Text-based recommender system with explanatory capabilities

Pablo Pérez-Núñez (pabloperez@uniovi.es)
University of Oviedo at Gijón
Jorge Díez
University of Oviedo at Gijón
Antonio Bahamonde
University of Oviedo at Gijón
Oscar Luaces
University of Oviedo at Gijón

Research Article

Keywords: Recommender systems, Explainability, Cold-start, Text-based recommendations

Posted Date: May 19th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1536768/v2

Additional Declarations: No competing interests reported.

Text-based recommender system with explanatory capabilities

Pablo Pérez-Núñez, Jorge Díez, Antonio Bahamonde and Oscar Luaces

Artificial Intelligence Center, University of Oviedo at Gijón, Campus de Viesques, S/N, Gijón, 33240, Asturias, Spain.

*Corresponding author(s). E-mail(s): pabloperez@uniovi.es;
Contributing authors: jdiez@uniovi.es; abahamonde@uniovi.es; oluaces@uniovi.es;

Abstract

Recommender systems have proven their usefulness both for companies and customers. The former increase their sales and the latter get a more satisfying shopping experience. These systems can benefit from the advent of explainable artificial intelligence, since a well-explained recommendation will be more convincing and may broaden the customer's purchasing options. Many approaches offer justifications for their recommendations based on the similarity (in some sense) between users, past purchases, etc., which require some knowledge of the users. In this paper we present a recommender system with explanatory capabilities which is able to deal with the so-called cold-start problem, since it does not require any previous knowledge of the user. Our method learns the relationship between the products and some relevant words appearing in the textual reviews written by previous customers for those products. Then, starting from the textual query of a user's request for recommendation, our approach elaborates a list of products and explains each recommendation on the basis of the compatibility between the query's words and the relevant terms for each product.

Keywords: Recommender systems, Explainability, Cold-start, Text-based recommendations

1 Introduction

The layout of the products offered in stores (physical or on-line) has the main purpose of encouraging our consumption, thus increasing the sales. Usually, some items are highlighted because the sellers want to promote them, or because the users (consumers) most likely buy them. But detecting the personal tastes of consumers is not straightforward, so special algorithms called recommender systems (RS) have emerged (Resnick et al, 1994; Ricci et al, 2011). Ideally, recommender systems take into account the interactions between users and items, inducing a predictive model able to make personalized recommendations.

It has been proven that the use of recommender systems increases the consumption of products and services offered by online platforms (Pathak et al, 2010). Clearly, consumers are pleased with the recommendations provided by the RS, thus consuming more. On the other hand, they are also beneficial for the sellers, for obvious reasons: increasing their sales increase their benefits.

However, a lack of confidence in the decisions taken by algorithms has raised lately. This mistrust is due to the bias that some algorithms may present in their behavior (Banker and Khetani, 2019), which can induce the users to make wrong choices.

Explainable Artificial Intelligence (XAI) has been devised to tackle this issue. It is a recent branch of AI aimed at explaining the output of intelligent algorithms in a human-friendly manner, such that users can understand the reasons behind the algorithms' decisions (Monroe, 2018).

More specifically, XAI is being applied in the field of recommender systems (Zhang and Chen, 2018; Burke et al, 2021) to provide convincing recommendations. The benefits of these explanations are, mainly, the following (Tintarev and Masthoff, 2007):

i) *Transparency*: users can understand how the recommender works.

ii) *Scrutability*: wrong recommendations can be analyzed, and the model results can be tuned by human conscious intervention.

iii) *Trust*: the reasons for a recommendation are explained to the users.

iv) *Persuasiveness*: explanations can emphasize those aspects of the product in which the user is really interested.

v) *Effectiveness* and *efficiency*: users can make good choices quickly when the reasons for the recommendations are known.

vi) *Satisfaction*: users are pleased with the recommended items, saving time in their choices.

In this paper we present an approach to build a recommender system capable of explaining its recommendations to a user thanks to the textual reviews written by other users. In order to train the model, we will take advantage of some descriptive terms used by the customers in their reviews. The model will be trained to find out the relationship between the textual reviews, represented using a *bag of words (BoW)* encoding, and the items they belong to. The BoW will be carried out considering only the mentioned descriptive terms, so we will have to preprocess the texts in order to filter out the irrelevant words. Once trained, the model will be able to recommend the most adequate items with respect to the terms included in a user query, as well as to elaborate an explanation based on those terms.

