Expert Systems With Applications 236 (2024) 121268

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Computing recommendations from free-form text

Lukas Eberhard [a,*], Kristina Popova [a], Simon Walk [b], Denis Helic [c]

[a] Graz University of Technology, Graz, Austria
[b] Avery Dennison - atma.io, Graz, Austria
[c] Modul University Vienna, Vienna, Austria

ARTICLE INFO

Keywords:
Deep learning
Keyword extraction
Named entity recognition
Narrative-driven recommendations
Aspect-based sentiment analysis

ABSTRACT

While searching for consumer goods, users frequently ask for suggestions from their peers by writing short free-form textual requests. For example, when searching for movies users may ask for "*Drama movies with a mind-bending story and a surprise ending, such as Fight Club*" in one of the many online discussion boards. Despite the recent developments in large language models (LLMs) and natural language processing (NLP), modern recommender systems still struggle to process such requests. Therefore, in this paper we evaluate several approaches for annotating structured information from such short, free-form natural language user texts to calculate recommendations. We set up this evaluation as a two phase processes including (a) identification of the best NLP approach to identify key elements of users' requests, and (b) assessment of the quality of recommendations computed with such elements.

For our evaluation, we use a gold-standard reddit movie recommendation dataset consisting of annotations, manually created by crowdworkers who extracted keywords, actor names and movie titles. Using this dataset we evaluate a collection of more than 30 NLP and five recommender approaches. In addition, we perform an ablation study to assess relative annotation importance for movie recommendations. We find that domain-specific deep learning models, trained on a subset of data as well as embedding-based recommendation approaches are able to match the recommendation performance of recommendations computed from manual annotations. These promising results warrant further investigation in automatic processing of short free-form texts for computation of recommendations. Specifically, we provide insights into which NLP models and configurations work best for automatically annotating free text to compute (movie) recommendations, hence substantially reducing the search space for combinations of NLP and recommendation algorithms in the movie and potentially other domains.

1. Introduction

Although users commonly write free-form questions in online discussion forums, group chats or social media to ask other users for recommendations (Bogers & Koolen, 2017), current recommender systems still provide only limited possibilities for retrieving recommendations via natural text. Responses from other users to such questions are neither instantaneous nor guaranteed and frequently lack in quality, which results in a poor user experience. Hence, this situation raises the need for automatic computation of recommendations from free-form text. This need is further corroborated by recent increases in popularity and adoption of voice assistants and chatbots (Langevin et al., 2021; Sabir et al., 2022; Setlur & Tory, 2022), which represent only one of many novel and unique outlets to interact with users and present recommendations (Eberhard et al., 2020; Montazeralghaem et al., 2021; Wu et al., 2021). For example, recent advancements of large language models (LLMs) such as OpenAI's chatbot ChatGPT[1] could be adopted for use in recommender systems as some initial studies have already showcased (Liu et al., 2023).

However, despite the fact that recent progress in natural language processing (NLP) (Bhattacharjee et al., 2020; Hu et al., 2019; Li et al., 2019, 2021), and in particular neural NLP (Do et al., 2019; Lample et al., 2016; Truşcă et al., 2020) and LLMs (Cui et al., 2022; Kasneci et al., 2023; Zhang et al., 2021), allows us to, for example, detect sentiment, extract important keywords from text with high accuracy, enter into dialog with users, or generate high quality answers to user prompts, recommender systems are still missing studies and solutions that combine these various approaches for computing recommendations from text.

* Corresponding author.
E-mail addresses: lukas.eberhard@tugraz.at (L. Eberhard), kristina.popova@alumni.tugraz.at (K. Popova), simon.walk@eu.averydennison.com (S. Walk), denis.helic@modul.ac.at (D. Helic).

[1] https://chat.openai.com/

https://doi.org/10.1016/j.eswa.2023.121268
Received 15 February 2023; Received in revised form 26 June 2023; Accepted 19 August 2023
Available online 27 August 2023

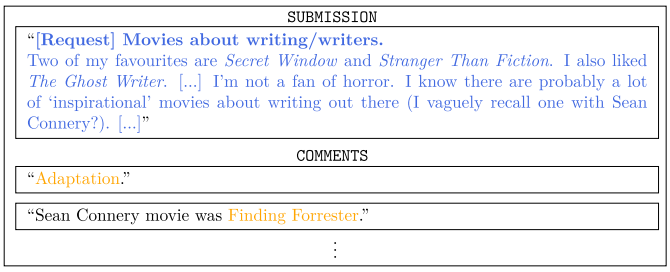

Fig. 1. Request & Suggestions Example. In the request of this reddit submission[2] we can annotate three positive movies (i.e., *Secret Window*, *Stranger Than Fiction*, *The Ghost Writer*), a negative genre (i.e., *horror*), several positive keywords (i.e., *writing*, *writers*, *inspirational*), and a positive actor name (i.e., *Sean Connery*). As suggestions from the reddit community, we can extract the movies *Adaptation* and *Finding Forrester* from the comments.

To close this research gap, we set out to systematically analyze and evaluate a large collection of NLP and recommender algorithms for computing recommendations from short, free-form, user-generated text. To that end, we conduct a two-staged experiment consisting of (i) state-of-the-art NLP methods annotating important words that summarize user recommendation requests, and (ii) recommendation algorithms computing recommendations from the annotated words. We opt for such a two-staged approach to decouple elicitation of user preferences (first stage) from the computation of recommendations (second stage). Among others, such a decoupling allows us to (i) gain insight in how users express their preferences in a short free-form text, (ii) improve explainability of recommendations by presenting parts of the text that most contributed to computation of recommendations, (iii) or build detailed user profiles as a basis for collaborative recommendation algorithms.

Hence, with our work, we answer the following research questions (RQs):

1. **Annotation Methods.** Which NLP methods are suitable for automatic annotation of short user texts asking for recommendations?
2. **Recommendation Accuracy.** How does recommendation accuracy of automatic annotations compare with accuracy of high-quality manual annotations?
3. **Annotation Importance.** Which annotations are most important for computing accurate recommendations?

To answer these research questions we use movie recommendations as a case study. Specifically, we compute recommendations from suggestion requests posted on the online discussion platform reddit[3] (subreddit r/MovieSuggestions[4]). A typical user post can look as follows: "*I really like the story of the new Tom Cruise movie, Oblivion. Could anyone suggest me more movies like that?*". Hence, we start by annotating keywords, genres, movie titles or names of actors. For our example, we can annotate *Tom Cruise* as a named entity (i.e., actor name), *Oblivion* as a named entity (i.e., movie title), and *story* as a keyword (cf. Fig. 1 for another more detailed example). Also, all these annotations have a positive sentiment, as the user stated that they liked the movie and the actor. We can then proceed and use these annotations to compute and rank recommendations for the user.

For evaluation of NLP annotations and their recommendations in our experiment, we use a collection of manual annotations and user recommendations obtained with a crowdworker experiment (Eberhard et al., 2019). More specifically, for the automatic annotations, we assess multiple keyword extraction (KE) and deep learning (DL) models commonly used for named entity recognition (NER) (**RQ 1**). Also, we systematically assess the recommendation accuracy with automatic annotations by comparing them with the user and recommendations computed from the crowdworker annotations (**RQ 2**). Lastly, we evaluate the relative importance of various annotations with an ablation study by leaving-out/including different sets of annotations while computing recommendations (**RQ 3**).

In our dataset, we obtain the best performance with DL models trained for NER in combination with external embedding architectures. Further, the recommendation accuracy of an NLP ensemble method combining best performing NLP models for individual annotation types (i.e., keywords, movie titles, actor names) is comparable with the recommendations computed from manual annotations. Also, the sentiment of the automatically extracted annotations does not significantly improve recommendation performance as compared to manually labeled data with sentiment. Finally, our results suggest that movie titles and movie keywords are the most important annotations for accurate recommendations.

With our work, we provide a solid foundation for practitioners and researchers of recommender systems to advance the developments in narrative-driven recommendation scenarios. Specifically, we provide insights into which NLP models and configurations work best for automatically annotating free text to compute (movie) recommendations, hence substantially reducing the search space for combinations of NLP and recommendation algorithms (**RQ 1**). Further, we show that the recommendation accuracy with automatic annotations is comparable to the accuracy obtained with high-quality manual annotations, which makes a strong case for further investigations and developments in recommendation approaches for free-form text (**RQ 2**). Finally, we provide insight into relative importance of different types of annotations for computing accurate movie recommendations from free-form text (**RQ 3**).

2. Related work

2.1. Recommender systems

Recommender systems have become an integral part of our society and can be found in nearly every modern (online) application (Lamprecht et al., 2015; Pereira et al., 2018; Schedl et al., 2018; Smith & Linden, 2017). Their goal is to help users find and uncover items, services, or content that they are potentially interested in Ricci et al. (2011). Traditional research in this field focuses on algorithmic developments in processing user historic data and their profiles to compute recommendations (Christakou et al., 2007; Ghosh et al., 1999; Mak et al., 2003; Perny & Zucker, 2001). In general, Adomavicius and Tuzhilin (2005) differentiate in their work between collaborative, content-based, and hybrid approaches for computing recommendations.

Collaborative Filtering. Collaborative filtering extract patterns and preferences shared among users (Terveen & Hill, 2001). To make predictions about the preferences of the current user, collaborative filtering employs preferences of other users that are similar to the current user. The basic idea is that if users share preferences for items then this will be observed in these users rating those items similarly (Adomavicius & Tuzhilin, 2005). In contrast to such user-based algorithms, item-based collaborative filtering measures similarities between items based on user ratings. Typical similarity metrics that are used in such scenarios are cosine similarity or Jaccard similarity, which quantify and measure the overlap of attributes/features between items and/or users.

More advanced collaborative filtering approach to extracting user preferences for recommendation systems is matrix factorization. Matrix factorization aims at decomposing the user-item interaction matrix into two separate matrices (i.e., a user and an item matrix) that relate users, respectively items to a set of latent factors. With this method

[2] https://www.reddit.com/r/MovieSuggestions/comments/ssuhu
[3] https://www.reddit.com/
[4] https://www.reddit.com/r/MovieSuggestions/

it is possible to generate recommendations for every user-item pair by computing the product of the two matrices (Koren, 2008).

Neural Embeddings. Recently, neural networks and embedding approaches are being extensively used in collaborative filtering research, partly exhibiting outstanding performances (Cenikj & Gievska, 2020; Musto et al., 2015; Ozsoy, 2016; Polignano et al., 2021; Stiebellehner et al., 2018; Wang et al., 2019). Several such approaches are based on the neural probabilistic language model `word2vec`, which Mikolov, Chen, et al. (2013) introduced for extraction of high-quality embeddings for words in texts. In such models each word is mapped to a unique vector (Levy et al., 2015; Mikolov, Sutskever, et al., 2013; Mnih & Hinton, 2008). The `doc2vec` extension of `word2vec` introduced by Le and Mikolov (2014) allows vector representations of complete paragraphs and documents. The applications of language models in recommendation are numerous. For example, in their job posting recommendation scenario, Elsafty et al. (2018) showed that `doc2vec` outperforms `word2vec` as well as the well-established content-based recommender approach called term frequency–inverse document frequency (TF–IDF) when using job titles combined with full-text job descriptions. Stiebellehner et al. (2018) applied a `doc2vec` representation of users and items for calculating recommendations of mobile apps. The authors used textual app descriptions as well as app usage histories to learn vectors each representing a mobile app user out-performing other state-of-the-art algorithms.

In the field of recommender system `item2vec` was proposed by Barkan and Koenigstein (2016) to embed sequences of items in a collection of shopping baskets. Building upon this work, recommendation approaches based on `item2vec` were proposed in the last years revealing promising results (Feng et al., 2017; Liang et al., 2016; Wölbitsch et al., 2019).

More recent work by Wang et al. (2019) also shows the effectiveness of graph embeddings in a recommendation scenario. They evaluated graph neural networks with label smoothness regularization on four real-world datasets of movie, book, music, and restaurant recommendations. Their method outperforms state-of-the-art baselines and achieves strong performance in cold-start scenarios as well. Furthermore, Cenikj and Gievska (2020) propose to boost recommender systems with advanced embedding models. They ascertained the potential benefits of merging a language representation model with node embeddings generated by GraphSage (Hamilton et al., 2017) for improving the quality of product recommendations on Amazon. In their work, Janchevski and Gievska (2019) used `word2vec` and the graph embedding approach `node2vec` (Grover & Leskovec, 2016) for generating text and node embeddings. They combined natural language processing of user profiles and user generated text with graph embeddings to outperform traditional models for subreddit recommendations.

Content-Based Approaches. In contrast to collaborative filtering, content-based approaches rely on similarities between items based on their features, such as actors and directors of movies, genres of songs, or referenced politicians and other features in news articles (Okura et al., 2017). Fu and Ma (2021) proposed an online marketing recommendation algorithm based on the integration of content and collaborative filtering. They use the content-based methods to discover existing interests of users. They achieve better results than traditional content-based methods in terms of accuracy, recall, and diversity. The work of Fessahaye et al. (2019) focuses on an approach to improving music recommendation systems. Their algorithm uses a hybrid of content-based and collaborative filtering as input to a DL classification model to produce accurate recommendations with real-time prediction. They applied their approach on data from Spotify obtaining highly promising results.

This Work. In this paper, for our two-staged experiment, we apply and evaluate several state-of-the-art recommender approaches in the second stage, including collaborative filtering, content-based approaches as well as neural embeddings using `doc2vec`.

2.2. Context-aware recommender systems

Besides user profiles and histories, context-aware recommender systems employ contextual information to generate recommendations that fit the current needs of the user. Contextual information can be, for example, the location or interests of a user in a specific situation, the time of the day, or conditions which influence the decisions of a user (e.g., weather or physical condition). For example, Adomavicius et al. (2005) investigated the effectiveness of contextual information for movie recommendations. They suggested to exploit the information when, where, and with whom a movie was seen to improve the recommendations. In another example, Hariri et al. (2013) used popular tags from social networks as additional information for article and music recommendations. In a study using 17 different contextual parameters Oku et al. (2006), the authors utilized information, such as car parking possibilities, non-smoking sections, live concerts, or whether a restaurant has an ocean view, to improve recommendations for restaurants. A convincing work with the goal of helping practitioners and researchers understand how contextual information can be effectively combined with recommendation mechanisms was published by Villegas et al. (2018). The authors conducted a meta-analysis of 87 context-aware recommender system papers about content-based, collaborative filtering and hybrid approaches. The main results provide a clear understanding about where context information is usually integrated into the recommendation process, available techniques to exploit context information, and the most common used evaluation mechanisms, including properties, metrics, and protocols.

Session-Based Recommendations. In recent years, session-based recommendations as a specific type of context-aware recommendation have gained a lot of attention from our community. The goal of such session-based recommendation systems is to predict immediate next actions of a user given past actions from an ongoing session (Kouki et al., 2020; de Souza Pereira Moreira et al., 2021). The studies of Ludewig and Jannach (2018) and Wang et al. (2021) evaluated the most recent approaches in the field of session-based recommendations. Their results indicate comparable and in some cases even significantly better performance of session-based methods as compared to more complex approaches based on, for example, deep neural networks suggesting the importance of contextual information when computing recommendations.

