELINE will be exploited. Similarly, if the generation of a more diversified justification (maybe at the expense of some transparency) is a primary requirement, the TS-PIPELINE can be used. Alternatively, the NLP-PIPELINE can be also a good pick as well, since it can build simple but effective justifications with a small effort.

Regardless the specific characteristics of each pipeline, it is worth to note that the justifications generated through our methodology share a common weakness, that is to say, they are *not personalized*. This means that all the users will receive the *same* justification for the *same* item. This is a limitation of the current approach that needs to be tackled, since it is likely that the inclusion of the preferences of the target user can lead to more targeted and tailored justifications. Concretely, such an extension could be obtained by introducing in the scoring formulas we previously discussed a *personalization factor* that provides a higher score to the *aspects* that are particularly relevant for the user. As an example, if target user in interested in movies having great plots, excerpts that praise the plot of the recommendations will be likely to be introduced in the justifications.

However, the choice of designing a non-personalized strategy to generated justification also brings benefits, since all the justifications can be computed *off-line* and can be generated once for all the users. This makes the methodology very suitable to be integrated in real time environment and scenarios, since no significant computation is needed.

Moreover, another limitation of the current approach regards the values of the parameters we used in the equations implemented in the ASPECT RANKING phase, since they are all equally set in an empirical fashion. In this case, it is likely that automatic tuning of the parameters may further improve the quality of the justifications. Finally, some research effort needs to be devoted to identification more complex aspects, such as bi-grams. This can be particularly interesting for the CONTEXT-PIPELINE since the identification of sequences of adjectives and nouns (*e.g.,* romantic dinner) can be particularly interesting to highlight relevant aspects.

Finally, we want to further emphasize the novelty of the proposed methodology with respect to the relevant literature in the area of review-based justifications:

- Differently from [13], we did not bound the potential aspects to a predefined and fixed set. Conversely, we let the algorithm identify the most relevant aspects by itself, and this is likely to lead to more differentiated and heterogeneous justifications that rely on a potentially larger set of aspects.
- Differently from [35], who just showed to the user the main characteristics of the item emerging from users' reviews, we generate a justification based on natural language templates which is filled in with reviews excerpts. Even if the extraction and identification process follow two similar strategies, our approach aims to build a more comprehensive justification since the

adoption of natural language makes the output closer to the way people justify and explain their suggestions to other people.
- Differently from our previous research [37,38], we built a natural language justification by exploiting users' reviews instead of looking for overlapping properties that describe both the recommendation and the user profile. In this case, our intuition is that our new strategy generates a justification which may result as less trivial than that built by using our previous framework (*e.g.*, you should watch '300' since you liked other movies shot by the same director).
- Differently from [9], we generate a natural language justification that completely relies on natural language processing and sentiment analysis techniques, without involving crowd-workers and crowd-sourcing platforms.
- The exploitation of *text summarization* techniques to generate justifications and the generation of *context-aware justifications* can be considered as totally *novel* for the area. Indeed, to the best of our knowledge, there is no literature investigating neither the effectiveness of text summarization techniques in this specific task nor deepening the intuition of diversifying the justifications on varying of different contextual settings.

6 Conclusions and Future Work

In this paper, we presented a framework to build *post-hoc natural language justifications* supporting the suggestions generated by a recommendation algorithm, and we introduced three different implementations of the methodology. The first one (NLP-Pipeline) exploits natural language processing and sentiment analysis techniques to identify relevant and distinguishing aspects of the items that are often discussed in users' reviews, and combines compliant reviews' excerpts in a natural language justification. The second implementation (TS-Pipeline) tries to improve the effectiveness of the justifications by introducing a method for automatic aspect extraction based on the Kullback-Leibler divergence. Moreover, we also exploited automatic text summarization to build more diverse and satisfying justifications. Finally, we tried to tackle the problem of diversifying justification on the ground of different contextual settings. To this end, in the Context-Pipeline we automatically learned a lexicon for each context and we used such a lexicon to adapt the justifications to the different contextual situations.

In the experimental evaluation we carried out three user studies that provided us with interesting findings. In particular we noted that: *(i)* reviews-based justifications were preferred to simple features-based explanations; *(ii)* automatic text summarization can lead to more satisfying justifications; *(iii)* the users liked the idea of diversifying justifications on the ground of the different contextual settings, since context-aware justifications were chosen by a vast majority of the sample.

As future work we will first make our justifications as *personalized*, that is to say, adapted to the different interests and preferences of the target user.

Moreover, we will try to compare our justifications to those generated through recent recommendation methods, such those based on deep architectures [47] or methods based on users' reviews that jointly learn to recommend and explain [31]. Finally, thanks to the modular structure of our pipelines different and more sophisticated implementations for the different components will be took into account. As an example, we may evaluate the extraction of bi-grams to further improve our justifications.