One important advantage of our approach is that we can apply our recommender system in different countries and languages, without the need to make changes to the system, as we will explain later.

Additionally, our approach does not require to have any previous knowledge of a user, given that we are not going to handle user profiles. Instead, our aim is to recommend items starting from a simple textual request in natural language.

Therefore, one remarkable advantage of this approach is that our system is able to deal with the so-called *cold-start* problem (Schein et al, 2002). This can be particularly useful in contexts where items have lots of assessments/reviews, but each user reviewed only a very small number of items.

The rest of the paper is organized as follows: in the next section we revise and comment some previous works related to our approach. Then, we detail our proposed method and, finally, we describe an experimental setting, where we introduce the datasets used and discuss the obtained results.

2 Related work

Recommender systems can be classified in two main classes attending to the kind of explanations they offer: they can show how the recommendation was elaborated, or they can justify it (Tintarev and Masthoff, 2012; Jia et al, 2020). The former are more transparent, allowing the users to understand the causes for an item to be recommended, while the justifications offered by the latter are typically supported by a neighborhood relationship between users and items. Transparent recommenders can be critically analyzed (scrutinized) by the users (Pu et al, 2012), so that they can make a better use of these systems, thus obtaining an improvement in their suggestions.

Zhang and Chen (2018) collected a comprehensive bibliography on explainable recommender systems, although there are not many works which use the text of the reviews. Almahairi et al (2015) presented one of the first works incorporating textual reviews to improve recommendations (although not to explain them) using a *deep learning* approach, more specifically, recurrent neural networks.

The same kind of neural networks were used in the work of Costa et al (2018), which was aimed at generating reasonably well-formed explanatory

sentences together with the expected opinion of the user. In this sense, Bartoli et al (2016) warn about the risk posed by automatic generation of reviews, since they can manipulate the users' opinions.

Dias et al (2017) presented a more similar approach to ours, in which they create an explainable text-based recommendation system able operate in cold-start situations. The authors proposed a variation of the *word2vec* (Mikolov et al, 2013) method in order to learn a common embedding both for the words of the reviews and the concepts (users and the items). However, their approach requires retraining the network for every new user, and the complexity to obtain explanations is not negligible, given that they have to compute the similarity between the user embedding and the rest of embeddings.

In this regard, it is worth noting the importance of the encoding method used to represent textual information. Prior to the advent of deep learning in natural language processing, the most effective and widely used method for text encoding was the BoW. This method consists of representing a text as the sum of the one-hot vectors of the terms (or words) they are composed of, taken from a previously defined vocabulary. In other words, a text is represented as a vector with the length of the vocabulary, having values greater than 0 only in the corresponding positions of the terms present in the text.

This approach to represent texts has been refined using techniques such as TF-IDF or the use of n-grams, so it is still a widely used method that yields results comparable to those of other more sophisticated methods (Zhao and Mao, 2017). Indeed, there is no significant difference in some applications between using bigrams as terms in BoW, or word embeddings (Shao et al, 2018).

3 Proposed method

In this section we explain our proposal for the construction of a Text-based Recommender with eXplanations (TRecX).

In the context of recommender systems we usually start from a set of users, $\mathcal{U}$, and a set of items or products, $\mathcal{P}$. These data are collected in a dataset, $\mathcal{D}$, such that there is a record for every interaction between any user $u \in \mathcal{U}$ with any product $p \in \mathcal{P}$, in which the user expressed his/her satisfaction by means of a score, together with a textual review, r, explaining the reasons for that assessment. The scores are useless in our approach, as we will explain below, so we will take into account only the textual reviews. Thus, our dataset can be denoted as a set of triples:

$$(u, p, r) \in \mathcal{D}. \tag{1}$$

In particular, we want to focus in those contexts where most of the users have very few interactions with different products, thus hindering the construction of personalized recommenders. Instead of taking advantage of the (almost non-existent) consumption history, our approach will recommend products whose reviews are somewhat related to the user's query for recommendation. Moreover, this relationship is also used to yield explanations. Our method, unlike the work of Costa et al (2018), does not seek to generate sentences, which can sometimes result somewhat artificial, but to explain the recommendation based on the occurrence of some relevant terms in the user's query that are also relevant to the recommended products, as stated by other users in their reviews.