Hence, while in the wider domain of context-aware recommender systems we understand fairly well how to use contextual information to improve recommendations, we still do not understand fully how to leverage free-form narratives describing items and preferences in recommender systems. Such systems are, however, sought after in many practical applications, such as interactive recommendation bots for online boards or social media. One such scenario is based on users providing unstructured texts to discuss and inquire recommendations with and from peers. This specific situation is usually referred to as *narrative-driven recommendation scenario* as proposed by Bogers and Koolen (2017). Therefore, in this paper we set out to analyze how to process and annotate relevant information from free-form texts that can be utilized in a downstream recommendation task.

2.3. Narrative-driven recommender systems

Narrative-driven recommendations are related to conversational-based recommender systems (Christakopoulou et al., 2016; Li et al., 2020; Montazeralghaem et al., 2021; Penha & Hauff, 2020), where the algorithm of a recommender system establishes a dialog-driven conversation with the user, or—in a more traditional way—where users ask for suggestions in a community and other users then come up with suggestions and possible explanations for their choices. In the recommendation calculation process, a narrative explanation of the current recommendation needs of a user serves as contextual information (Bogers

& Koolen, 2017; Eberhard et al., 2020, 2019). However, traditional recommender systems still struggle to automatically transform and map annotations or intuitions about desired entities (i.e., movie titles) from free-form narratives to relevant recommendations.

As a result, there exist many online message boards across different domains, allowing users to ask the community, for example, for game,[5] movie,[6] or music[7] suggestions, by describing their current preferences and what they are looking for. Other users then offer relevant recommendations, taking these provided narratives into account. In our previous work (Eberhard et al., 2019) about narrative-driven movie recommendations we created a gold-standard dataset with annotations manually labeled by crowdworkers from reddit submissions, where users asked the community for movie suggestions by providing such narratives (cf. Fig. 1 for an example). Using this dataset, we evaluated the performance of well-established algorithmic recommender systems against human suggestions of the reddit community. Although Glenski and Weninger (2017) showed that simple models are able to predict user interactions, such as votes, clicks, likes or views, on reddit, recommending movies based on narrative requests on reddit exposed to be a hard problem (Eberhard et al., 2019). The results of the analysis of this problem in our previous work (Eberhard et al., 2020) revealed that suggestions from the reddit community are oftentimes more diverse than the requests, making the automatic generation of suitable movie recommendations on reddit a challenging task. A limitation of this work is that we made use of crowdworkers to annotate useful information from the narrative requests to generate their recommendations instead of automatically processing and annotating valuable data from the free text.

This Work. The most recent work—to the best of our knowledge—in the research field of narratives in the course of the movie domain was proposed by Musto et al. (2022). The authors focus on exploiting narratives for automatically recognizing and extracting objective and subjective user preferences from narrative descriptions of user needs obtained by a questionnaire. In contrast to that, in this paper we deal with observational data from reddit and extract user preferences solely from their short free-form texts.

2.4. Annotating free-form text

In this paper, in the first stage of our experiment we automatically annotate relevant entities (e.g., movie titles, genres, actors) and keywords from free-form reddit narratives and determine whether they were mentioned in a positive or negative context. Therefore, we implement and evaluate several state-of-the art KE and DL methods for NER. We list here the most important ones.

Keyword Extraction. One simple and efficient algorithm for KE is Rapid Automatic Keyword Extraction (RAKE) (Rose et al., 2010). Its efficiency, low complexity, as well as the different way of approaching KE, are all factors in deciding to use this algorithm in our work. In 2004, Mihalcea and Tarau (2004) proposed a novel solution for the problem of KE by representing the input text as graph called TextRank. The algorithm does not require any deep linguistic or domain knowledge to run and is fairly simple to implement. We use this algorithm in addition to RAKE as a computationally inexpensive baseline for the neural network models explained later in this section.

Named Entity Recognition. The past two decades yielded an extensive number of research works and systems for the NER task. Early approaches are either rule-based, statistical, or hybrid, while recent works are based on language modeling with DL-based architectures (Zitouni, 2014). Rule-based and statistical approaches involve extraction of lexical and domain-specific features, thus classifying entities into categories with one-layered machine learning classifiers (Mansouri et al., 2008). Most common features include: syntactic features, part-of-speech tags, common n-gram features, and additional lexical features (Ashwini & Choi, 2014; Chieu & Ng, 2003; Gundogdu et al., 2018; Ritter et al., 2011). With the rise of neural network architectures there were significant improvements in the way NER is treated. As a replacement to previous heavily engineered features, DL methods, such as LSTMs (feature-based) or BERT (fine-tuning), were able to achieve impressive results with minimal domain-specific knowledge needed (Lample et al., 2016). Feature-based DL architectures perform by combination of various features, for instance, pre-trained word embeddings with lexical features, and passing them through a (most often) LSTM architecture, as described by Hochreiter and Schmidhuber (1997), that outputs token-level class labels. These architectures are accompanied by an extra classification layer on top of the network (Lample et al., 2016; Panchendrarajan & Amaresan, 2019). Fine-tuning approaches (BERT), on the other hand, require no feature engineering and work effectively with limited amounts of labeled data (Bhattacharjee et al., 2020).

At the highest level, there are two ways to approach NER with DL methodologies: (i) by passing target text to existing NER solutions, and (ii) by performing preprocessing and feature extraction steps and training task-specific DL classifiers. The first possibility of handling NER is with the usage of NER tools implemented within NLP libraries, consisting of pre-trained entity recognizers on large corpuses. One popular library that offers handling of NER is SpaCy,[8] an open-source industrial-strength NLP library implemented in Python, which offers multiple functionalities for text processing, tokenization, visualization, etc. These tools have been tested against the state-of-the-art benchmarks and have shown to achieve high accuracy[9] in NER tasks. Its computational speed when handling large volumes of data, along with the ability to re-train models or use separate components of its language modeling pipelines make it useful for usage solely as a NER tool as well as a feature-extraction tool for our latter approaches.

The second option is creating task-specific NER pipelines from scratch. We focused on two main DL approaches. One is BERT, a powerful language representation model used for a variety of complex NLP tasks and has achieved state-of-the-art results in most NLP sub-fields. The model takes tokenized sentences as input and the parameters are fine-tuned in an end-to-end manner, which makes it convenient for any downstream task (Devlin et al., 2018). In this paper, we use BERT for identifying and labeling of a custom-defined set of entities (e.g., movie titles, actors, etc.) in a token-wise fashion. Later research works offer modifications of the basic BERT architecture which vary in different parameters (e.g., number of layers, trainable hyperparameters, training sets). For our experiments we choose RoBERTa by Liu et al. (2019) and XLM-RoBERTa by Conneau et al. (2020). In addition, we also explore the BiLSTMs (Bidirectional Long Short-Term Memory Neural Networks) which are designed for complex NLP tasks, and as such, they handle NER successfully, as shown by Chiu and Nichols in their work from 2016. BiLSTMs are able to classify based on a combination of input features. As word embedding features, we use deep contextualized word representations from Embeddings from Language Models (ELMO), a sophisticated language modeling technique which captures the context meaning of a word, along with its complex characteristics, such as syntax and semantics (Peters et al., 2018). We retrieve the other feature groups that we use in our experiments with the SpaCy and NLTK packages for text processing.

Sentiment Analysis. Aspect-based sentiment analysis (ABSA) is an NLP subtask that inherits its challenges from NER, with the additional task

[5] https://www.reddit.com/r/gamingsuggestions
[6] https://www.reddit.com/r/MovieSuggestions
[7] https://www.reddit.com/r/musicsuggestions
[8] https://spacy.io/
[9] https://spacy.io/usage/facts-figures

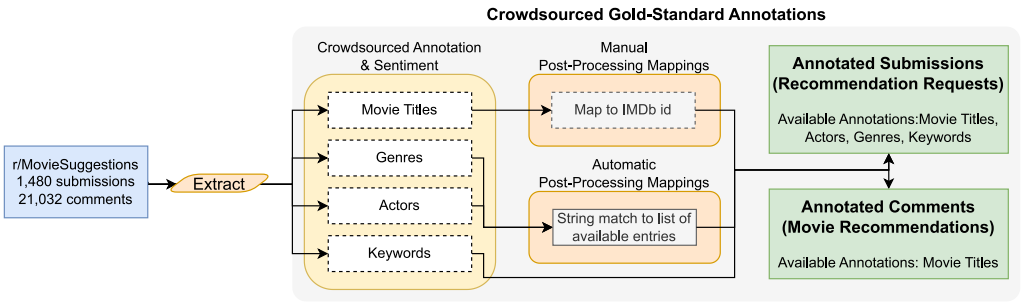

Fig. 2. Data Preparation. This figure depicts the data processing pipeline for creating the manually labeled gold-standard annotations that we use in this paper. Crowdworkers annotated movie titles, genres, actor names and keywords from r/MovieSuggestions submissions as well as movie titles from comments (i.e., suggestions from other users), including their sentiment. For our automatically extracted annotations, we use NER, KE approaches and sentiment detection to automate the work conducted by crowdworkers for the gold-standard annotations.

of classifying sentiments expressed towards an identified entity into positive or negative.

Related works have shown that heavily relying on hand-crafted features in ABSA does not lead to optimal results, which are rather achieved by combining features with word representations as contextual embeddings, obtained with pre-trained DL architectures (i.e., BERT or ELMO) (Do et al., 2019; Li et al., 2019; Truşcă et al., 2020). Aspect-based sentiment analysis is performed either as a sequence of subtasks, where NER and the target sentiment analysis are separate subtasks, or is solved in an end-to-end fashion (Hu et al., 2019; Li et al., 2021). In our experiments, we opt for the latter as the previous findings show better performance in this case. For instance, Cai et al. (2020) explain that treating ABSA as separate subtasks fails to capture the implicit relationships between aspects and their corresponding sentiments, while Liu et al. (2020) describe how systems that treat ABSA subtasks simultaneously are superior at modeling the connections between the target and the sentiment. Zeng et al. (2019) discuss how applying one model that unifies the label space for ABSA rather than using a pipeline of models is more practical in real-world applications. Example are systems where the response time to the user is important and separate models increase this time, as discussed by Vazan and Razmara (2021) in their work.

Sequence Labeling Format. The extraction of entities from sentences is usually performed by labeling the sentences "token-wise". Typically, the assigned labels indicate the boundary and entity type of the current token (Jurafsky & Martin, 2009). In this process, the BIO labeling scheme, proposed by Ramshaw and Marcus (1999), is the most commonly used format with three types of labels that can be assigned to a token: B (begin of an entity), I (inside an entity), and O (outside of any entity). Following our previous discussion on ABSA, we have opted for a unifying label strategy, similar to the one of Zeng et al. (2019), that captures all information for a word in one label (border, entity and sentiment), especially for its simplicity and ease of maintenance for practical solutions.

This Work. In contrast to these previous studies, in this paper we focus on creating a fully automated recommendation framework. We build upon our previous work in narrative-driven recommendations and a large body of literature in NLP. Hence, we do not build new NLP methods but evaluate the existing ones for their efficiency for narrative-driven recommendations. In particular, we use the manually labeled reddit dataset to train NLP models which we then apply to automatically annotate important entities in narratives. Moreover, we analyze the importance of the types of annotation to get a better understanding of the impact of the annotations for the final recommendation quality.

3. Materials & experiments

Dataset. For our experiments, we use a crowdworker-created dataset (Eberhard et al., 2019), which consists of annotated submissions and comments from reddit (Baumgartner et al., 2020), more specifically from the subreddit r/MovieSuggestions. Particularly, the dataset contains reddit submission IDs, titles, and original texts, as well as, annotations from crowdworkers including keywords, movie titles, actor names, and genres. In addition, all annotations have sentiment (positive or negative) depending on whether the users liked or disliked a particular entity in their submissions. Lastly, we also link all movie titles with their corresponding IDs from the Internet Movie Database[10] (IMDb). In total, we have 1480 submissions with 1908 unique movies and 21,032 comments including more than 43,000 individual suggestions and 6071 unique movies as suggestions. Fig. 2 depicts the data preparation process.

Each recommendation request in the dataset includes one or more positively mentioned movies, which are examples of movies that the user liked before. Moreover, requests often include additional information, such as negatively mentioned movies (i.e., movies that the user did not like before), positively and negatively mentioned keywords (describing further aspects of the movies), as well as genres, and actors. We list all the details of our dataset in Table 1.

Table 1
Reddit dataset statistics.

#Requests	1,480
#Request Authors	1,244
#Movies in Requests	5,521
#Unique Movies in Requests	1,908
#Requests With Positive Movies	1,480
#Requests With Negative Movies	77
#Keywords in Requests (Without Common Words)	4,492 (3,947)
#Unique Keywords in Requests (Without Common Words)	1,878 (1,762)
#Requests With Positive Keywords	1,202
#Requests With Negative Keywords	152
#Genres in Requests	762
#Unique Genres in Requests	25
#Requests With Positive Genres	459
#Requests With Negative Genres	55
#Actors in Requests	100
#Unique Actors in Requests	79
#Requests With Positive Actors	73
#Requests With Negative Actors	7
#Suggestions	43,402
#Unique Suggestions	6,071
#Suggestion Authors	7,431
Average #Suggestions per Request	29.33
Average Duration Between Request and Suggestion	31 h 41 min

This table shows the statistics of the manually labeled reddit dataset (Eberhard et al., 2020, 2019).

[10] https://www.imdb.com

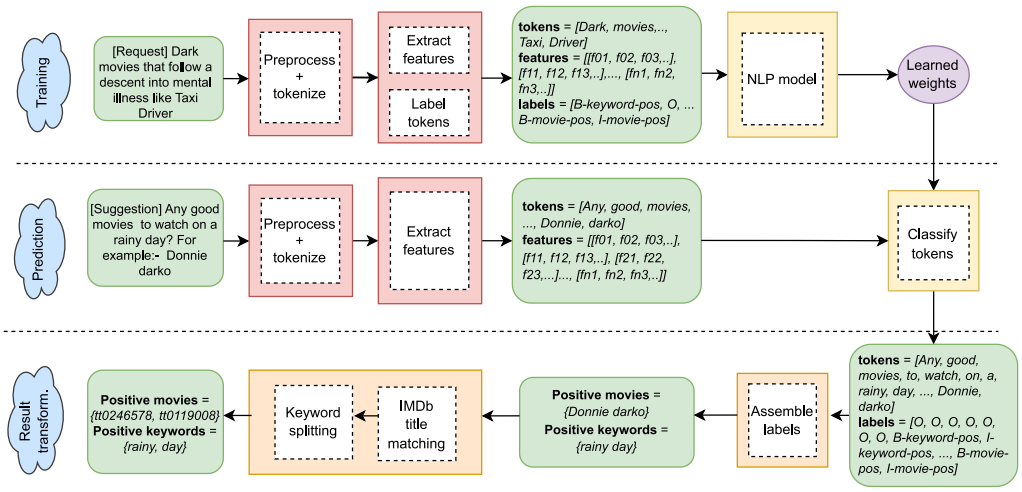

Fig. 3. Process Flow of Automatic Annotation. This figure shows the entire process flow of extracting entities and keywords with NLP pipeline. The figure is divided in three subsections. The topmost shows the training process from the moment the submission enters the process to the moment it exits the preprocessing steps and enters the NLP model. The model then outputs the learned weights and can be used to label new test submissions which undergo the same preprocessing flow (apart from token labeling). The model outputs a set of tokenized sentences and the corresponding label for each token. This output enters the final stage of transformation (in the last subsection) where consecutive labels are identified and their corresponding tokens are assembled into words and phrases. As a final step, the titles are matched with their IMDb identifier.

