Explainable Recommendations

Rose Catherine Kanjirathinkal

November 2017

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Prof. William W. Cohen, Chair, Carnegie Mellon University
Prof. Maxine Eskenazi, Carnegie Mellon University
Prof. Ruslan Salakhutdinov, Carnegie Mellon University
Prof. Jure Leskovec, Stanford University

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Abstract

We believe that Personalized Recommender Systems should not only produce good recommendations that suit the taste of each user but also provide an explanation that shows why each recommendation would be interesting or useful to the user, to be more effective. Explanations may serve many different purposes. They can show how the system works (transparency) or help users make an informed choice (effectiveness). They may be evaluated on whether they convince the user to make a purchase (persuasiveness) or whether they help the user make a decision quickly (efficiency). In general, providing an explanation has been shown to build user's trust in the recommender system [51].

Most often, the type of explanation that can be generated is constrained by the type of the model. In this thesis, we focus on generating recommendations and explanations using knowledge graphs as well as neural networks.

Knowledge graphs (KG) show how the content associated with users and items are interlinked to each other. Using KGs have been shown to improve recommender accuracies in the past. In the first part of this thesis, we show how recommendation accuracy can be improved using a logic programming approach on KGs. Additionally, we propose how explanations could be produced in such a setting by jointly ranking KG entities and items.

KGs however operate in the domain of discrete entities, and are therefore limited in their ability to deal with natural language content. Free form text such as reviews are a good source of information about both the user as well as the item. In the second part of this thesis, we shift our focus to neural models that are more amenable to natural language inputs, and we show how a teacher-student like architecture could be used to transform latent representations of user and item into that of their joint review to improve recommendation performance. We also show how such a framework could be used to select / predict a candidate review that would be most similar to the joint review. Such a review could possibly serve as an explanation of why the user would potentially like the item.

Different users are interested in different aspects of the same item. Therefore, most times, it is impossible to find a single review that would reflect all the interests of a user. A succinct explanation shown to a user for an item is ideally a personalized summary of all relevant reviews for that item. In the final part of this thesis, we propose a neural model that can generate a personalized abstractive summary as explanation and describe how such a model could be evaluated.

Contents

Chapter 1

Introduction

Personalized Recommendation Systems are an important aspect of e-commerce and are becoming increasingly prevalent in all aspects of our daily lives. They have become ubiquitous in a number of domains. Personalized services enable users to quickly find anything, be it a shopping item, movie, news or restaurants, that best suits their tastes, from the countless choices.

Personalized Recommendation Systems have garnered much attention over the past two decades and continue to be an important topic of research. Although predicting a user's rating for an item has been one of the main research goals, past research on recommender systems have also focused on a variety of other aspects of the system. For example, recommendations for a user in a particular context like geo-location, time or day, persons accompanying them, and mood, is very different those made outside of that context. Similarly, in certain applications, recommendation for a session is a research sub-area in itself because a session usually corresponds to a particular user intent. Interactive recommender systems that modify their recommendations in an online fashion as they consume user's feedback may use considerably different algorithms from mainstream systems. In addition to these, the domain of the application also calls for specialized algorithms. For example, video and music recommendations make use of features extracted from the data which are unavailable in other domains to improve their performance.

Although producing good recommendations is the primary goal, it is also desirable that such systems provide an explanation accompanying the recommendation. Explanations may serve one or more purposes [115]. Below are some of the important ones:

1. Transparency: Explanations that describe how a recommendation was chosen makes the system transparent to the user. Although many users may not care about the internal workings of a recommender system, transparency is a desirable property when the system shows non obvious recommendations.
2. Trust: Explanations that aim to increase users' confidence in the system fall into this category. An example is when the system reveals that it is not very confident that the user would like the recommendation, showing that the system is open and honest.
3. Persuasiveness: This is different from the other categories because the goal is to persuade the user to act on the recommendation for the benefit of the system. For example, the recommendations provided by an online shopping website may be optimized to increase their revenues and need not necessarily provide the best choices for the user.

4. Effectiveness: The goal of such explanations is to enable users to make good decisions by helping them understand why they would like or dislike a particular recommendation.
5. Efficiency: Such explanations help users make a decision quicker. Many of such explanations are usually structured as a comparison between competing items.
6. Satisfaction: Studies have shown that explanations could also increase the perceived satisfaction of the users with the recommender system.

If employed wisely, an explanation could contribute substantially to the acceptance and success of a recommender system [38, 115]. Therefore, there is a renewed interest in research concerning the generation of good explanations.