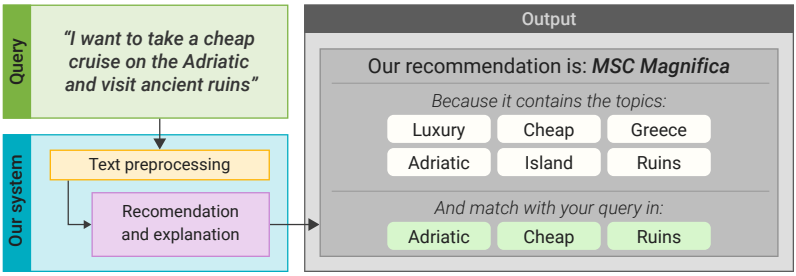


Fig. 1: Workflow for recommending cruises using our approach. The input to the recommender system is a textual query in natural language. After some processing, it will eventually recommend *MSC Magnifica*, and explains the recommendation on the basis of the relationship with some terms included both in the query and in the reviews provided by other users for this product.

Let's take, for instance, the case of sea cruises: each cruise may have a lot of reviews, but it is unusual for a user to enjoy many cruises. In this hypothetical context, our system will operate as depicted in Figure 1.

In order to make this possible, our model must learn the relationship between the words in the review and the product they refer to. In other words, we will train the model to predict to which product belongs each review. The rationale behind this approach is to consider users' queries (asking for a recommendation) as a kind of *a priori* review; that is, we address the problem as: what would we recommend so that the user will eventually write a review with the terms included in his/her request?

Obviously, what we express in a request and in a review is rather different from a semantic point of view. Our hypothesis is that the difference is mainly due to the verbs used, but both reviews and requests are similar regarding the things they mention and their characteristics, that is, the *nouns* and *adjectives* used.

3.1 Selection of relevant terms

The construction of the set of relevant terms is a key point to allow the model both to infer a good recommendation as well as to provide good explanations. Our proposal for this task consists of applying Part-of-Speech (PoS) tagging techniques (Schmid, 1994) to get rid of all words except nouns and adjectives. Then, we will consider as relevant terms a percentage of the most used nouns and adjectives in all the reviews. This percentage may be context-dependent so we recommend to determine it experimentally, in order to achieve an adequate trade-off between performance and explicability.

Once we have the set of relevant terms we can encode the textual reviews using BoW, in order to feed the model during training as well as during its operation, once trained.

One important advantage of this approach is that we can apply our recommender system in different countries and languages, as we will show in Section 4. For this purpose, we take advantage of the fact that computational linguistics provides reliable computer-based PoS tagging techniques for multiple languages.

3.2 Computing the recommendation

The proposed model is a classifier based on a multinomial logistic regression, as depicted in Figure 2. The model is trained with textual reviews, previously encoded using a bag of words approach, as explained above. Each review is encoded in a vector $\boldsymbol{t}$, where each component records the number of occurrences of the corresponding term in the text. This vector is then normalized using the L1 norm, thus gathering the probability of occurrence of each term in the textual review. This is also known as *term frequency (TF)* encoding.

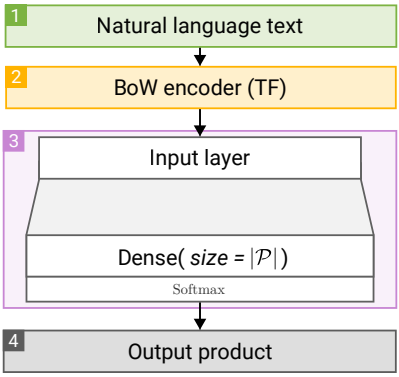


Fig. 2: Architecture of the proposed system. Textual reviews, 1, encoded using BoW, 2, are processed using 3, a fully connected dense layer with a linear activation function, and with as many output elements as products, followed by a softmax layer. The output, 4, is a vector with the probabilities of belonging to each product.