We show a typical example of such a request from our dataset in Fig. 1, in which crowdworkers annotated three positive movies (i.e., *Secret Window*, *Stranger Than Fiction*, *The Ghost Writer*), one negative genre (i.e., *horror*), three positive keywords (i.e., *writing*, *writers*, *inspirational*), and one positive actor (i.e., *Sean Connery*). We also show two examples of comments, each of them having a single suggestion, the movie *Adaptation* in the first and the movie *Finding Forrester* in the second one.

Experiments. In this paper, we conduct our experiments in the following two stages:

1. First, we train KE and NER models by processing free-form text contributions (i.e., the recommendation requests and replies to these). We evaluate the annotation accuracy of the individual approaches using our manually labeled reddit dataset as a gold-standard.
2. Second, we evaluate the performance of the best KE and NER models (and their ensembles) from the first stage by using their annotations as input for several state-of-the-art recommendation algorithms. Also, for this evaluation we use our gold-standard dataset.

With our two-staged experimental setup we specifically aim at evaluating intermediate results produced by NLP to gain insights into which NLP methods are well suited for such recommendation scenarios. These new insights may result in a substantial reduction of the number of possible combinations of NLP methods and recommendation algorithms in future developments, e.g., by reducing the complexity and computation time of downstream hyperparameter optimization tasks. Hence, with our experiments we lay a solid foundation for future research as well as practical implementations in the field of narrative-driven recommendations.

4. Automatic annotations

In this section, we describe our experiments with various state-of-the-art NLP models for automatically annotating short, free-form texts. To this end, we develop two general scenarios: (i) we use combinations of KE and pre-trained NER models in a transfer learning setting with minimal preprocessing (NLP baselines) and, (ii) we build complete DL pipelines with several distinct processing and training stages (NLP pipelines). On the one hand, NLP baselines should provide us with general insights on suitability of off-the-shelf NLP methods for annotations of free-form recommendation requests for movies. Potentially, these results and insights can be transferred to further domains such as music, games, or books. On the other hand, NLP pipelines are domain-specific models, requiring extensive training but optimized for a specific task and domain such as movies. Here, processing and training steps, as well as, DL architectures can be possibly transferred to additional domains.

NLP Baselines. We opt for minimal preprocessing and only remove accents, URLs, markdown, non-latin characters, and stopwords. We pass the preprocessed submissions to existing KE and NER solutions, which then output the relevant keywords and entities. We extract and evaluate each annotation type separately. We use the following methods: RAKE (Rose et al., 2010) and TextRank (Mihalcea & Tarau, 2004) for keyword retrieval and SpaCy's pre-trained large English NER model[11] for actor names and movie titles (we use its 'person' and 'work-of-art' entities as a replacement for 'actor' and 'movie'). In addition, for genre identification, we use the IMDb's list of 24 movie genres and perform simple word matching on the submissions. The baseline algorithms we have selected have one common characteristic—they all behave as a black box, i.e., they accept raw text (or in our case minimally processed text) and output annotations without model training or text transformation.

NLP Pipelines. For each NLP approach that we analyze, we build and train a classification pipeline for extraction of all relevant annotations including movie titles, actors, keywords, and genres. The pipeline consists of five stages: (i) preprocessing of the submissions text, (ii) feature extraction, (iii) class labeling, (iv) classification of labeled data, and (v) assembly and annotations matching. Fig. 3 visually represents the entire flow of a submission through an NLP pipeline, including both the training and testing/inference processes.

Preprocessing. We first concatenate the submission title and text and remove URLs/HTML elements with regular expressions. Next, we remove upper and lower case variations of the words "Request" and "Suggestion" at the beginning of the submission title. We keep stopwords, numbers, and punctuation, as they are frequently contained in

[11] https://spacy.io/models/en#en_core_web_lg

Table 2
Classifier Input After Preprocessing, Feature Extraction, and Labeling.

ID	Token/Word	POS	TF	isAlpha	isStopword	Label
v830o	A	90	.00209	1	1	O
v830o	psychological	84	.00025	1	0	B-keyword-pos
v830o	Thriller	92	.000053	1	0	B-genre-pos
v830o	in	85	.00823	1	1	O
v830o	the	90	.02118	1	1	O
v830o	vein	92	.000169	1	0	O
v830o	of	85	.017237	1	1	O
v830o	Se7en	93	.0002217	0	0	B-movie-pos

We depict the results of preprocessing, feature extraction and labeling on the first eight tokens from a submission (cf. Fig. 4) from our dataset. Such results serve as input to our DL classifiers. We also show a selection of features that we extract including: POS tags (enumerated by SpaCy), term frequency, and isAlpha/isStopword (is the token alphanumerical or a stopword). Note that we extract further features, which we do not depict here due to the space restrictions. These features include word embeddings, sentiment and further lexical/syntactical features. Finally, the last column shows the assigned labels. These labels represent the classification target for the classifier, which is trained in the next step of NLP pipelines.

movie titles. Lastly, we tokenize the submissions with the NLTK regular expression tokenizer[12] preserving the punctuation.

Feature Extraction. In this stage, we extract a total of 147 hand-picked features and 1024 deep contextual (ELMO) features for each token. In particular, we extract the following feature groups:

(i) Lexical and syntactic features describing the tokens in terms of their position in the sentence, their syntactical dependencies to the other tokens, their format, and their lexical categories. These features include: word root form (lemma), syntactic parent (based on syntactic dependency parser[13]), POS tag, language, Brown cluster (Brown et al., 1992), as well as boolean-value features that indicate whether the token is alphanumerical, ascii, upper/lower case, brackets, and so on. The complete list of features are listed in the SpaCy documentation.[14]
(ii) Contextual embeddings from language models, i.e., ELMO[15] for each word/token.
(iii) Context-sensitive tensors (or dense word vector representations[16]) are word embeddings provided by SpaCy's small ("_sm") English models[17]. Note: We use SpaCy's large English model in our baseline models, but the small version for computing features for the NLP pipelines.
(iv) Sentiment features, which we calculate with Vader.[18] We extract positive, negative, neutral and compound sentiment scores for each token, as well as for its neighboring tokens. As an additional feature, we also compute the ratio between the compound score of the current word and the current submission.
(v) TF–IDF: we calculate the frequency and importance of the token in the dataset. We compute TF as the total number of occurrences divided by the total number of tokens in the corpus and IDF as the logarithm of the total number of submissions divided by the number of submissions that contain the token.

Labeling. We apply Beginning, Inside and Outside (BIO) token labeling scheme on each annotation type, e.g., B-movie, I-movie, B-actor, and so on. To represent the token sentiment, we extend the labels with a sentiment post-fix. For example, for this part of a sentence "*[...] loved the new Top Gun [...]*", we first obtain the following tokens [loved, the, new, Top, Gun] and then label them as [O, O, B-keyword-pos, B-movie-pos, I-movie-pos] for Outside, Outside, Beginning of a positive keyword, Beginning of a positive movie, and Inside of a positive movie.

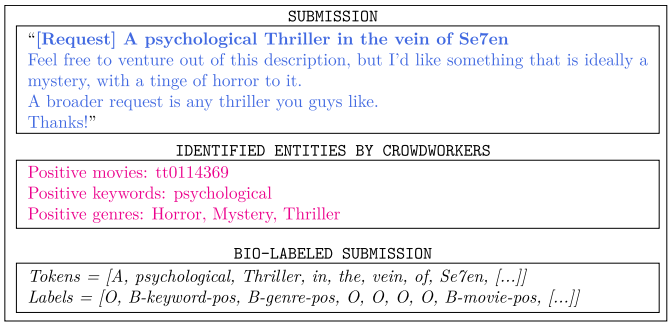

Fig. 4. Annotation Example. An example of a Request[19] where the crowdworkers identified one movie and keyword and three genres, all positive. Based on these annotations, our labeling component created a BIO label for each token of the tokenized submission. Note: The original annotation contains the movie IDs. We first retrieve the actual title and then we perform the labeling.

We label the data automatically by using the crowdworker annotations from our dataset. In particular, for every token:

(i) If the token matches any annotation from crowdworkers regardless its position, we label it with the corresponding type (e.g., actor), otherwise we label it as 'O'.
(ii) We add the corresponding sentiment post-fix if the crowdworkers annotated sentiment.
(iii) If we labeled the current token with an annotation type, we check its predecessor to add positional labels. Hence, if the predecessor is labeled as 'O' or a different type, we assign 'B' to the current token. Otherwise, if the predecessor has the same type, we additionally label the current token with 'I'.
(iv) We perform a second pass to correct possibly wrongly labeled tokens based on a set of pre-defined rules. For example, if in the first pass we labeled "is" with an annotation label such as movie, we check whether the neighboring tokens are also labeled with movie labels. If that is not the case, we change the label of "is" to 'O'. We keep a manually curated list of such stopwords that frequently get wrongly labeled with annotation labels for the second labeling pass.

Fig. 4 shows the comparison between an original submission and crowdworker annotations against the tokenized submission and transformed labels to BIO format. The extracted submission features are depicted in Table 2.

Classifiers. Using the labeled data we build several token classification models. The classification models can be divided into two groups,

[12] https://www.nltk.org/api/nltk.tokenize.regexp.html
[13] https://spacy.io/api/dependencyparser
[14] https://spacy.io/api/token
[15] https://tfhub.dev/google/elmo/3
[16] https://spacy.io/api/doc
[17] https://spacy.io/usage/spacy-101/
[18] https://www.nltk.org/api/nltk.sentiment.vader.html
[19] https://www.reddit.com/r/MovieSuggestions/comments/v830o

Table 3
Baseline Results.

Model \ Metric	Precision	Recall	F1 score
RAKE (*Keywords*)	0.073 [0.063 – 0.082]	0.869 [0.844 – 0.894]	0.134 [0.118 – 0.150]
SpaCy (*Movie Titles*)	0.758 [0.595 – 0.930]	0.067 [0.047 – 0.086]	0.123 [0.089 – 0.156]
SpaCy (*Actor Names*)	0.066 [0.022 – 0.106]	0.929 [0.857 – 1.000]	0.124 [0.049 – 0.196]
IMDb Match. (*Genres*)	0.675 [0.591 – 0.766]	0.468 [0.412 – 0.525]	0.552 [0.499 – 0.613]

The table shows scoring results for all four annotation types. We list the averages and the bootstrapped confidence intervals.

Table 4
Results Associated With Positive Entities.

Model \ Metric	Precision	Recall	F1 score
Keywords			
BiLSTM (ELMO)	0.514 [0.467 – 0.561]	0.498 [0.442 – 0.552]	0.506 [0.465 – 0.548]
RoBERTa large	0.502 [0.454 – 0.550]	0.413 [0.359 – 0.466]	0.454 [0.411 – 0.498]
BERT base multilingual cased	0.598 [0.536 – 0.661]	0.301 [0.255 – 0.348]	0.400 [0.352 – 0.450]
BERT large cased	0.598 [0.542 – 0.654]	0.405 [0.353 – 0.455]	0.483 [0.436 – 0.531]
Genres			
BiLSTM (ELMO)	0.775 [0.699 – 0.858]	0.723 [0.647 – 0.805]	0.748 [0.690 – 0.816]
RoBERTa large	0.735 [0.590 – 0.885]	0.181 [0.109 – 0.248]	0.291 [0.196 – 0.387]
BERT base multilingual cased	0.763 [0.686 – 0.847]	0.613 [0.532 – 0.698]	0.680 [0.616 – 0.753]
BERT large cased	0.797 [0.724 – 0.875]	0.662 [0.583 – 0.745]	0.723 [0.663 – 0.793]
Actors			
BiLSTM (ELMO)	0.156 [0.000 – 0.282]	0.294 [0.017 – 0.526]	0.204 [0.017 – 0.367]
RoBERTa large	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]
BERT base multilingual cased	0.353 [0.063 – 0.581]	0.400 [0.133 – 0.646]	0.375 [0.135 – 0.607]
BERT large cased	0.279 [0.010 – 0.465]	0.333 [0.128 – 0.524]	0.303 [0.106 – 0.488]
Movies			
BiLSTM (ELMO)	0.798 [0.754 – 0.843]	0.673 [0.623 – 0.721]	0.730 [0.692 – 0.768]
RoBERTa large	0.773 [0.728 – 0.818]	0.476 [0.415 – 0.536]	0.589 [0.539 – 0.641]
BERT base multilingual cased	0.716 [0.675 – 0.760]	0.438 [0.398 – 0.479]	0.544 [0.507 – 0.582]
BERT large cased	0.700 [0.650 – 0.750]	0.428 [0.390 – 0.465]	0.531 [0.496 – 0.567]

feature-based and fine-tuned models. Feature-based models consist of several BiLSTM-based models built in Keras[20] differing in their architectures, while for the second group of models we use a fine-tuning approach of several pre-trained BERT-based models.

Each feature-based model is built in a similar manner with an equal number of levels. However, at each level we add a different number of layers depending on the number of features. We experiment with different features in terms of groups, for instance, ELMO features and hand-picked features, only hand-picked features, and so on. Each feature group gets processed in a separate flow of stacked layers before they are passed to a concatenation layer prior to a BiLSTM layer. Finally, for each model, the output layer is a time distributed dense layer with a softmax activation function. After the feature concatenation layer and the BiLSTM layer, we experiment with dropout values in the range $[0.25, 0.75]$ in steps of 0.05.

For the second group of models we fine-tune several BERT-like implementations using the Hugging Face.[21] We extend and fine-tune these models by adding the token clasifier at the top of each individual model. Specifically, we use the following architectures: BERT base (cased and uncased), BERT large (cased and uncased), BERT base multilingual (cased and uncased) (Devlin et al., 2018), RoBERTa base and RoBERTa large (Liu et al., 2019) and XLM-RoBERTa base (Conneau et al., 2020). We train all of our fine-tuning models in 20 epochs.

Assembly and Annotations Matching. As our classification models only predict the BIO labels for tokens, in the final step we assemble annotations (with corresponding types and BIO labels) and perform a similarity matching for movie titles with movies from IMDb. We perform the similarity matching with a variant of the Ratcliff-Obershelp longest connected matching subsequence algorithm (Ratcliff & Metzener, 1988), which computes the similarity in the range $[0, 1]$. We experiment with several similarity thresholds from 0.6 to 0.9, selecting only those movie titles exceeding this similarity threshold.

Evaluation. For the evaluation of our models, we compare the automatically annotated entities with the manually labeled reddit annotations. We chronologically split our data for training (80%) and testing (20%). As our baseline models are pre-trained, training data is only used for training of NLP pipelines. However, the same test data is used for evaluation of both the baselines and the pipelines to ensure a fair comparison. Also, as we do not extract sentiment with our baselines, we treat all of the detected entities as positive during baselines evaluation.