References

1. Adomavicius, G., Tuzhilin, A.: Context-aware Recommender Systems. In: Recommender systems handbook, pp. 217–253. Springer (2011)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. Springer (2007)
3. Balazs, J.A., Velásquez, J.D.: Opinion Mining and Information Fusion: a Survey. Information Fusion **27**, 95–110 (2016)
4. Baral, R., Zhu, X., Iyengar, S., Li, T.: ReEL: Review-Aware Explanation of Location Recommendation. In: Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, pp. 23–32. ACM (2018)
5. Basile, P., Novielli, N.: Uniba: Sentiment analysis of english tweets combining microblogging, lexicon and semantic features. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pp. 595–600 (2015)
6. Bilgic, M., Mooney, R.J.: Explaining Recommendations: Satisfaction vs. Promotion. In: Beyond Personalization, IUI WS, vol. 5 (2005)
7. Biran, O., Cotton, C.: Explanation and Justification in Machine Learning: A Survey. In: IJCAI-17 Workshop on Explainable AI (XAI), p. 8 (2017)
8. Bizer, C.: The Emerging Web of Linked Data. IEEE Intelligent Systems **24**(5), 87–92 (2009)
9. Chang, S., Harper, F.M., Terveen, L.G.: Crowd-based Personalized Natural Language Explanations for Recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 175–182. ACM (2016)
10. Chen, C., Zhang, M., Liu, Y., Ma, S.: Neural Attentional Rating Regression with Review-level Explanations. In: Proceedings of the 2018 World Wide Web Conference, pp. 1583–1592. International World Wide Web Conferences Steering Committee (2018)
11. Chen, G., Chen, L.: Augmenting Service Recommender Systems by Incorporating Contextual Opinions from User Reviews. User Modeling and User-Adapted Interaction **25**(3), 295–329 (2015)
12. Chen, L., Chen, G., Wang, F.: Recommender Systems based on User Reviews: the state of the art. User Modeling and User-Adapted Interaction **25**(2), 99–154 (2015)
13. Chen, L., Wang, F.: Explaining Recommendations based on Feature Sentiments in Product Reviews. In: Proceedings of the 22nd International Conference on Intelligent User Interfaces, pp. 17–28. ACM (2017)
14. Coyle, M., Smyth, B.: Explaining Search Results. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI'05, pp. 1553–1555. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2005)
15. Cramer, H., Evers, V., Ramlal, S., Van Someren, M., Rutledge, L., Stash, N., Aroyo, L., Wielinga, B.: The Effects of Transparency on Trust and Acceptance of a Content-based Art Recommender. User Modeling and User-Adapted Interaction **18**(5), 455–496 (2008)
16. De Filippo, A., Lombardi, M., Milano, M.: Non-linear optimization of business models in the electricity market. In: International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, pp. 81–97. Springer (2016)
17. Friedrich, G., Zanker, M.: A Taxonomy for Generating Explanations in Recommender Systems. AI Magazine **32**(3), 90–98 (2011)

18. Gedikli, F., Jannach, D., Ge, M.: How Should i Explain? a Comparison of Different Explanation Types for Recommender Systems. International Journal of Human-Computer Studies **72**(4), 367–382 (2014)
19. Goodman, B., Flaxman, S.: European Union Regulations on Algorithmic Decision-Making and a" Right to Explanation". arXiv preprint arXiv:1606.08813 (2016)
20. Guha, R., Gupta, V., Raghunathan, V., Srikant, R.: User Modeling for a Personal Assistant. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, pp. 275–284. ACM (2015)
21. Haveliwala, T.H.: Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. IEEE Trans. Knowl. Data Eng. **15**(4), 784–796 (2003)
22. He, X., Chen, T., Kan, M.Y., Chen, X.: Trirank: Review-aware Explainable Recommendation by Modeling Aspects. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 1661–1670. ACM (2015)
23. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining Collaborative Filtering Recommendations. In: CSCW 2000, pp. 241–250 (2000)
24. Hernández-Rubio, M., Cantador, I., Bellogín, A.: A Comparative Analysis of Recommender Systems based on Item Aspect Opinions Extracted from User Reviews. User Modeling and User-Adapted Interaction pp. 1–61 (2018)
25. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems: an Introduction. Cambridge University Press (2010)
26. Johnson, H., Johnson, P.: Explanation Facilities and Interactive Systems. In: Proceedings of the 1st international conference on Intelligent user interfaces, pp. 159–166. ACM (1993)
27. Knijnenburg, B., Bostandjiev, S., O'Donovan, J., Kobsa, A.: Inspectability and Control in Social recommenders. In: RecSys 2012, pp. 43–50 (2012)
28. Knijnenburg, B., Willemsen, M.: Evaluating Recommender Systems with User Experiments. In: Recommender Systems Handbook., pp. 309–352. Springer (2015)
29. Kullback, S., Leibler, R.A.: On Information and Sufficiency. The annals of mathematical statistics **22**(1), 79–86 (1951)
30. Liu, B., Zhang, L.: A survey of Opinion Mining and Sentiment Analysis. In: Mining text data, pp. 415–463. Springer (2012)
31. Lu, Y., Dong, R., Smyth, B.: Why I Like It: Multi-Task Learning for Recommendation and Explanation. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 4–12 (2018)
32. Marcus, M., Kim, G., Marcinkiewicz, M.A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., Schasberger, B.: The Penn Treebank: Annotating Predicate Argument Structure. In: Proceedings of the workshop on Human Language Technology, pp. 114–119. Association for Computational Linguistics (1994)
33. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: NIPS, pp. 3111–3119 (2013)
34. Misztal, J., Indurkhya, B.: Explaining contextual recommendations: Interaction design study and prototype implementation. In: IntRS@ RecSys, pp. 13–20 (2015)
35. Muhammad, K.I., Lawlor, A., Smyth, B.: A Live-User Study of Opinionated Explanations for Recommender Systems. In: Proceedings of the 21st International Conference on Intelligent User Interfaces, pp. 256–260. ACM (2016)
36. Musto, C., de Gemmis, M., Semeraro, G., Lops, P.: A multi-criteria recommender system exploiting aspect-based sentiment analysis of users' reviews. In: Proceedings of the eleventh ACM conference on recommender systems, pp. 321–325 (2017)
37. Musto, C., Narducci, F., Lops, P., De Gemmis, M., Semeraro, G.: ExpLOD: A Framework for Explaining Recommendations based on the Linked Open Data Cloud. In: Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16, pp. 151–154. ACM, New York, NY, USA (2016). DOI 10.1145/2959100.2959173. URL `http://doi.acm.org/10.1145/2959100.2959173`
38. Musto, C., Narducci, F., Lops, P., de Gemmis, M., Semeraro, G.: Linked Open Data-based Explanations for Transparent Recommender Systems. International Journal of Human-Computer Studies **121**, 93–107 (2019)