The normalized BoW vector, $\boldsymbol{t}_{L1}$, is used as input to a fully connected layer with a linear activation function. The output of the layer is then transformed into a vector of probabilities using a *softmax* layer. Thus, the classifier outputs a vector whose dimension must coincide with the number of products, $\mathcal{P}$, that can be recommended:

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W}\boldsymbol{t}_{L1} + \boldsymbol{b}) \tag{2}$$

where $\hat{\boldsymbol{y}}_p = \Pr(p|\boldsymbol{t})$, i.e., the p-th component of $\hat{\boldsymbol{y}}$ is the estimated probability of the text to be a review of the p-th product. We pose a multiclass learning task, where we learn the matrix $\boldsymbol{W}$ and the bias $\boldsymbol{b}$, and where the loss function to be optimized is the usual categorical cross entropy:

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{p=1}^{\mathcal{P}} \boldsymbol{y}_p \cdot \log(\hat{\boldsymbol{y}}_p) \tag{3}$$

where $\boldsymbol{y}$ is the one-hot vector with the ground truth.

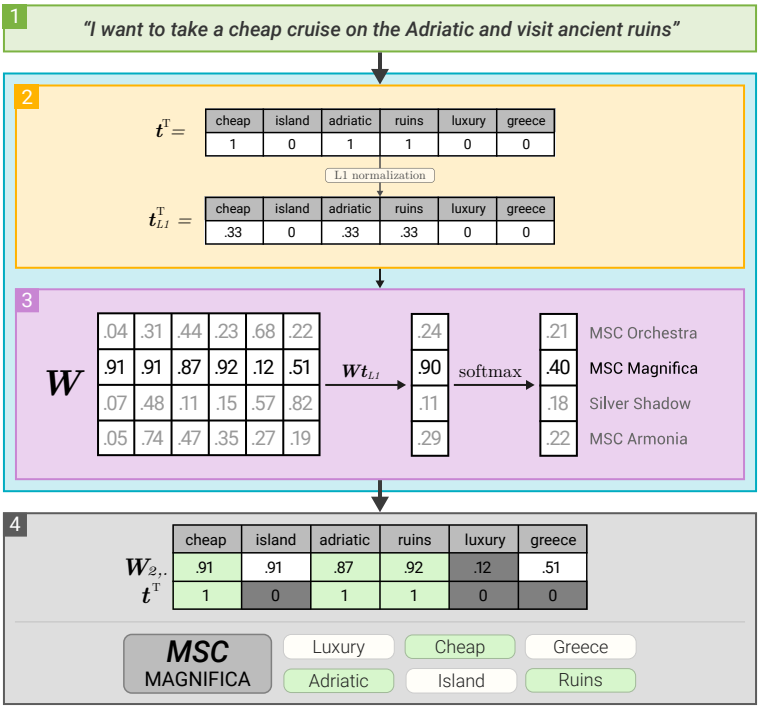


Fig. 3: Detail of the recommendation and explanation tasks in a toy example. For simplicity, the bias vector b was omitted. The user's query, 1, is encoded in an L1-normalized vector, 2, which is then used to predict the output probabilities, 3. Setting side by side the probabilities and the terms in the user's query we can elaborate an explanation, 4.

The benefits of such a simple model are two-fold: we expect to obtain a good generalization with no overfitting, and we can easily draw out explanations to justify the results of the classifier (the recommendation).

3.3 Explaining the recommendation

We have just seen that the output probabilities (2) of the products depend on the parameters learned, that is, $\boldsymbol{W}$ and $\boldsymbol{b}$. Taking into account that the bias of each product p, i.e., $\boldsymbol{b}_p$, is constant, we can focus on $\boldsymbol{W}_{p,\cdot}$ (row p) to analyze the relevance of every linguistic term for the recommendation of the p-th product.

Let's detail how TRecX proceeds with a toy example depicted in Figure 3, in the context of cruise recommendation previously introduced. For the sake of simplicity, we are going to consider only six relevant terms (*"cheap"*, *"island"*, *"adriatic"*, *"ruins"*, *"luxury"*, and *"greece"*), and only four different cruises that can be recommended (MSC Orchestra, MSC Magnifica, Silver Shadow and MSC Armonia).