We evaluate our pipelines in two stages. First, we evaluate the performance of classifiers at the level of individual tokens. Second, we select the best performing models from the previous stage to assembly and match the annotations. We then evaluate these final annotations in the same way as the baseline models, with the crucial difference that annotations from the pipelines also include sentiment. For consistency, we keep the same percentage of the reddit data for testing, but additionally we split the training set into training and validation sets for the pipelines. We do a random split, but keep the same seed for every model, so that we can have comparable validation results.

For all evaluation tasks we compute precision, recall, and F1 score using the crowdworked dataset as the ground truth. Since the number of annotations per submissions vary (i.e., imbalanced classes) we compute the micro average for each metric to weigh each annotation equally.

4.1. Results

Baseline Methods. In Table 3 we show the baseline results (only for positive annotations). For genre annotations, the simple matching of tokens against a set of IMDb genres made correct guesses in approximately 68% of cases. We apply SpaCy for annotating both, actor names and movie titles, and obtain opposing precision and recall scores. We hypothesize that this is due to the different structure of these

[20] https://keras.io/
[21] https://huggingface.co/

Table 5
Results associated with negative entities.

Model \ Metric	Precision	Recall	F1 score
Keywords			
BiLSTM (ELMO)	0.389 [0.148 – 0.653]	0.108 [0.020 – 0.185]	0.169 [0.052 – 0.287]
RoBERTa large	0.290 [0.117 – 0.508]	0.169 [0.032 – 0.303]	0.214 [0.078 – 0.379]
BERT base multilingual cased	0.286 [0.000 – 0.571]	0.031 [0.000 – 0.062]	0.056 [0.000 – 0.111]
BERT large cased	0.351 [0.153 – 0.536]	0.200 [0.091 – 0.300]	0.255 [0.142 – 0.380]
Genres			
BiLSTM (ELMO)	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]
RoBERTa large	0.400 [0.000 – 0.800]	0.143 [0.000 – 0.286]	0.211 [0.000 – 0.421]
BERT base multilingual cased	0.778 [0.556 – 0.956]	0.500 [0.231 – 0.769]	0.609 [0.417 – 0.865]
BERT large cased	0.818 [0.636 – 1.000]	0.643 [0.363 – 0.911]	0.720 [0.543 – 0.964]
Actors			
BiLSTM (ELMO)	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]
RoBERTa large	1.000 [1.000 – 1.000]	0.250 [0.000 – 0.500]	0.400 [0.000 – 0.800]
BERT base multilingual cased	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]
BERT large cased	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]
Movies			
BiLSTM (ELMO)	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]
RoBERTa large	0.900 [0.800 – 1.000]	0.214 [0.000 – 0.400]	0.346 [0.087 – 0.637]
BERT base multilingual cased	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]	0.000 [0.000 – 0.000]
BERT large cased	1.000 [1.000 – 1.000]	0.024 [0.000 – 0.048]	0.047 [0.013 – 0.093]

entities. In particular, we obtain a recall of 0.929 and precision of 0.066 for actors indicating a lot of false positives. As SpaCy recognizes the entity 'person' rather than 'actor' a lot of persons other than actors are also detected. On the other hand, movies obtain a precision of 0.758 and recall of 0.067 using SpaCy's label 'work-of-art'. We hypothesize that due to heterogeneity of movie titles, a little less than 10% are detected. When annotating keywords, we obtain a high recall (0.879) with substantial amount of false positives resulting in a precision of only 0.073. We obtain these results with RAKE, as shown in Table 3.

DL Pipeline Evaluation. In the first evaluation stage (precision, recall, and F1 score measured on token labeling), we evaluated 23 feature-based and 9 fine-tuning models. Out of those 32 classifiers, we select four classifiers with the highest average precision scores for all tokens to evaluate them in the final assembly and matching stage. Those four models include a BiLSTM with ELMO embeddings as features (feature-based), BERT large cased model (fine-tuned), BERT base multilingual cased (fine-tuned), and a RoBERTa large model (fine-tuned). See Appendix A for the average results on all our model configurations.

In Tables 4 and 5 we show the performance metrics for all annotations with DL methods. We start with the top section of both tables showing the results of annotating positive and negative keywords, respectively. While BERT large cased model has the best precision on positive keywords, BiLSTM (ELMO) outperforms the rest of the models with a recall of 0.498 obtaining also the highest F1 score of 0.506. This recall is significantly lower than the keywords baseline recall of 0.879, but the model reduces the false positives by approximately 40%. For negative keywords, the BERT large cased model scored the highest F1 score of 0.255, outperforming other models.

For the annotations of positive genres, all models achieve a high precision (> 0.73). BiLSTM (ELMO) outperforms the baseline and the rest of the models with recall of 0.723. However, BiLSTM (ELMO) fails to detect any negative keywords (resulting in precision and recall of 0). On the other hand, BERT large cased captures approximately 64% of all the negative genres and nearly 82% of its negative genre predictions were correct (cf. Table 5).

Overall, for the annotation of actors we obtain poor results. We hypothesize that this is due to a small sample size and a major class imbalance between positively and negatively mentioned actors (e.g., there were only 7 negatively mentioned actors; cf. Table 1). Nevertheless, BERT base multilingual cased scored higher than the rest of the models for detecting positive actors, whereas for the negative actors, RoBERTa large is the only model that made any predictions.

Finally, in Tables 4 and 5 we show the results for annotation of movie titles, both positive and negative in the bottom sections. We report the results that we obtain with the best performing similarity threshold of 0.7. As the number of positive movie titles is the largest of all annotation types resulting in a sufficiently large training dataset, we obtain better results than for annotation of actors. BiLSTM (ELMO) outperforms other models at annotating positive movies (F1 score 0.730) while RoBERTa large has the best results for negative movies (e.g., precision of 0.9 and recall of 0.214).

In summary, we observe that different models favor different annotation types and therefore we combine the best performing models for individual annotation types (in terms of F1 score) into a combined ensemble approach. In Table 6 we provide an overview of this ensemble approach with the NLP models used. In Fig. 5 we show an example of a submission and the annotations that we produce with our ensemble approach.

Table 6
Ensemble Approach.

Entity	Sentiment	Model
Movies	Positive	BiLSTM (ELMO)
	Negative	RoBERTa large
Keywords	Positive	BiLSTM (ELMO)
	Negative	BERT large cased
Genres	Positive	BiLSTM (ELMO)
	Negative	BERT large cased
Actors	Positive	BERT base multilingual cased
	Negative	RoBERTa large

This table shows the models that we use for our ensemble approach that we eventually use in the second stage of our experiments, the recommendation task. For each entity type and sentiment, we use the best performing NLP model from the first stage, respectively.

5. Movie recommendations

Similar to our previous work (Eberhard et al., 2019), we implement several recommender approaches including item-based collaborative filtering (CF), matrix factorization, TF–IDF, `doc2vec`, and a network-based approach that is built on a graph of actors. To that end, we collect user reviews and individual user ratings for all movies on IMDb in addition to the publicly available IMDb dataset. We only keep movies and discard all other types, such as TV series or episodes. To allow for fair comparisons between the different approaches and to minimize noise in the data, we only consider movies that have (i) more than 1000 user ratings, (ii) at least one user review, (iii) a movie description, and (iv) at least one person in the cast. We obtain the rating thresholds

SUBMISSION

"**[Request] Movies about writing/writers.**
Two of my favourites are *Secret Window* and *Stranger Than Fiction*. I also liked *The Ghost Writer*. [...] I'm not a fan of horror. I know there are probably a lot of 'inspirational' movies about writing out there (I vaguely recall one with Sean Connery?). [...]"

Fig. 5. Example of Entity Annotation by NLP Ensemble Approach. Based on the same reddit submission[22] as used in Fig. 1, this example shows the desired annotation by our ensemble approach. The goal is that the used KE and NER models annotate the following entities: three positive movies (i.e., *Secret Window*, *Stranger Than Fiction*, *The Ghost Writer*) annotated by BiLSTM (ELMO), a negative genre (i.e., *horror*) annotated by BERT large cased, several positive keywords (i.e., *writing*, *writers*, *inspirational*) annotated by BiLSTM (ELMO), and a positive actor name (i.e., *Sean Connery*) annotated by BERT base multilingual cased.

for movies (1000) and users (no limit) via hyperparameter grid search in our previous work (Eberhard et al., 2019). Further, to improve the overall performance of all implemented recommender approaches we compute centered ratings (Desrosiers & Karypis, 2011; Resnick et al., 1994) by removing user and item bias (Eberhard et al., 2019).

We generate recommendations by computing similarities between a movie given in the recommendation request and all other movies available in our dataset. As similarity measure we use either cosine similarity or Euclidean distance (determined via hyperparameter optimization in our previous work (Eberhard et al., 2019)). In cases where there is more than one input movie we aggregate similarity values as evaluated in our previous work (Eberhard et al., 2019).

For the rating-based approaches (i.e., matrix factorization and item-based CF) we leverage the centered user ratings to create an IMDb user-ratings matrix. We factorize the IMDb user-ratings matrix in a standard manner by minimizing a regularized squared error with a stochastic gradient descent (Funk, 2006). We then use cosine similarity to compute similarity between the obtained movie factors. For the item-based CF approach we use the IMDb user-ratings vectors of two movies to compute cosine similarity (Eberhard et al., 2019).

For the text-based recommender algorithms (i.e., TF–IDF and `doc2vec`) we leverage movie descriptions and user reviews from IMDb to find similar movies by their similarity. We compute the TF–IDF score (Salton & McGill, 1983) of terms in the movie description and user reviews for each movie. To compute the similarity between movies we use normalized TF–IDF vectors and the reciprocal of Euclidean distance (Eberhard et al., 2019). We use `doc2vec` to generate a document vector for each movie and use these vectors to compute cosine similarities between movies.

After training and optimizing our recommender approaches on IMDb data, we fine-tune them on our manually labeled reddit dataset (cf. Fig. 6). In particular, using the annotations from submissions as input (e.g., annotated movie titles) for IMDb recommendation engines, we first retrieve candidate recommendations (i.e., movies most similar to the input movies) and then use a set of post filters to re-rank these recommendations to better match user suggestions from a given submission. In addition to the algorithmic score (i.e., similarity values obtained from respective recommendation approach) of each candidate recommendation, we add several scores computed by re-ranking modifiers. In our fine-tuning step, we learn the optimal weights of these post filter scores by cross-validation as described in our previous work (Eberhard et al., 2019). Aggregating all these weighted scores for each candidate movie results in final recommendation scores that provide the order of the final recommendation list. In this paper, we apply the following re-ranking modifiers:

(i) *Popularity and rating modifier* increases IMDb recommendation scores proportional to the popularity of a movie on IMDb (i.e., how many users rated the movie on IMDb) and its average IMDb rating.
(ii) *Genre, year, and keyword modifier* to strengthen the similarity between movies with similar genres, plot keywords, or the years the movies were released, respectively based on IMDb data.
(iii) *Submission genre and keyword modifier* to produce better ranks in the final recommendation list for movies with genres or keywords provided in the narrative of the submission on reddit.

To learn the optimal modifier weights for fine-tuning, we conduct a grid search experiment over all combinations of modifier weights in the range $[0.0, 1.0]$ in steps of 0.2. We fine-tune the weights for manually as well as automatic NLP annotations using our NLP ensemble. To further optimize the final recommendation list we implement additional post filters as follows:

(i) *Prequel and sequel filter* to remove potential movies from the recommendation list that follow or are followed by a given movie in a movie franchise (e.g., if a user asks for recommendations similar to the first Lord of the Rings movie, the other episodes of the trilogy would be removed from the recommendation list).
(ii) *Submission year and actor filter* to filter out movies that do not fulfill the requirements a user requested in the narrative on reddit in terms of the produced year of a movie or involved actors in a movie.

Evaluation. We evaluate the performance of the fine-tuned crowdworker and NLP recommenders by comparing their recommendations with the user suggestions from reddit submissions. In addition to fully fine-tuned recommenders, we also evaluate a simple NLP baseline approach without sentiment (i.e., each entity is considered as positive) and without fine-tuning (i.e., we use the modified weights from crowdworker recommender based on `doc2vec`). For evaluation, we chronologically split the dataset in 80% for validation and hyperparameter optimization and 20% for testing.

To assess the relative importance of each annotation type we conduct an ablation study. Hence, in addition to the general experiment with all annotations, we also run four further experiments using (i) only movie titles, (ii) movie titles and genres, (iii) movie titles and keywords, and (iv) movie titles and actors as input for computation of recommendations. We repeat each of these experiments two times, once with sentiment and once without sentiment (i.e., we set the sentiment for all annotations to positive).

Lastly, as the NLP ensemble approach has a coverage of 90% (i.e., it annotates movie titles in 9 out of 10 submissions), for fair comparison with crowdworker recommender, we reduce our test set to include only covered submissions. For comparison we calculate common evaluation metrics for recommender systems (i.e., precision, recall, F1 score, nDCG, and MAP @10[23]) for each submission and report their overall means over the whole test set (i.e., macro average).

5.1. Results

Overall, the `doc2vec` algorithm outperforms all other algorithms. To save space, we describe detailed recommendation results only for this recommendation approach. The detailed results for the remaining algorithms are in Appendix C. The best performing `doc2vec` configuration uses vectors of dimension 500 (optimized over range $[100, 1000]$ in steps of 100) and cosine similarity (optimized among inverse Euclidean distance and cosine similarity) to compute similarities between input movies and movies from the IMDb (cf. Eberhard et al., 2019).

[22] https://www.reddit.com/r/MovieSuggestions/comments/ssuhu

[23] This means that recall@10 and F1 score@10 have an upper limit of 0.45 and 0.59 respectively, as the number of movie suggestions from the community varies per submission in the test set (i.e., 29.42 movie suggestions on average).

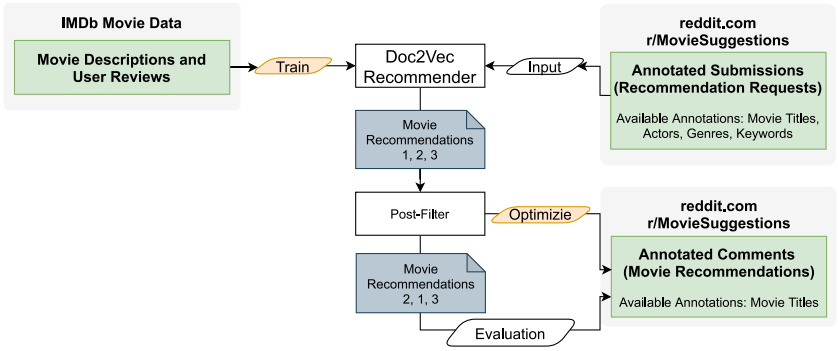

Fig. 6. Recommender Experiment. This figure depicts the recommendation experiment. In a first step, we use IMDb data to train our doc2vec model. Second, we use annotated submissions (i.e., automatically or manually from the crowdsourced gold-standard dataset) from r/MovieSuggestions and optimize post filters on the movies suggested by users on r/MovieSuggestions.

Table 7
Recommender results.