39. Musto, C., Rossiello, G., de Gemmis, M., Lops, P., Semeraro, G.: Combining text summarization and aspect-based sentiment analysis of users' reviews to justify recommendations. In: Proceedings of the 13th ACM Conference on Recommender Systems, pp. 383–387 (2019)
40. Nakagawa, H., Mori, T.: A Simple but Powerful Automatic Term Extraction Method. In: Coling 2002: second international workshop on computational terminology-Volume 14, pp. 1–7. Association for Computational Linguistics (2002)
41. Nunes, I., Jannach, D.: A Systematic Review and Taxonomy of Explanations in Decision Support and Recommender Systems. User Modeling and User-Adapted Interaction **27**(3-5), 393–444 (2017)
42. Qiu, F., Cho, J.: Automatic Identification of User Interest for Personalized Search. In: Proceedings of the 15th international conference on World Wide Web, pp. 727–736. ACM (2006)
43. Radev, D.R., Jing, H., Sty, M., Tam, D.: Centroid-based Summarization of Multiple Dcuments. Information Processing and Management **40**(6), 919–938 (2004)
44. Rossiello, G., Basile, P., Semeraro, G.: Centroid-based Text Summarization through Compositionality of Word Embeddings. In: G. Giannakopoulos, E. Lloret, J.M. Conroy, J. Steinberger, M. Litvak, P.A. Rankel, B. Favre (eds.) Proceedings of the Workshop on Summarization and Summary Evaluation Across Source Types and Genres, MultiLing@EACL 2017, Valencia, Spain, April 3, 2017, pp. 12–21. Association for Computational Linguistics (2017). URL `https://aclanthology.info/papers/W17-1003/w17-1003`
45. Schilit, B.N., Adams, N., Want, R., et al.: Context-aware Computing Applications. Xerox Corporation, Palo Alto Research Center (1994)
46. Sinha, R., Swearingen, K.: The Role of Transparency in Recommender Systems. In: CHI'02 extended abstracts on Human factors in computing systems, pp. 830–831. ACM (2002)
47. Suglia, A., Greco, C., Musto, C., De Gemmis, M., Lops, P., Semeraro, G.: A deep architecture for content-based recommendations exploiting recurrent neural networks. In: Proceedings of the 25th conference on user modeling, adaptation and personalization, pp. 202–211 (2017)
48. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: MoviExplain: a Recommender System with Explanations. In: Proceedings of the third ACM conference on Recommender systems, pp. 317–320. ACM (2009)
49. Tintarev, N., Masthoff, J.: A survey of explanations in recommender systems. In: Data Engineering Workshop, 2007 IEEE 23rd International Conference on, pp. 801–810. IEEE (2007)
50. Tintarev, N., Masthoff, J.: Evaluating the Effectiveness of Explanations for Recommender Systems. UMUAI **22**(4-5), 399–439 (2012)
51. Vig, J., Sen, S., Riedl, J.: Tagsplanations: Explaining Recommendations Using Tags. In: Proceedings of the 14th international conference on Intelligent user interfaces, pp. 47–56. ACM (2009)