In this setting, and starting from a query written in natural language (step 1), we filter out irrelevant words to keep only relevant terms, which are used to create the vector $\boldsymbol{t}$ containing the number of occurrences of each term (BoW encoding). In the given query we have only 3 out of 6 relevant words, appearing only once each one; thus, the corresponding components of $\boldsymbol{t}$ are 1's. This vector is then normalized using the L1 norm, yielding $\boldsymbol{t}_{L1}$ (step 2). The query, encoded as an L1-normalized 6-dimensional vector $\boldsymbol{t}_{L1}$, is then used as input to a multinomial logistic regression (step 3), which estimates the probability of the input text to be related to each one of the possible products.

In the event that the training process yielded the matrix $\boldsymbol{W}$ shown in Figure 3, the logistic regression (we have omitted the intercept vector, $\boldsymbol{b}$, for clarity) will yield an output vector with probabilities indicating that the most recommendable cruise is MSC Magnifica.

An interesting side-effect of this approach is that we can justify or *explain the recommendation* on the basis of the relevant terms found both in the user request and in the reviews of the recommended products. Having into account the parameters learned during the training of the model, i.e., the matrix $\boldsymbol{W}$, we can finally provide a justification for the recommendation (step 4). Thus, looking at the second row of the weight matrix, $\boldsymbol{W}_{2,\cdot}$, and comparing it with the input vector, t, we observe that the relevant terms in the query are also very relevant for the corresponding product, MSC Magnifica, due to their

sign (positive) and their magnitude (the highest in the row). Therefore, this information can be used to provide an explanation in the following sense: *"*MSC Magnifica *is recommended because its reviews relate (with the highest probability) this product to* cheap*,* adriatic *and* ruins*, as requested by the user".*

4 Experimental validation

We carried out some experiments in order to validate our approach. For this purpose we we constructed a group of datasets with textual reviews downloaded from TripAdvisor[1]. Restaurant recommendation is a task where the average number of reviews per product is much larger than per user. Therefore, this task fits in the kind of problems our approach is intended to address.

In the rest of this section we describe how we created the datasets, as well as the implementation details of our approach. Finally we present and discuss the experimental results, also illustrating how our proposal could be eventually used to build a restaurant recommender application with explanation capabilities.

4.1 Datasets

We downloaded textual reviews of restaurants in five cities around the world from the TripAdvisor web. The cities were selected taking into account several factors such as the population, number of restaurants, geographical location and language. The choice of cities in different countries is important to prove that our approach is robust regarding the language used in the reviews. On the other hand, selecting cities with different values of population, ranging from less than half a million up to more than eight million inhabitants, let us

[1] https://www.tripadvisor.com

Table 1: Main characteristics of the five datasets after filtering out reviews with more than 2000 characters and restaurants with less than 100 reviews. It should be noted that, for each city, we only included reviews in its native language. *Popularity* represents the percentage of total reviews that have been written for the most popular restaurant.

City	Gijón	Barcelona	Madrid	Paris	New York
Abbreviation	GJN	BCN	MDR	PRS	NYC
Population	300K	1.6M	3.2M	2.1M	8.3M
Language	Spanish	Spanish	Spanish	French	English
# Reviews	39889	320236	474688	807535	826117
# Restaurants	148	1325	1634	3423	1988
Avg revs/rest	269.5	241.7	290.5	235.9	415.6
Popularity	6.8%	0.7%	1.3%	0.8%	1.9%

analyze the behavior of our approach in a variety of sizes of the datasets (the larger the population, the larger the number of restaurants and reviews).

Table 1 summarizes the most important characteristics of the datasets after filtering out examples whose restaurant had less than 100 reviews. We have made these datasets publicly available[2] in its raw format, i.e., with no preprocessing.

Worth of mention is the number of restaurants in each dataset, which determines the size of the output layer of its corresponding model. Another noticeable characteristic is the average number of reviews per restaurant (Avg revs/rest), which is very similar for all the cities except for New York City, whose value is almost double than for the others. Finally, the value shown as *Popularity* corresponds to the percentage of reviews of the most popular (i.e., reviewed) restaurant. The city of Gijón has the highest value in this characteristic with a large difference with respect to the rest of the cities. Our guess is that this is due to the fact that Gijón is the smallest city, the one with the smallest restaurant offer, and this fact could lead to a concentration of popularity in a few specific restaurants.