Entities	Sentiment	👥	[CI]	🤖	[CI]
Movies	all pos.	0.1174	[0.1033, 0.1309]	0.1148	[0.1008, 0.1283]
	pos./neg.	0.1185	[0.1044, 0.1320]	0.1148	[0.1008, 0.1283]
Movies | Genres	all pos.	0.1201	[0.1059, 0.1337]	0.1173	[0.1029, 0.1310]
	pos./neg.	0.1204	[0.1061, 0.1340]	0.1168	[0.1024, 0.1306]
Movies | Keywords	all pos.	0.1250	[0.1107, 0.1387]	0.1211	[0.1068, 0.1349]
	pos./neg.	0.1253	[0.1112, 0.1390]	0.1228	[0.1085, 0.1365]
Movies | Actors	all pos.	0.1152	[0.1016, 0.1284]	0.1145	[0.1004, 0.1280]
	pos./neg.	0.1183	[0.1044, 0.1316]	0.1145	[0.1004, 0.1280]
Movies | Genres | Keywords | Actors	all pos.	0.1244	[0.1103, 0.1380]	0.1228	[0.1083, 0.1368]
	pos./neg.	**0.1257**	[0.1115, 0.1395]	**0.1244**	[0.1099, 0.1384]

This table compares the doc2vec performances (F1 scores) of the approach with manual (👥) and automatic NLP annotations (🤖) once considering all entities as positive (all pos.) and once with positive and negative sentiment (pos./neg.). The best performing option is highlighted in bold face. The most important annotations for both the manual and the automatic approach are movies and keywords. Adding genres and actors only slightly improves the performance in both cases. The bootstrapped 95% confidence intervals ([CI]) show no significant differences between any of the configurations.

In Table 7 we list the results of our recommendation experiments including the ablation study. Note that we list only F1 scores@10 as the rankings are same for all other evaluation metrics. As expected, we achieve the best performance for both, the crowdworker and automatic approach, with all annotations included, closely followed by recommendations computed only from movie titles and keywords. The overlapping 95% confidence intervals (on a bootstrapped test dataset) show that there are no significant differences neither between automatic and manual annotations regardless of the annotations included nor within automatic or crowdworker recommendations regardless of annotation type or sentiment.

In Fig. 7 we depict the fine-tuned modifier weights for both manual and automatic annotations. Although there are only minor differences, we find that for the crowdsourced approach the algorithmic score computed via doc2vec exhibits higher importance than for the approach with automatically labeled data. As the optimized weight for the IMDb genre modifier is zero for both approaches, the similarity of genres on IMDb between movies does not improve the recommendation accuracy and is therefore totally neglected for generating the final recommendations.

Finally, we achieve an F1 score of 0.1063 for the baseline NLP approach. However, as the coverage of this approach in the test set is only ∼12.5% (i.e., it annotates movie titles in only 1 out of 8 submissions), this approach fails to produce movie recommendations in a huge majority of cases.

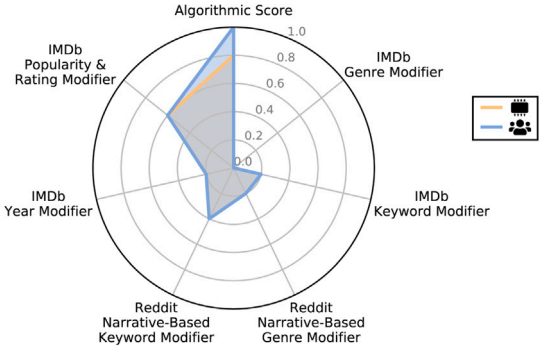

Fig. 7. Post Filter Weights. This figure shows the optimized post filter weights based on manually labeled data (👥) and based on automatically extracted annotations via NLP (🤖). We find only one minor difference between the post filter weights of the two approaches which is the algorithmic score. The algorithmic score of the manual approach has a marginally higher weight (i.e., 1.0) than the NLP approach (i.e., 0.8). All other post filters have identical ideal weights.

6. Discussion

Annotation Methods (RQ 1). Despite the recent successes of BERT-based approaches at various language modeling tasks (Min et al., 2021), in our dataset BiLSTM with ELMO was the best performing approach for detecting positive movies, keywords, and genres. We hypothesize that this is due to specific representation of tokens via contextual features. In particular, during preprocessing we preserved several punctuation characters that are frequently written in front of and/or after a movie title. For example, in our dataset, users have frequently enclosed movie titles in brackets, quotation marks, or have listed movie titles in separate lines of text. Hence, by learning representations of such characters, the model learned to better detect a movie title (among other features) by its specific position in the sentence, i.e., following or being in front of such punctuation and white-space characters.

Further, all of our models performed significantly better while annotating positive entities as opposed to negative ones. We believe that this is due to a substantial data imbalance (we had approximately 95% of positive and only 5% of negative annotations, cf. Table 1), indicating that in our dataset, users tend to talk of positive experiences rather than negative ones, resulting in a positivity bias (Huang et al., 2020; Pradel et al., 2012). One possibility to improve the performance of negative annotations could be to increase the number of negative samples with data augmentation techniques (Chen et al., 2021; Sutiono & Hahn-Powell, 2022). However, data augmentation cannot solve the problems of an indirect and implicit expression of sentiment by users. For instance, the following sentence is a typical example of how users express negative sentiment: "*[...] And while we're at it, nothing with Arnold Schwarzenegger in it, because I've seen his movies so many times, he used to be my hero.*". The model correctly identified Arnold Schwarzenegger as an actor but a positively mentioned actor rather than a negatively mentioned one. The model has a problem recognizing negative sentiment in such cases, as there are no explicit words that express a negative sentiment, and additionally, this sentence contains a positive sentiment towards the actor in the past, but a negative one in the present.

Regarding the detection of keywords and genres, we observed that they often get mixed up by classifiers as genres can be seen as a special kind of keywords coming from a predefined set. For this reason a simple string matching baseline for genres did almost as well as DL models.

Comparing crowdworker and automatic annotations of movie titles, crowdworkers did significantly better on recognizing abbreviations. For example, crowdworkers were able to recognize abbreviations, such as "*LOTR*" as "*Lord of the Rings*", or "*movies by QT*" as movies by Quentin Tarantino, whereas NLP approaches did not, as our automatic labeling process was not able to match abbreviation with full movie titles. While one possible solution to this problem would be to define a dictionary of frequent abbreviations, this dictionary would need constant updates as new movies, new actors, or new directors appear.

Also, while expanding annotated movie titles to provide more input for recommendation engines we observed some problems, in particular for the cases of sequels and prequels. For example, in a submission including "*Lord of The Rings*", the model correctly annotates the movie title. Expanding this movie title to include potential prequels or sequels fails in some cases as the complete titles, such as "*Lord of the Rings: The Fellowship of the Ring*", are not directly match-able. However, reducing the similarity threshold to match prequels and sequels also increases the likelihood for false positives and movie titles resulting in "*Lord of the Dogs*" being matched.

In summary, despite the data limitations (e.g., small sample and data imbalance), our NLP pipelines still perform reasonably well in the detection of non-traditional entities such as movie titles or movie keywords. We do believe that we have not used the full potential of the BERT architectures, and further adjustments could lead to even better results. Obtaining labeled data in the proper format and of good quality, still remains a challenge, which is why we believe that future approaches should be optimized to work in a fully end-to-end fashion. We see this investigation as an interesting and important avenue for our future research.

Recommendation Accuracy and Annotation Importance (RQ 2 &3). In our dataset we observed that automatic NLP annotations lead to comparable (statistically indistinguishable) recommendation accuracy as manual annotations. The high overlap in optimized post filter weights between the manual and the automatic approach corroborates the similarity of the extracted entities that serve as basis for the recommender engine. However, automatic annotations scale significantly better than the manually extracted ones (Appelt et al., 1993; Cimiano et al., 2004). Also, as we did not observe statistically significant differences in recommendation results when using different annotation types, we suggest to concentrate the efforts in improving the NLP extraction on a few most important annotations. In our case, these were movie titles and keywords. This result is in line with the findings of our previous work (Eberhard et al., 2020), where we analyzed the importance of entities—annotated by crowdworkers—in free-form text, and found that other annotations (e.g., actors) do not improve the recommendation accuracy. The importance of keywords in movie recommender systems was also already highlighted by Stanescu et al. (2013) and Szomszor et al. (2007). We speculate that in similar domains that deal with consumer goods, such as music, books, or video games, the results would replicate. However, this should be confirmed with further experiments and studies, which we see as another promising direction for future work.

Our recommendation results with the NLP baseline showed that off-the-shelf KE and NER models without training and optimizing are typically not sufficiently good for free-form text recommendation tasks, except in some special cases, such as movie genre annotations. Rather, training and fine-tuning of more sophisticated NLP pipelines is needed. Moreover, constructing ensemble approaches with the best performing models for each annotation type is of primary importance as our results suggest. This finding is also in line with the work from Tao et al. (2021) where an ensemble NLP approach led to the best annotation results. We expect that this translates in a similar way also to other domains.

Limitations & Future Work. In the process of annotating movie titles, we compare them to a pool of only ~12,000 English movie titles of popular movies on IMDb (cf. Eberhard et al., 2019). This includes only a small fraction of all movies ever made (i.e., more than 500,000 on IMDb) and hence constitutes a limitation of this work. Therefore, the selection likely introduces biased results due to the limited size of our potential recommendation candidates, which we need to be aware of. For a real world application, filtering of movies should be reduced to a minimum to keep the recommendation candidate pool as large as possible. However, while potentially not all 500,000 movies are relevant, we want to identify the best balance between recommendation novelty, diversity and accuracy. A possible solution for this problem could be the application of `doc2vec` using all possible movie titles.

Another limitation of our work is that we applied our whole NLP and recommender pipeline on a single dataset only. Moreover, our dataset is composed only from reddit data, in particular from a single subreddit. In addition, we also work with data from IMDb. Hence, besides the relatively small movie pool from IMDb, the used number of reddit submissions with 1480 is also rather small. However, we would like to point out that our dataset is of high quality as it was manually curated in a crowdsourcing experiment. Moreover, to substantiate our results we applied statistical testing in all of our experimental evaluations. Hence, we bootstrapped the dataset to obtain bootstrapped confidence intervals and statistically test and compare various approaches. Therefore, we are confident that our results may transfer to further movie datasets and potentially even to further domains after fine-tuning and updating the models. However, to substantiate our findings, investigations on further, larger movie datasets such as Movielens[24] as well as further domains, such as book, board game, or video game recommendations, are necessary.

Another promising research avenue for future work is the investigation of LLMs for elicitation of user preferences, understanding of complex user requests, as well as direct computation of recommendations. As LLMs can perform both tasks from our two-staged approach, i.e., extraction of user preferences and recommendation computations we plan a detailed analysis of their applicability for each of the stages individually as well as for both stages combined in a unified narrative recommendation framework.

[24] https://grouplens.org/datasets/movielens/

7. Conclusions

In this paper, we analyzed several algorithms specialized for KE and NER and two DL approaches for identification of custom defined entities (including keywords), and compared the obtained entities with the ones annotated by crowdworkers in our gold-standard.

We achieved the best results for positive movies, keywords and genres using a BiLSTM network with a wrapped ELMO layer for extracting contextual embeddings, a cased BERT model (large) for the recognition of negative keywords and genres, a large RoBERTa model for negative movie titles and actor names, and a base multilingual cased BERT for the identification of positive actor names. We then applied state-of-the-art recommender algorithms onto the automatically annotated data and compared the recommendation results to the results based on our gold-standard manually labeled annotations. With this fully automated annotation and recommendation process we found only small and not significant performance leaps to the manually labeled version.

The contributions of this paper are twofold. First, we evaluate the performance of KE and NER algorithms/models compared to manually labeled annotations. Second, we measure the impact of the differences between automatically extracted annotations and manually labeled annotations in a downstream recommendation experiment. We strongly believe that our contributions are an important building block for existing and future recommender systems that aim to improve their functionality with specific user generated requests and interactive components.

CRediT authorship contribution statement

Lukas Eberhard: Conceptualization, Methodology, Software, Validation, Writing – original draft, Visualization. **Kristina Popova:** Methodology, Software, Validation, Data curation, Writing – original draft, Visualization. **Simon Walk:** Conceptualization, Writing – review & editing. **Denis Helic:** Conceptualization, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request

Appendix A. Model token level evaluation

In this section we provide the average precision, recall and F1 scores from the token classification stage on all models that we trained. We also provide details on all of our model configurations.

Table A.8 shows the average scores of our fine-tuned models. Since fine-tuning them with different set of hyperparameters did not produce mentionable differences in the outcomes, we include only one configuration of each fine-tuned model. All of the fine-tuned models we present have the following configurations: *learning rate = 0.00001*, *batch size = 4*, *gradient threshold = 1.0* and we set number of *training epochs = 20*. For the feature-based models, we take slightly different approach. Since we train them completely from scratch, we experiment with: the type of embedding layer, number of features, dropout, neurons in the BiLSTM and recurrent dropout.

We group the models based on the embedding layer into: **base** models (have a regular embedding input layer that enumerates the input tokens), **feature** models (same embedding layer as base models, concatenated with an input of hand-picked features), **ELMO** (the basic embedding layer gets replaced with an ELMO layer that produces contextual embeddings), **ELMO features** (concatenation of the contextual embeddings with the hand-picked features.) We combine features according to their feature groups: 40 (lexical, syntactical, TF and IDF features), 50 (previous features plus sentiment features), 143 (previous features plus dense word representations). The rest of the configurations for each of the models is shown in Table A.9.

We give the detailed results of the best highlighted models from Tables A.8 and A.9 in Table A.10.

Appendix B. Model architectures

In this section we depict the model architectures used in our NLP pipelines. The figures show pre-processed and tokenized submissions as inputs to the models and how the models produce the labeled tokens (see Figs. B.8–B.10).

Appendix C. Detailed recommender experiment results

This section provides detailed results of the recommender experiment. Fig. C.11 shows the fine-tuned modifier weights for both manual C.11(a) and automatic C.11(b) annotations. We find a similar pattern across all approaches. The algorithmic recommendation score, the popularity of movies as well as the narrative-based keywords are the most important features. This result corroborates the findings of our previous work (Eberhard et al., 2020).

Table C.11 shows the results (F1 scores) of all applied approaches with manual and automatic NLP annotations once considering all entities as positive and once with positive and negative sentiment. The evaluated approaches are `doc2vec`, item-based collaborative filtering (CF), matrix factorization (MF), a TF–IDF approach, and a network-based approach. The most important annotations for both the manual and the automatic approach are movies and keywords. Adding genres and actors only slightly improves the performance in both cases. The bootstrapped 95% confidence intervals show no significant differences between the performances based on different entities. We find that the `doc2vec` approach leads to significantly better results in some of the cases than other approaches.

[25] https://huggingface.co/docs/transformers/index
[26] https://tfhub.dev/google/elmo/3

Table A.8
Fine-Tuned Models Average Results.