[2] https://doi.org/10.5281/zenodo.5644891.

4.2 Restaurant recommendation: implementation details and results

We devised an experimental setting based on the data downloaded from TripAdvisor in order to assess the performance of our approach when recommending restaurants. The textual reviews were preprocessed as follows: case normalization (all letters in lower case), elimination of punctuation symbols, lemmatization (transforming all the variations and inflected forms of words into its common lemma), and elimination of accent marks and numbers. We also removed long reviews, with more than 2000 characters, and restaurants with less than 100 textual reviews.

After the preprocessing, we reserved 10% of the examples in each dataset (1) for testing purposes, 80% for training and the remaining 10% for validation. The validation sets were used to seek for adequate hyperparameters for each model/dataset. The splits were made randomly, stratified by restaurant, in order to maintain the distribution as well as to ensure that any restaurant in the test/validation sets could be eventually recommended (unknown products cannot be recommended).

The implementation of our recommender, TRecX, was made in Python using TensorFlow (Abadi et al, 2016). The text preprocessing techniques mentioned previously, including lemmatization and PoS tagging (needed to the BoW encoding), were applied using the spaCy library (Honnibal et al, 2020).

The optimization during the training was carried out with the Adam algorithm (Kingma and Ba, 2014) and an *early stopping* strategy. The hyperparameters of each model, i.e., the learning rate and batch size, were chosen ad hoc for each city after a grid search on the corresponding validation subset.

On the other hand, the vocabulary of relevant terms used for BoW encoding was made up with 10% of the most used nouns and adjectives found in the

textual reviews of each dataset/city. The value for this hypeparameter was chosen after a series of tests on the validation sets of several cities.

We compared the performance of TRecX with a more complex classifier, whose architecture was based on a *long-short term memory (LSTM)* network Hochreiter and Schmidhuber (1997), much more powerful for NLP tasks such as the one at hand. The architecture of this classifier, labeled as LSTM2rst, consisted of an LSTM layer with 256 elements, connected to an output layer with the required number of elements (as many as restaurants in the corresponding dataset) and a softmax layer, in order to obtain a vector of probabilities.

The input encoding used for LSTM2rst was also more sophisticated: we used a *word2vec* (Mikolov et al, 2013) approach instead of the BoW encoding used by TRecX. Caselles-Dupré et al (2018) claim that recommender systems using word2vec encodings obtained from the texts of the task at hand yield better performance than those obtained from generic texts, so we trained our word2vec encoders with the (preprocessed) textual reviews, using a window size of 5 words and 300-dimensional output vectors.

The reason for comparing TRecX with LSTM2rst, a more complex approach with a richer input representation, is to test if it is worth the trade-off between the possible penalty in performance of TRecX and its ability to offer explanations for its recommendations.

Table 2 displays the scores of Precision (@1, @5 and @10) obtained by TRecX and LSTM2rst in the test sets of each city. We also included, as a *baseline* reference, the precision obtained when always recommending the most popular restaurants. This baseline recommender is clearly the worst, but the difference in precision between the smallest city (Gijón) and the rest is noteworthy. In fact, we already noticed some peculiarities regarding the popularity

Table 2: Precision@{1,5,10} in percentage obtained by the different systems. In bold are the best results for every city (rows) in every measure.

Dataset	baseline			TRecX			LSTM2rst		
	P@1	P@5	P@10	P@1	P@5	P@10	P@1	P@5	P@10
GJN	6.8	17.6	25.7	**35.9**	**60.2**	**70.5**	34.8	58.4	69.6
BCN	0.7	3.0	4.8	28.1	46.5	54.5	**29.0**	**49.1**	**57.7**
MDR	1.3	4.4	6.7	**34.6**	53.8	61.0	34.4	**54.9**	**63.0**
PRS	0.8	2.1	3.1	23.7	38.5	44.9	**27.4**	**43.3**	**50.3**
NYC	1.9	5.9	8.8	38.8	57.6	64.8	**40.4**	**58.9**	**66.5**