Model \ Metric	P	R	F1
BERT base (uncased)	0.40	0.40	0.37
BERT base (cased)	0.50	0.39	0.42
BERT large (uncased)	0.41	0.29	0.33
BERT large (cased)	**0.55**	**0.40**	**0.45**
BERT base multilingual (uncased)	0.44	0.40	0.39
BERT base multilingual (cased)	**0.56**	**0.38**	**0.43**
RoBERTa base	0.36	0.23	0.27
RoBERTa large	**0.58**	**0.37**	**0.42**
XLM RoBERTa base	0.13	0.07	0.06

The table shows nine fine-tuned models from the BERT family. Overall, as entity extraction is highly case-sensitive, the cased models perform better than the uncased ones. BERT large (cased), BERT base multilingual (cased) and RoBERTa large show superior overall results in comparison to the rest of the models.

Table A.9
Feature-Based Models Average Results.

Model \ Metric	P	R	F1
Base BiLSTM 1 (dropout 0.1, units 100, recurrent dropout 0.3)	0.42	0.33	0.34
Base BiLSTM 2 (dropout 0.5, units 100, recurrent dropout 0.3)	0.47	0.30	0.32
Base BiLSTM 3 (dropout 0.3, units 150, recurrent dropout 0.3)	0.42	0.30	0.32
Base BiLSTM 4 (dropout 0.5, units 200, recurrent dropout 0.7)	0.37	0.34	0.33
Base BiLSTM 5 (dropout 0.5, units 150, recurrent dropout 0.7)	0.41	0.32	0.33
Features (143) 1 (dropout 0.3, units 200, recurrent dropout 0.5)	0.39	0.29	0.31
Features (143) 2 (dropout 0.3, units 150, recurrent dropout 0.5)	0.40	0.33	0.35
Features (143) 3 (dropout 0.1, units 100, recurrent dropout 0.3)	0.43	0.32	0.35
Features (40) 1 (dropout 0.1, units 200, recurrent dropout 0.4)	0.41	0.35	0.36
Features (40) 2 (dropout 0.3, units 100, recurrent dropout 0.5)	0.39	0.33	0.36
Features (50) 1 (dropout 0,1, units 100, recurrent dropout 0.5)	0.46	0.32	0.34
Features (50) 2 (dropout 0.1, units 100, recurrent dropout 0.3)	0.47	0.33	0.34
Features (50) 3 (dropout 0.3, units 100, recurrent dropout 0.5)	0.42	0.31	0.32
ELMO 1 (dropout 0.2, units 512, recurrent dropout 0.35)	0.52	0.39	0.43
ELMO 2 (dropout 0.5, units 512, recurrent dropout 0.35)	**0.54**	**0.45**	**0.48**
ELMO 3 (dropout 0.35, units 150, recurrent dropout 0.35)	0.50	0.40	0.42
ELMO 4 (dropout 0.35, units 512, recurrent dropout 0.5)	0.51	0.43	0.46
ELMO features (40) 1 (dropout 0.3, units 512, recurrent dropout 0.5)	0.53	0.46	0.49
ELMO features (40) 2 (dropout 0.2, units 512, recurrent dropout 0.5)	0.51	0.43	0.45
ELMO features (40) 3 (dropout 0.2, units 512, recurrent dropout 0.3)	0.53	0.42	0.45
ELMO features (50) 1 (dropout 0.3, units 512, recurrent dropout 0.7)	0.51	0.42	0.45
ELMO features (50) 2 (dropout 0.1, units 512, recurrent dropout 0.7)	0.52	0.40	0.43
ELMO features (50) 3 (dropout 0.7, units 512, recurrent dropout 0.5)	0.52	0.43	0.45

The table shows the average precision, recall and F1 scores of the token classification stage for the feature-based models. For every sub-group of models we experimented with different dropout levels and units in the BiLSTM layer, as well as recurrent dropout values. The results show that, despite carefully chosen, the hand-picked features do not play a significant role in the results improvement, and the greatest jump in model performance is due to the ELMO contextual embbeddings. We tried other variations of hyperparameters, but they, however, do not show significant changes to the models in the table.

Table A.10
Best Models Results.

Model	ELMO			RoBERTa			BERT base multilingual cased			BERT large cased		
Tag \ Metric	P	R	F1	P	R	F1	P	R	F1	P	R	F1
B-movie-pos	0.89	0.89	0.89	0.77	0.66	0.71	0.85	0.66	0.74	0.84	0.69	0.76
I-movie-pos	0.90	0.87	0.88	0.80	0.79	0.79	0.92	0.72	0.80	0.93	0.70	0.80
B-genre-pos	0.73	0.80	0.76	0.79	0.27	0.40	0.77	0.74	0.75	0.72	0.77	0.74
I-genre-pos	0.95	0.80	0.86	0.88	0.80	0.83	0.90	0.86	0.88	0.88	0.32	0.47
B-actor-pos	0.58	0.54	0.56	0.50	0.10	0.17	0.50	0.27	0.35	0.61	0.37	0.46
I-actor-pos	0.57	0.50	0.53	0.69	0.35	0.46	0.65	0.42	0.51	0.73	0.42	0.54
B-keyword-pos	0.61	0.45	0.51	0.48	0.36	0.41	0.61	0.39	0.48	0.65	0.54	0.59
I-keyword-pos	0.48	0.32	0.38	0.45	0.33	0.38	0.51	0.23	0.32	0.50	0.32	0.39
B-movie-neg	0.00	0.00	0.00	0.38	0.15	0.21	0.00	0.00	0.00	0.50	0.05	0.09
I-movie-neg	0.00	0.00	0.00	0.67	0.14	0.23	0.00	0.00	0.00	0.00	0.00	0.00
B-genre-neg	0.50	0.06	0.10	0.50	0.06	0.10	0.80	0.44	0.57	0.67	0.56	0.61
B-actor-neg	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
I-actor-neg	0.00	0.00	0.00	1.00	0.25	0.40	0.00	0.00	0.00	0.00	0.00	0.00
B-keyword-neg	0.67	0.15	0.24	0.31	0.21	0.35	0.36	0.10	0.15	0.33	0.24	0.28
I-keyword-neg	0.75	0.14	0.23	0.27	0.13	0.18	1.00	0.04	0.08	0.40	0.09	0.14

The table shows the scores for every token type for the best four models: ELMO, RoBERTa, BERT base multilingual (cased), and BERT large (cased). The results for 'I-genre-neg' are not included as there was no token associated with the tag in the test set present.

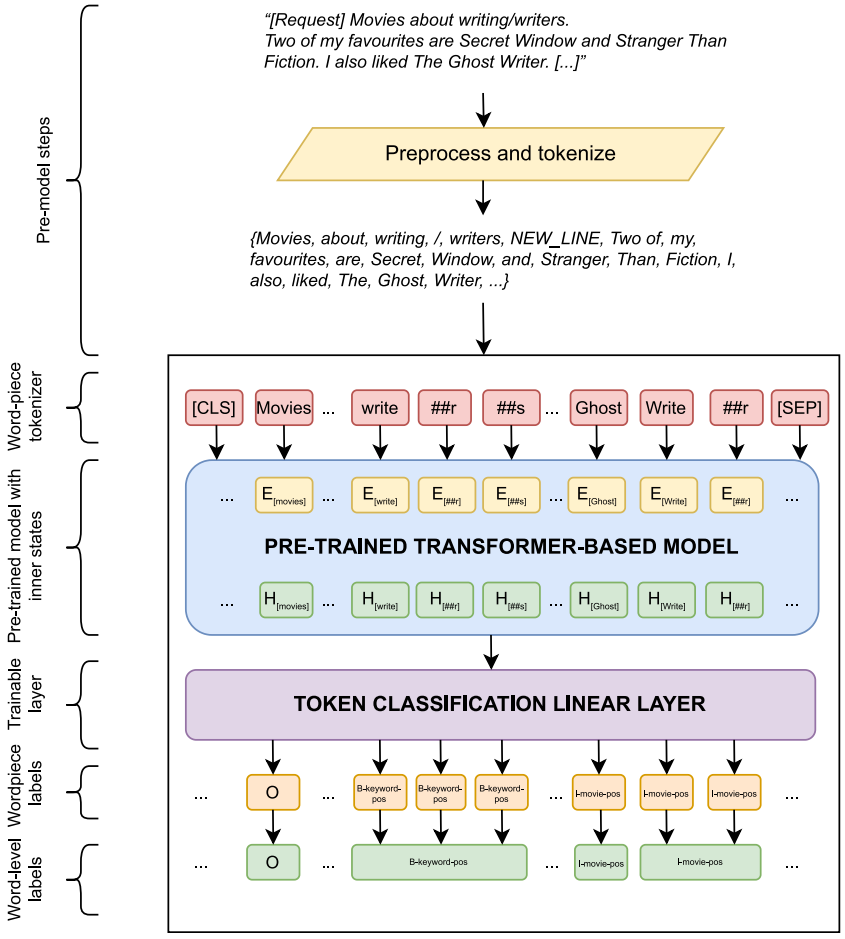

Fig. B.8. Architecture of Fine-Tuning Models. This figure shows the overall architecture used for all of our fine-tuning models. For all of the models listed in Table A.8 we used its corresponding pre-trained model from the Transformers package.[25] We did not alter anything within these models, but only fine-tuned them by altering the parameters for the linear token classification layer. All of the pre-trained models come with a word-piece tokenizer that transforms the tokens such that it splits them further to a base token and a continuation token (starts with '##'). The tokenizer also appends special tokens such as 'CLS' and 'SEP', which indicate the class and the separator to the next submission, respectively. All of the consecutive WordPiece tokens first inherit the label of the base token. These labels are then transformed to word-level labels in the subsequent phase of the classification.

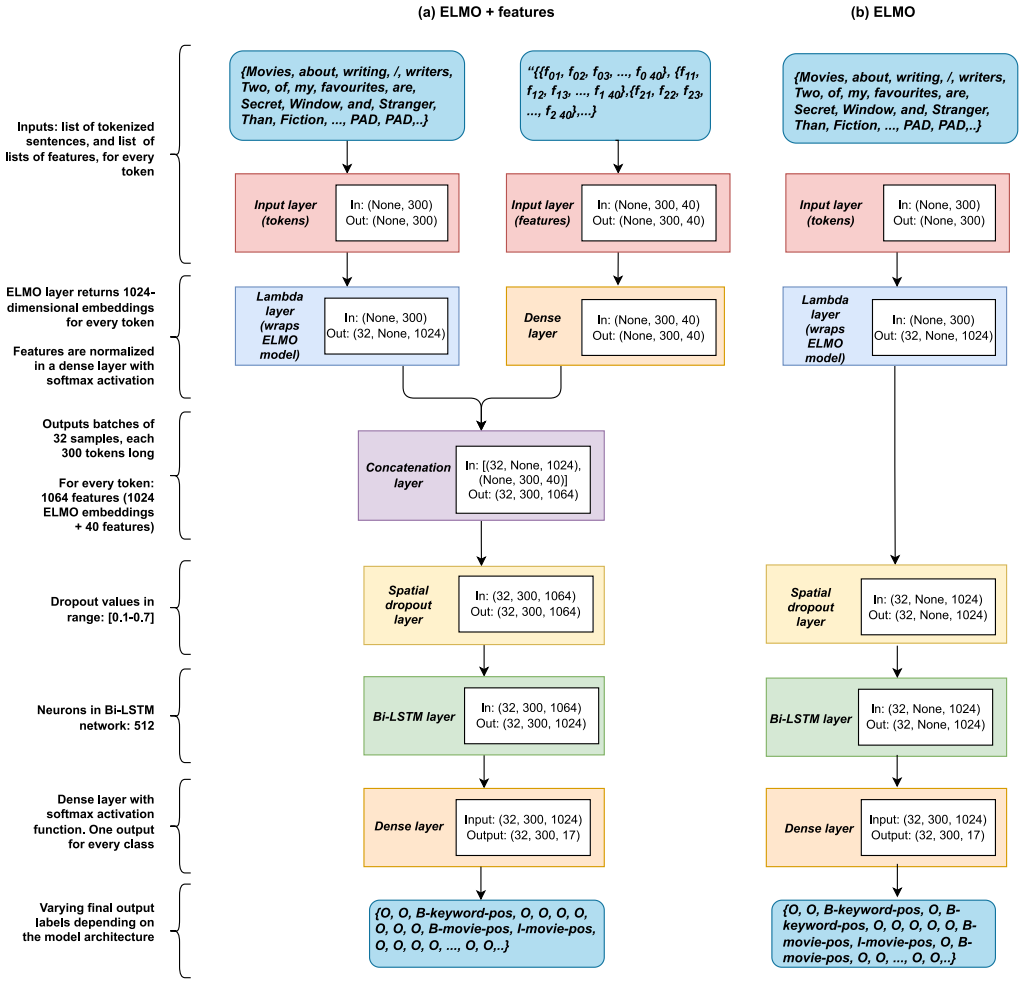

Fig. B.9. Architecture of Feature-Based Models With ELMO Embeddings. This figure shows the overall architecture used for our feature-based models with an ELMO embedding layer. Subfigure (a) is an example of a model with additional hand-picked features, passed in a separate input layer, and then concatenated with the ELMO embeddings. Models in subfigure (b) follow the same architecture but without hand-picked features and without concatenation layer. The embeddings are generated in a lambda layer, which wraps an entire pre-trained ELMO model. This is a separate model and we do not alter its architecture, we only set its output dimensions.[26]

Table C.11
Recommender Results.

Entities	Sentiment	[CI]					[CI]				
		doc2vec	CF	MF	TF–IDF	Network	doc2vec	CF	MF	TF–IDF	Network
Movies	all pos.	0.1174 [0.1033, 0.1309]	0.0961 [0.0846, 0.1071]	0.1088 [0.0963, 0.1208]	0.0930 [0.0813, 0.1042]	0.0562 [0.0475, 0.0644]	0.1148 [0.1008, 0.1283]	0.0901 [0.0781, 0.1014]	0.1028 [0.0905, 0.1145]	0.0883 [0.0768, 0.0992]	0.0530 [0.0437, 0.0617]
	pos./neg.	0.1185 [0.1044, 0.1320]	0.0975 [0.0859, 0.1085]	0.1095 [0.0970, 0.1214]	0.0926 [0.0808, 0.1038]	0.0568 [0.0481, 0.0650]	0.1148 [0.1008, 0.1283]	0.0901 [0.0782, 0.1014]	0.1026 [0.0903, 0.1144]	0.0878 [0.0763, 0.0988]	0.0530 [0.0437, 0.0617]
Movies | Genres	all pos.	0.1201 [0.1059, 0.1337]	0.0982 [0.0862, 0.1096]	0.1130 [0.1001, 0.1253]	0.0938 [0.0821, 0.1050]	0.0573 [0.0481, 0.0660]	0.1173 [0.1029, 0.1310]	0.0905 [0.0788, 0.1017]	0.1046 [0.0924, 0.1164]	0.0908 [0.0794, 0.1018]	0.0529 [0.0437, 0.0615]
	pos./neg.	0.1204 [0.1061, 0.1340]	0.0991 [0.0871, 0.1106]	0.1135 [0.1006, 0.1258]	0.0932 [0.0814, 0.1045]	0.0576 [0.0483, 0.0663]	0.1168 [0.1024, 0.1306]	0.0901 [0.0785, 0.1014]	0.1042 [0.0920, 0.1160]	0.0897 [0.0783, 0.1008]	0.0529 [0.0437, 0.0615]
Movies | Keywords	all pos.	0.1250 [0.1107, 0.1387]	0.1114 [0.0992, 0.1233]	0.1228 [0.1100, 0.1352]	0.1087 [0.0970, 0.1200]	0.0620 [0.0528, 0.0708]	0.1211 [0.1068, 0.1349]	0.1080 [0.0952, 0.1203]	0.1147 [0.1019, 0.1269]	0.1053 [0.0929, 0.1171]	0.0587 [0.0497, 0.0674]
	pos./neg.	0.1253 [0.1112, 0.1390]	0.1120 [0.0998, 0.1238]	0.1214 [0.1086, 0.1338]	0.1104 [0.0987, 0.1216]	0.0611 [0.0520, 0.0697]	0.1228 [0.1085, 0.1365]	0.1096 [0.0968, 0.1218]	0.1150 [0.1023, 0.1272]	0.1057 [0.0934, 0.1175]	0.0584 [0.0493, 0.0670]
Movies | Actors	all pos.	0.1152 [0.1016, 0.1284]	0.0939 [0.0833, 0.1043]	0.1068 [0.0953, 0.1179]	0.0911 [0.0800, 0.1017]	0.0544 [0.0461, 0.0623]	0.1145 [0.1004, 0.1280]	0.0900 [0.0781, 0.1014]	0.1023 [0.0900, 0.1141]	0.0879 [0.0764, 0.0989]	0.0523 [0.0430, 0.0610]
	pos./neg.	0.1183 [0.1044, 0.1316]	0.0968 [0.0858, 0.1075]	0.1090 [0.0970, 0.1206]	0.0915 [0.0801, 0.1023]	0.0557 [0.0472, 0.0638]	0.1145 [0.1004, 0.1280]	0.0900 [0.0781, 0.1014]	0.1022 [0.0898, 0.1139]	0.0874 [0.0759, 0.0984]	0.0523 [0.0430, 0.0610]
Movies | Genres | Keywords | Actors	all pos.	0.1244 [0.1103, 0.1380]	0.1124 [0.1007, 0.1239]	0.1232 [0.1112, 0.1351]	0.1066 [0.0950, 0.1179]	0.0614 [0.0523, 0.0702]	0.1228 [0.1083, 0.1368]	0.1068 [0.0943, 0.1188]	0.1171 [0.1046, 0.1292]	0.1059 [0.0939, 0.1175]	0.0578 [0.0489, 0.0662]
	pos./neg.	0.1257 [0.1115, 0.1395]	0.1151 [0.1028, 0.1271]	0.1233 [0.1104, 0.1357]	0.1094 [0.0975, 0.1210]	0.0617 [0.0523, 0.0707]	0.1244 [0.1099, 0.1384]	0.1079 [0.0954, 0.1199]	0.1166 [0.1041, 0.1287]	0.1055 [0.0935, 0.1172]	0.0571 [0.0483, 0.0656]

This table compares the performances (F1 scores) of all applied approaches with manual (👥) and automatic NLP annotations (🤖) once considering all entities as positive (all pos.) and once with positive and negative sentiment (pos./neg.). The evaluated approaches are doc2vec, item-based collaborative filtering (CF), matrix factorization (MF), a TF–IDF approach, and a network-based approach. The most important annotations for both the manual and the automatic approach are movies and keywords. Adding genres and actors only slightly improves the performance in both cases. The bootstrapped 95% confidence intervals ([CI]) show no significant differences between the performances based on different entities. We find that the doc2vec approach leads to significantly better results in some of the cases than other approaches.

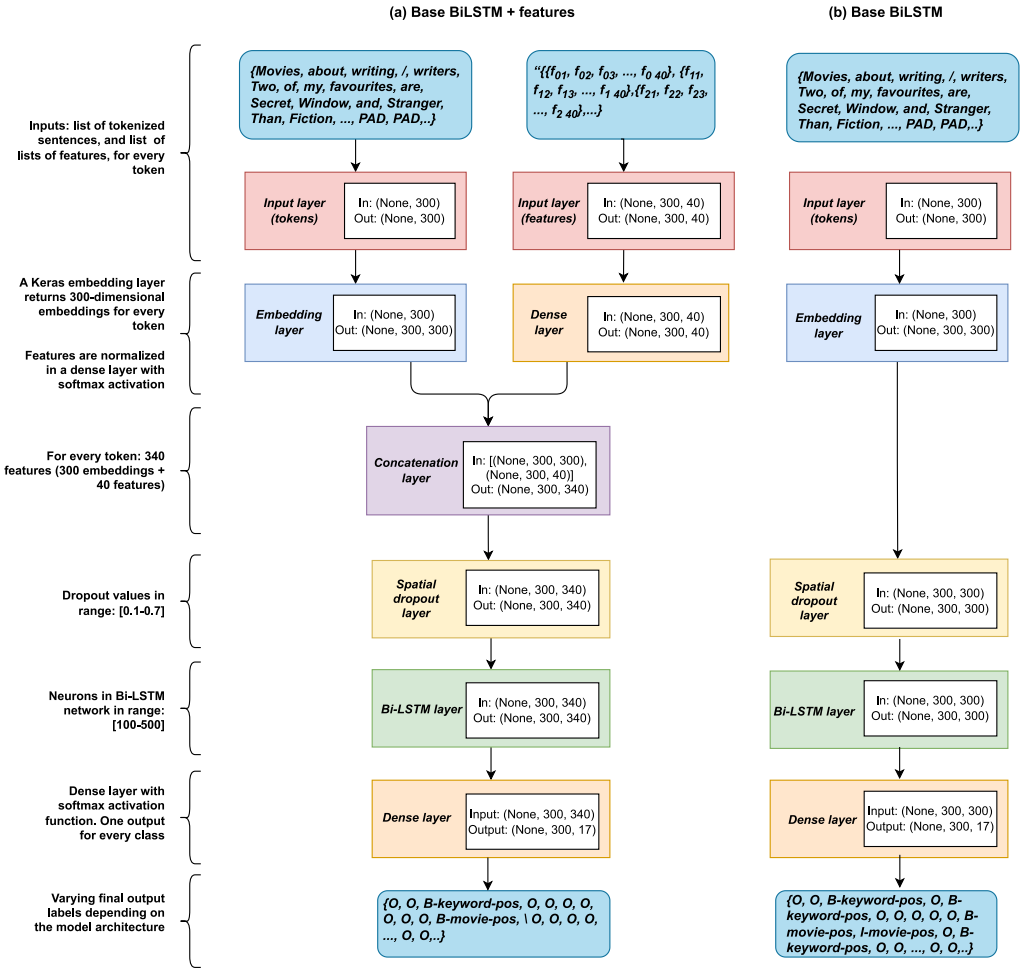

Fig. B.10. Architecture of Base Feature-Based Models. This figure shows the overall architecture used for our base feature-based models. They only differ from the ELMO models in the embeddings they use. These models use a regular Keras Embedding layer that turns tokens into dense vectors of fixed size. These vectors are not as highly context-sensitive as ELMO embeddings, resulting in inferior performance.

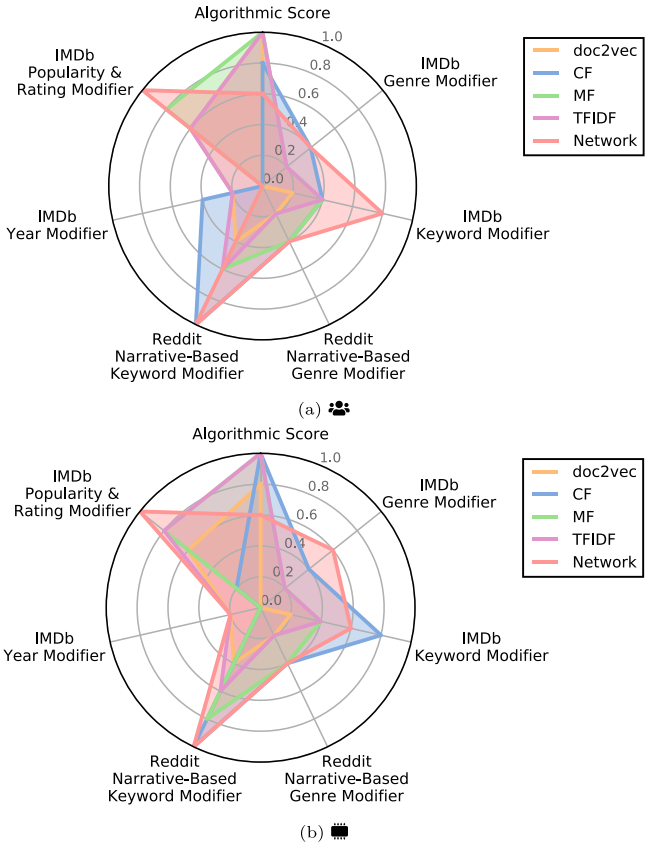

Fig. C.11. Post Filter Weights. These plots show the optimized post filter weights based on manually labeled data (👥) in Fig. C.11(a) and based on automatically extracted annotations via NLP (🤖) in Fig. C.11(b) for every evaluated approach. We find a similar pattern across all approaches. The algorithmic recommendation score, the popularity of movies as well as the narrative-based keywords are the most important features.

References

Adomavicius, G., Sankaranarayanan, R., Sen, S., & Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information systems (TOIS)*, *23*(1), 103–145. http://dx.doi.org/10.1145/1055709.1055714.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749. http://dx.doi.org/10.1109/TKDE.2005.99.

Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D., & Tyson, M. (1993). FASTUS: A finite-state processor for information extraction from real-world text. In *IJCAI, vol. 93* (pp. 1172–1178).

Ashwini, S., & Choi, J. D. (2014). Targetable named entity recognition in social media. *Computing Research Repository (CoRR)*, http://dx.doi.org/10.48550/arXiv.1408.0782.

Barkan, O., & Koenigstein, N. (2016). ITEM2vec: Neural item embedding for collaborative filtering. In *2016 IEEE 26th international workshop on machine learning for signal processing* (pp. 1–6). http://dx.doi.org/10.1109/MLSP.2016.7738886.

Baumgartner, J., Zannettou, S., Keegan, B., Squire, M., & Blackburn, J. (2020). The pushshift reddit dataset. In *Proceedings of the international AAAI conference on web and social media, vol. 14* (pp. 830–839). http://dx.doi.org/10.1609/icwsm.v14i1.7347.

Bhattacharjee, K., Ballesteros, M., Anubhai, R., Muresan, S., Ma, J., Ladhak, F., & Al-Onaizan, Y. (2020). To BERT or not to BERT: Comparing task-specific and task-agnostic semi-supervised approaches for sequence tagging. In *Proceedings of the 2020 conference on empirical methods in natural language processing* (pp. 7927–7934). Online: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.emnlp-main.636.

Bogers, T., & Koolen, M. (2017). Defining and supporting narrative-driven recommendation. In *Proceedings of the eleventh ACM conference on recommender systems* (pp. 238–242). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3109859.3109893.

Brown, P. F., Della Pietra, V. J., deSouza, P. V., Lai, J. C., & Mercer, R. L. (1992). Class-based *n*-gram models of natural language. *Computational Linguistics*, *18*(4), 467–480, URL: https://aclanthology.org/J92-4003.

Cai, H., Tu, Y., Zhou, X., Yu, J., & Xia, R. (2020). Aspect-category based sentiment analysis with hierarchical graph convolutional network. In *Proceedings of the 28th international conference on computational linguistics* (pp. 833–843). Barcelona, Spain (Online): International Committee on Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.coling-main.72, URL: https://aclanthology.org/2020.coling-main.72.

Cenikj, G., & Gievska, S. (2020). Boosting recommender systems with advanced embedding models. In *Companion proceedings of the web conference 2020* (pp. 385–389). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3366424.3383300.

Chen, S., Aguilar, G., Neves, L., & Solorio, T. (2021). Data augmentation for cross-domain named entity recognition. http://dx.doi.org/10.48550/arXiv.2109.01758, arXiv.

Chieu, H. L., & Ng, H. T. (2003). Named entity recognition with a maximum entropy approach. In *Proceedings of the seventh conference on natural language learning at HLT-NAACL 2003* (pp. 160–163). URL: https://aclanthology.org/W03-0423.

Chiu, J. P., & Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, *4*, 357–370. http://dx.doi.org/10.1162/tacl_a_00104.

Christakopoulou, K., Radlinski, F., & Hofmann, K. (2016). Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 815–824). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2939672.2939746.

Christakou, C., Vrettos, S., & Stafylopatis, A. (2007). A hybrid movie recommender system based on neural networks. *International Journal on Artificial Intelligence Tools*, *16*(05), 771–792. http://dx.doi.org/10.1142/S0218213007003540.

Cimiano, P., Handschuh, S., & Staab, S. (2004). Towards the self-annotating web. In *Proceedings of the 13th international conference on world wide web* (pp. 462–471). http://dx.doi.org/10.1142/S0218213007003540.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 8440–8451). Online: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/2020.acl-main.747.

Cui, Z., Ma, J., Zhou, C., Zhou, J., & Yang, H. (2022). M6-Rec: Generative pre-trained language models are open-ended recommender systems. http://dx.doi.org/10.48550/arXiv.2205.08084, arXiv:2205.08084.

Desrosiers, C., & Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook* (pp. 107–144). Boston, MA: Springer US, http://dx.doi.org/10.1007/978-0-387-85820-3_4.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *Computing Research Repository (CoRR)*, http://dx.doi.org/10.48550/arXiv.1810.04805.

Do, H. H., Prasad, P., Maag, A., & Alsadoon, A. (2019). Deep learning for aspect-based sentiment analysis: A comparative review. *Expert Systems with Applications*, *118*, 272–299. http://dx.doi.org/10.1016/j.eswa.2018.10.003.

Eberhard, L., Walk, S., & Helic, D. (2020). Tell me what you want: Embedding narratives for movie recommendations. In *Proceedings of the 31st ACM conference on hypertext and social media* (pp. 301–306). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3372923.3404818.

Eberhard, L., Walk, S., Posch, L., & Helic, D. (2019). Evaluating narrative-driven movie recommendations on reddit. In *Proceedings of the 24th international conference on intelligent user interfaces* (pp. 1–11). New York, NY, USA: Association for Computing Machinery, URL: https://doi.acm.org/10.1145/3301275.3302287.

Elsafty, A., Riedl, M., & Biemann, C. (2018). Document-based recommender system for job postings using dense representations. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, vol. 3 (industry papers)* (pp. 216–224). New Orleans - Louisiana: Association for Computational Linguistics, URL: https://www.aclweb.org/anthology/N18-3027.

Feng, S., Cong, G., An, B., & Chee, Y. M. (2017). POI2vec: Geographical latent representation for predicting future visitors. AAAI 2017, (pp. 102–108). http://dx.doi.org/10.1609/aaai.v31i1.10500, Cited by: 159,

Fessahaye, F., Perez, L., Zhan, T., Zhang, R., Fossier, C., Markarian, R., Chiu, C., Zhan, J., Gewali, L., & Oh, P. (2019). T-recsys: A novel music recommendation system using deep learning. In *2019 IEEE international conference on consumer electronics* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/ICCE.2019.8662028.