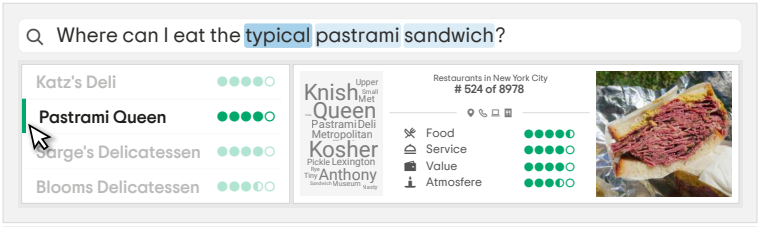
when presenting the characteristics of the datasets in Table 1, where we indicated the percentage of reviews of the most popular restaurant of every city. As we mentioned previously, we guess that popularity is more important in places with a limited offer of restaurants. Notice also that the scores in Precision@1 coincide with the percentage of reviews of the most popular restaurant in every city due to the stratified split of data.

On the other hand, the performance of LSTM2rst is, in general, better than that of TRecX, as expected. The word2vec embedding learned from the data, together with a more sophisticated NLP approach used on the textual reviews by means of the LSTM network, make LSTM2rst more accurate than TRecX. However, the scores obtained by TRecX are only slightly worse, despite using only 10% of the words in the textual reviews as input information. Moreover, it is straightforward to elaborate an explanation for the recommendations, as we already introduced in Section 3.3. We illustrate this ability in the following with a real example.

4.3 Explanation of the recommendation

Many recommender systems can only justify their recommendations by some kind of similarity (neighborhood) between users, products and reviews, as we mentioned in Section 2. Using deep learning based models, such as LSTM

(a) Gijón



(b) New York City

(c) Paris

Fig. 4: Mock-up of a recommender application with explanations. The results shown were actually obtained from true recommendations and explanations of our TRecX in the three datasets/cities indicated.

networks, allows us to obtain more complex and more accurate encoding approaches to project those users and products in a space where we can compute such similarity. But the justification of the recommendation based on these sort of distance measures is barely understandable by the users of the recommender system, who cannot scrutinize the recommendation.

Our proposal, however, learns the relevance of the most important linguistic terms which characterize the products, and this relevance, gathered in the

matrix $\boldsymbol{W}$ of (2), is then used to elaborate explanations in the same linguistic terms appearing in the users' queries.

Figure 4 depicts a mock-up of a restaurant recommender application based on TRecX. The application will be able to operate in different languages, justifying its recommendations, following the idea explained in Section 4.3 with the synthetic example of cruises.

For instance, let's look at Figure 4b, which exemplifies a request of recommendation in New York City; the query *Where can I eat the typical pastrami sandwich?* triggers a suggestion of the top n (4 in the mock-up) restaurants with the highest probability to be related to the relevant terms of the query. Pointing to a given restaurant, the system can highlight some words of the query with an intensity proportional to their relevance for the recommendation, i.e., proportional to their weights in the corresponding row of $\boldsymbol{W}$. Additionally, the user can see a word cloud built up depending on all the weights of the row which corresponds to the selected restaurant.

In the example of the figure we focused on the restaurant *Pastrami Queen*, so we can see that the terms *"pastrami"*, *"sandwich"* and, more importantly, *typical*, are relevant to describe this restaurant. The word cloud associated to this restaurant shows that other relevant terms are, for instance, *"Knish"* and *"Kosher"*, giving a clue of the food which is frequently mentioned in its reviews, and helping the user to take a better decision for choosing one of the recommended restaurants.

It should be noted that all the results shown in the mock-up depicted in Figure 4 were actually obtained from the output provided by TRecX on the datasets of Gijón, New York City and Paris, respectively.

5 Conclusions

In this paper we have shown an approach to make a recommender system able to provide explanations based only on the textual reviews of products. From these reviews we have devised a transparent recommendation system that learns the relevance of the terms that define the most important aspects of the products. These terms and their relevance are combined to provide explanations in a user-friendly manner, so that users can understand how the system has built the recommendation. Looking at the most relevant terms that define a product, the system also helps users to make the right decision.

The lack of complexity of our recommender does not hinder significantly its performance, as we have shown in a comparison with a more complex LSTM-based approach on datasets of different cities and sizes.