Fu, L., & Ma, X. (2021). An improved recommendation method based on content filtering and collaborative filtering. *Complexity*, *2021*, http://dx.doi.org/10.1155/2021/5589285.

Funk, S. (2006). Netflix update: Try this at home.

Ghosh, S., Mundhe, M., Hernandez, K., & Sen, S. (1999). Voting for movies: The anatomy of a recommender system. In *Proceedings of the third annual conference on autonomous agents* (pp. 434–435). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/301136.301303.

Glenski, M., & Weninger, T. (2017). Predicting user-interactions on reddit. In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017* (pp. 609–612). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3110025.3120993.

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2939672.2939754.

Gundogdu, A. S., Sanghvi, A., & Harrigian, K. (2018). Recognizing film entities in podcasts. *Computing Research Repository (CoRR)*, http://dx.doi.org/10.48550/arXiv.1809.08711.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems, vol. 30*. Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

Hariri, N., Mobasher, B., & Burke, R. (2013). Query-driven context aware recommendation. In *Proceedings of the 7th ACM conference on recommender systems* (pp. 9–16). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2507157.2507187.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hu, M., Peng, Y., Huang, Z., Li, D., & Lv, Y. (2019). Open-domain targeted sentiment analysis via span-based extraction and classification. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 537–546). Florence, Italy: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/P19-1051.

Huang, J., Oosterhuis, H., de Rijke, M., & van Hoof, H. (2020). Keeping dataset biases out of the simulation: A debiased simulator for reinforcement learning based recommender systems. In *Proceedings of the 14th ACM conference on recommender systems* (pp. 190–199). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3383313.3412252.

Janchevski, A., & Gievska, S. (2019). A study of different models for subreddit recommendation based on user-community interaction. In *ICT innovations 2019. big data processing and mining* (pp. 96–108). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-33110-8_9.

Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing* (2nd ed.). USA: Prentice-Hall, Inc..

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., ... Kasneci, G. (2023). Chatgpt for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, *103*, Article 102274. http://dx.doi.org/10.1016/j.lindif.2023.102274.

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 426–434). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/1401890.1401944.

Kouki, P., Fountalis, I., Vasiloglou, N., Cui, X., Liberty, E., & Al Jadda, K. (2020). From the lab to production: A case study of session-based recommendations in the home-improvement domain. In *Proceedings of the 14th ACM conference on recommender systems* (pp. 140–149). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3383313.3412235.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies* (pp. 260–270). San Diego, California: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/N16-1030.

Lamprecht, D., Geigl, F., Karas, T., Walk, S., Helic, D., & Strohmaier, M. (2015). Improving recommender system navigability through diversification: A case study of IMDb. In *i-KNOW '15, Proceedings of the 15th international conference on knowledge technologies and data-driven business*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2809563.2809603.

Langevin, R., Lordon, R. J., Avrahami, T., Cowan, B. R., Hirsch, T., & Hsieh, G. (2021). Heuristic evaluation of conversational agents. In *Proceedings of the 2021 CHI conference on human factors in computing systems*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3411764.3445312.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of machine learning research: vol. 32, Proceedings of the 31st international conference on machine learning* (no. 2), (pp. 1188–1196). Bejing, China: PMLR, URL: https://proceedings.mlr.press/v32/le14.html.

Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, *3*, 211–225. http://dx.doi.org/10.1162/tacl_a_00134.

Li, X., Bing, L., Zhang, W., & Lam, W. (2019). Exploiting BERT for end-to-end aspect-based sentiment analysis. In *Proceedings of the 5th workshop on noisy user-generated text* (pp. 34–41). Hong Kong, China: Association for Computational Linguistics, http://dx.doi.org/10.18653/v1/D19-5505.

Li, C., Gao, F., Bu, J., Xu, L., Chen, X., Gu, Y., Shao, Z., Zheng, Q., Zhang, N., Wang, Y., & Yu, Z. (2021). SentiPrompt: Sentiment knowledge enhanced prompt-tuning for aspect-based sentiment analysis. *Computing Research Repository (CoRR)*, http://dx.doi.org/10.48550/arXiv.2109.08306.

Li, H., Sanner, S., Luo, K., & Wu, G. (2020). A ranking optimization approach to latent linear critiquing for conversational recommender systems. In *Proceedings of the 14th ACM conference on recommender systems* (pp. 13–22). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3383313.3412240.

Liang, D., Altosaar, J., Charlin, L., & Blei, D. M. (2016). Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 59–66). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2959100.2959182.

Liu, S., Li, W., Wu, Y., Su, Q., & Sun, X. (2020). Jointly modeling aspect and sentiment with dynamic heterogeneous graph neural networks. arXiv:2004.06427.

Liu, J., Liu, C., Lv, R., Zhou, K., & Zhang, Y. (2023). Is ChatGPT a good recommender? A preliminary study. http://dx.doi.org/10.48550/arXiv.2304.10149, arXiv:2304.10149.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *Computing Research Repository (CoRR)*, http://dx.doi.org/10.48550/arXiv.1907.11692.

Ludewig, M., & Jannach, D. (2018). Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, *28*(4), 331–390. http://dx.doi.org/10.1007/s11257-018-9209-6.

Mak, H., Koprinska, I., & Poon, J. (2003). Intimate: a web-based movie recommender using text categorization. In *Proceedings IEEE/WIC international conference on web intelligence* (pp. 602–605). http://dx.doi.org/10.1109/WI.2003.1241277.

Mansouri, A., Affendey, L., & Mamat, A. (2008). Named entity recognition approaches. *The International Journal of Computer Science*, *8*.

Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing* (pp. 404–411). Barcelona, Spain: Association for Computational Linguistics, URL: https://aclanthology.org/W04-3252.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. http://dx.doi.org/10.48550/arXiv.1301.3781, arXiv preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems, vol. 26*. Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heintz, I., & Roth, D. (2021). Recent advances in natural language processing via large pre-trained language models: A survey. *Computing Research Repository (CoRR)*, http://dx.doi.org/10.48550/arXiv.2111.01243.

Mnih, A., & Hinton, G. E. (2008). A scalable hierarchical distributed language model. In *Advances in neural information processing systems, vol. 21*. Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2008/file/1e056d2b0ebd5c878c550da6ac5d3724-Paper.pdf.

Montazeralghaem, A., Allan, J., & Thomas, P. S. (2021). Large-scale interactive conversational recommendation system using actor-critic framework. In *Proceedings of the 15th ACM conference on recommender systems* (pp. 220–229). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3460231.3474271.

Musto, C., Martina, A. F. M., Iovine, A., de Gemmis, M., & Semeraro, G. (2022). Tell me what you like: Introducing natural language preference elicitation strategies in a virtual assistant for the movie domain. *Preprint submitted to Expert Systems with Applications*, http://dx.doi.org/10.2139/ssrn.4140047, Available at SSRN 4140047.

Musto, C., Semeraro, G., De Gemmis, M., & Lops, P. (2015). Word embedding techniques for content-based recommender systems: An empirical evaluation. *Recsys Posters*, *1441*.

Oku, K., Nakajima, S., Miyazaki, J., & Uemura, S. (2006). Context-aware SVM for context-dependent information recommendation. In *7th international conference on mobile data management* (p. 109). http://dx.doi.org/10.1109/MDM.2006.56.

Okura, S., Tagami, Y., Ono, S., & Tajima, A. (2017). Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1933–1942). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3097983.3098108.

Ozsoy, M. G. (2016). From word embeddings to item recommendation. *Computing Research Repository (CoRR)*, http://dx.doi.org/10.48550/arXiv.1601.01356.

Panchendrarajan, R., & Amaresan, A. (2019). Bidirectional LSTM-CRF for named entity recognition.

Penha, G., & Hauff, C. (2020). What does BERT know about books, movies and music? Probing BERT for conversational recommendation. In *Proceedings of the 14th ACM conference on recommender systems* (pp. 388–397). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3383313.3412249.

Pereira, J. A., Matuszyk, P., Krieter, S., Spiliopoulou, M., & Saake, G. (2018). Personalized recommender systems for product-line configuration processes. *Computer Languages, Systems & Structures*, *54*, 451–471. http://dx.doi.org/10.1016/j.cl.2018.01.003.

Perny, P., & Zucker, J.-D. (2001). Preference-based search and machine learning for collaborative filtering: the "film-conseil" movie recommender system. *Information, Interaction, Intelligence*, *1*(1), 9–48.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *Computing Research Repository (CoRR)*, http://dx.doi.org/10.48550/arXiv.1802.05365.

Polignano, M., Musto, C., de Gemmis, M., Lops, P., & Semeraro, G. (2021). Together is better: Hybrid recommendations combining graph embeddings and contextualized word representations. In *Proceedings of the 15th ACM conference on recommender systems* (pp. 187–198). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3460231.3474272.

Pradel, B., Usunier, N., & Gallinari, P. (2012). Ranking with non-random missing ratings: Influence of popularity and positivity on evaluation metrics. In *Proceedings of the sixth ACM conference on recommender systems* (pp. 147–154). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2365952.2365982.

Ramshaw, L. A., & Marcus, M. P. (1999). Text chunking using transformation-based learning. In *Natural language processing using very large corpora* (pp. 157–176). Dordrecht: Springer Netherlands, http://dx.doi.org/10.1007/978-94-017-2390-9_10.

Ratcliff, J. W., & Metzener, D. E. (1988). Pattern-matching-the gestalt approach. *Dr Dobbs Journal*, *13*(7), 46.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (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). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/192844.192905.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1–35). Boston, MA: Springer US, http://dx.doi.org/10.1007/978-0-387-85820-3_1.

Ritter, A., Clark, S., Mausam, & Etzioni, O. (2011). Named entity recognition in tweets: An experimental study. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 1524–1534). USA: Association for Computational Linguistics.

Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). Automatic keyword extraction from individual documents. In *Text mining* (pp. 1–20). John Wiley & Sons, Ltd, http://dx.doi.org/10.1002/9780470689646.ch1.

Sabir, A., Lafontaine, E., & Das, A. (2022). Hey alexa, who am I talking to?: Analyzing users' perception and awareness regarding third-party Alexa skills. In *Proceedings of the 2022 CHI conference on human factors in computing systems*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3491102.3517510.

Salton, G., & McGill, M. J. (1983). Information retrieval: an introduction. In *Introduction to modern information retrieval* (pp. 1–23).

Schedl, M., Zamani, H., Chen, C.-W., Deldjoo, Y., & Elahi, M. (2018). Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, *7*(2), 95–116. http://dx.doi.org/10.1007/s13735-018-0154-2.

Setlur, V., & Tory, M. (2022). How do you converse with an analytical chatbot? Revisiting gricean maxims for designing analytical conversational behavior. In *Proceedings of the 2022 CHI conference on human factors in computing systems*. New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3491102.3501972.

Smith, B., & Linden, G. (2017). Two decades of recommender systems at Amazon.com. *IEEE Internet Computing*, *21*(3), 12–18. http://dx.doi.org/10.1109/MIC.2017.72.

de Souza Pereira Moreira, G., Rabhi, S., Lee, J. M., Ak, R., & Oldridge, E. (2021). Transformers4Rec: Bridging the gap between NLP and sequential / session-based recommendation. In *Proceedings of the 15th ACM conference on recommender systems* (pp. 143–153). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3460231.3474255.

Stanescu, A., Nagar, S., & Caragea, D. (2013). A hybrid recommender system: User profiling from keywords and ratings. In *2013 IEEE/WIC/ACM international joint conferences on web intelligence (WI) and intelligent agent technologies, vol. 1* (pp. 73–80). http://dx.doi.org/10.1109/WI-IAT.2013.11.

Stiebellehner, S., Wang, J., & Yuan, S. (2018). Learning continuous user representations through hybrid filtering with doc2vec. *Computing Research Repository (CoRR)*, http://dx.doi.org/10.48550/arXiv.1801.00215.

Sutiono, A. P., & Hahn-Powell, G. (2022). Syntax-driven data augmentation for named entity recognition. *Computing Research Repository (CoRR)*, http://dx.doi.org/10.48550/arXiv.2208.06957.

Szomszor, M., Cattuto, C., Alani, H., O'Hara, K., Baldassarri, A., Loreto, V., & Servedio, V. D. (2007). Folksonomies, the semantic web, and movie recommendation. In *4th European semantic web conference, bridging the gap between semantic web and web 2.0 (06/06/07)*. URL: https://eprints.soton.ac.uk/264007/.

Tao, J., Brayton, K. A., & Broschat, S. L. (2021). Automated confirmation of protein annotation using NLP and the UniProtKB database. *Applied Sciences*, *11*(1), http://dx.doi.org/10.3390/app11010024.

Terveen, L., & Hill, W. (2001). Beyond recommender systems: Helping people help each other. *HCI in the New Millennium*, *1*(2001), 487–509.

Truşcă, M. M., Wassenberg, D., Frasincar, F., & Dekker, R. (2020). A hybrid approach for aspect-based sentiment analysis using deep contextual word embeddings and hierarchical attention. In *Web engineering* (pp. 365–380). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-50578-3_25.

Vazan, M., & Razmara, J. (2021). Jointly modeling aspect and polarity for aspect-based sentiment analysis in Persian reviews. arXiv:2109.07680.

Villegas, N. M., Sánchez, C., Díaz-Cely, J., & Tamura, G. (2018). Characterizing context-aware recommender systems: A systematic literature review. *Knowledge-Based Systems*, *140*, 173–200. http://dx.doi.org/10.1016/j.knosys.2017.11.003.

Wang, S., Cao, L., Wang, Y., Sheng, Q. Z., Orgun, M. A., & Lian, D. (2021). A survey on session-based recommender systems. *ACM Computing Surveys*, *54*(7), http://dx.doi.org/10.1145/3465401.

Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., & Wang, Z. (2019). Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 968–977). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3292500.3330836.

Wölbitsch, M., Walk, S., Goller, M., & Helic, D. (2019). Beggars can't be choosers: Augmenting sparse data for embedding-based product recommendations in retail stores. In *Proceedings of the 27th ACM conference on user modeling, adaptation and personalization* (pp. 104–112). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3320435.3320454.

Wu, Y., Macdonald, C., & Ounis, I. (2021). Partially observable reinforcement learning for dialog-based interactive recommendation. In *Proceedings of the 15th ACM conference on recommender systems* (pp. 241–251). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3460231.3474256.

Zeng, Z., Ma, J., Chen, M., & Li, X. (2019). Joint learning for aspect category detection and sentiment analysis in Chinese reviews. In Q. Zhang, X. Liao, & Z. Ren (Eds.), *Information retrieval* (pp. 108–120). Cham: Springer International Publishing.

Zhang, Y., Ding, H., Shui, Z., Ma, Y., Zou, J., Deoras, A., & Wang, H. (2021). Language models as recommender systems: Evaluations and limitations. In *NeurIPS 2021 workshop on I (still) can't believe it's not better*. URL: https://www.amazon.science/publications/language-models-as-recommender-systems-evaluations-and-limitations.

Zitouni, I. (2014). *Natural language processing of semitic languages*. Springer, http://dx.doi.org/10.1007/978-3-642-45358-8.