Additionally, the presented approach does not require any previous information about the users, so it is well-suited for cold-start scenarios. Our model is induced just from textual reviews of products.

Finally, we would like to highlight that our system can be directly applied to datasets in different languages without any change, provided that the language supports PoS tagging.

Acknowledgements

This work was funded by grant PID2019-109238GB-C21 (Ministry of Science and Innovation, Spain). The participation of Pablo Pérez-Núñez was funded by the Principality of Asturias through the predoctoral granting program *Severo Ochoa* (ref. BP19-012). We also thank NVIDIA Corporation for the donation of a Titan Xp GPU used in this research.

References

Abadi M, Barham P, Chen J, et al (2016) Tensorflow: A system for large-scale machine learning. In: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016

Almahairi A, Kastner K, Cho K, et al (2015) Learning distributed representations from reviews for collaborative filtering. In: Proceedings of the 9th ACM Conference on Recommender Systems, pp 147–154

Banker S, Khetani S (2019) Algorithm overdependence: How the use of algorithmic recommendation systems can increase risks to consumer well-being. Journal of Public Policy & Marketing 38(4):500–515

Bartoli A, De Lorenzo A, Medvet E, et al (2016) Best dinner ever!!!: Automatic generation of restaurant reviews with lstm-rnn. In: 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI), IEEE, pp 721–724

Burke R, Ekstrand MD, Tintarev N, et al (2021) Preface to the special issue on fair, accountable, and transparent recommender systems. User Modeling and User-Adapted Interaction 31(3):371–375. https://doi.org/10.1007/s11257-021-09297-5

Caselles-Dupré H, Lesaint F, Royo-Letelier J (2018) Word2vec applied to recommendation: Hyperparameters matter. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp 352–356

Costa F, Ouyang S, Dolog P, et al (2018) Automatic generation of natural language explanations. In: Proceedings of the 23rd international conference on intelligent user interfaces companion, pp 1–2

Dias CE, Guigue V, Gallinari P (2017) Text-based collaborative filtering for cold-start soothing and recommendation enrichment. In: AISR2017

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9(8):1735–1780

Honnibal M, Montani I, Van Landeghem S, et al (2020) spaCy: Industrial-strength Natural Language Processing in Python, https://doi.org/10.5281/zenodo.1212303

Jia Y, Bailey J, Ramamohanarao K, et al (2020) Exploiting patterns to explain individual predictions. Knowledge and Information Systems 62(3):927–950

Kingma D, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980

Mikolov T, Chen K, Corrado G, et al (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781

Monroe D (2018) Ai, explain yourself. Communications of the ACM 61(11):11–13

Pathak B, Garfinkel R, Gopal RD, et al (2010) Empirical analysis of the impact of recommender systems on sales. Journal of Management Information Systems 27(2):159–188

Pu P, Chen L, Hu R (2012) Evaluating recommender systems from the user's perspective: survey of the state of the art. User Modeling and User-Adapted Interaction 22(4-5):317–355

Resnick P, Iacovou N, Suchak M, et al (1994) Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM

conference on Computer supported cooperative work, pp 175–186

Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook. In: Recommender systems handbook. Springer, p 1–35

Schein AI, Popescul A, Ungar LH, et al (2002) Methods and metrics for cold-start recommendations. In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pp 253–260

Schmid H (1994) Part-of-speech tagging with neural networks. cmp-lg/9410018

Shao Y, Taylor S, Marshall N, et al (2018) Clinical text classification with word embedding features vs. bag-of-words features. In: 2018 IEEE International Conference on Big Data (Big Data), IEEE, pp 2874–2878

Tintarev N, Masthoff J (2007) A survey of explanations in recommender systems. In: 2007 IEEE 23rd international conference on data engineering workshop, IEEE, pp 801–810

Tintarev N, Masthoff J (2012) Evaluating the effectiveness of explanations for recommender systems. User Modeling and User-Adapted Interaction 22(4-5):399–439

Zhang Y, Chen X (2018) Explainable recommendation: A survey and new perspectives. arXiv preprint arXiv:180411192

Zhao R, Mao K (2017) Fuzzy bag-of-words model for document representation. IEEE transactions on fuzzy systems 26(2):794–804