The focus of this thesis is on generating explanations together with high quality recommendations that are personalized to each user, and which will enable them to make an informed decision about the item. The approaches for explanation discussed in this proposal are most in line with 'Effectiveness' in the above list of purposes of providing explanations. Therefore, our approaches strive to also explain why a user may 'dislike' an item. i.e. they are not limited to finding only positives. Unless stated otherwise, both the recommendations as well as the explanations are personalized.

Our underlying hypothesis is that to be able to generate good recommendations and to effectively explain them, we need to model users' personal interests and show how these interests are reflected in the recommended product or item. Most often, the type of explanation that can be generated is constrained by the type of the underlying recommendation model. In this thesis, we focus on generating recommendations and explanations using knowledge graphs as well as neural networks.

Knowledge graphs (KG) show how the content associated with users and items are interlinked to each other. Using KGs have been shown to improve recommender accuracies in the past. In the first approach discussed in this proposal, we model the interests as entities that are interlinked via a knowledge graph. Items to be recommended are ranked using a random walk based approach in a logic programming framework (Chapter 2). The system generates explanations by jointly ranking the entities with the items (Chapter 3). This work was published in the 10^{th} and 11^{th} ACM Conferences on Recommender Systems (RecSys '16 & '17).

KG based methods have many limitations. For example, links between different entities are not always known. This is usually the case when deploying the system in a new domain. i.e. a KG is not always available. Also, since KGs are constructed using discrete entities, they are limited in their ability in dealing with natural language content such as user reviews. In many applications, user reviews are available. These free form text describe the user's experience with the item and are a good source of information about both the user as well as the item. However, KG based approaches make use only of the entities, and the context and sentiment in the surrounding text is overlooked.

In the second approach discussed in this proposal, we use neural models to leverage reviews written by users. The set of reviews written by a user represents his or her interests, as a latent vector space. Similarly, reviews written for an item are used to learn its overall representation. Items to be recommended are scored using a neural regression model (Chapter 4). This model generates a latent representation not of the entities, but of the user's predicted review. Therefore, the explanation provided is via the most similar review written for the item (Chapter 5) where the similarity is judged as the proximity in the latent space. This work was published in the 11^{th} ACM Conference on Recommender Systems (RecSys '17).

The main difficulty in evaluating explanations is the absence of gold standard data. While there

exists many datasets containing the user feedback on the actual recommendation itself, the same is not true when it comes to the explanations. In the last part of this proposal, we hypothesize that the ideal explanation given to a user for a particular item is their own review that they would have written after experiencing the item. Proposed work discussed in this proposal are approaches for generating the text corresponding to the latent representation obtained by the neural model of Chapters 4 and 5. Ideas toward this end are proposed in Chapter 6. In that chapter, we also propose various quantitative and qualitative measurements to judge the effectiveness of the generated text along different dimensions.

This thesis proposal is organized as follows: Chapters 2 and 3 detail techniques to improve recommendation accuracy and generate explanations using Knowledge Graphs. This is followed by Chapters 4 and 5 that discuss techniques for the same goals, but using neural networks. In Chapter 6, we propose the remaining work and a timeline for its completion. Prior work relevant to the research reported in this proposal is presented in the corresponding chapters.

Chapter 2

Completed Work: Rating Prediction using Knowledge Graphs

In this chapter, we show how Knowledge Graphs can be leveraged to improve personalized recommendations, using a general-purpose probabilistic logic system called ProPPR[126]. We formulate the problem as a probabilistic inference and learning task, and present three approaches for making recommendations. Our formulations build on a path-ranking approach called `Heterec_p` proposed in [142]. We show that a number of engineering choices, such as the choice of specific metapaths and length of metapaths, can be eliminated in our formalism, and that the formalism allows one to easily explore variants of the metapath approach. This work was published in the 10^{th} ACM Conference on Recommender Systems (RecSys '16) [19].

2.1 Background and Concepts: Knowledge Graphs

In this thesis proposal, we use the term *entity* as a generic term to denote a word or a phrase that can be mapped onto a knowledge base or an ontology. Since the knowledge bases used in this proposal are based on structured data, the mapping is straightforward. However, when using a generic knowledge base like Wikipedia[1], Yago [109] or NELL [85], one might require a wikifier or an entity linker [68]. Entities are typically generated from the content associated with the users and items. For users, these are typically their demographics. For items like movies, these may include the actors, genre, directors, country of release, etc. and for items like restaurants, these may include the location, cuisine, formal vs. casual, etc.

A Knowledge Graph (KG) is a graph constructed by representing each item, entity and user as nodes, and linking those nodes that interact with each other via edges. A related terminology used in literature is the Heterogenous Information Network (HIN), which is essentially a KG but with typed entities and links, and where there is more than one type of entity (heterogenous). Otherwise, the network becomes homogenous. A HIN is in contrast to prior works that use graphs or networks of only one type of nodes, like say, a friend network. A KG, as referred to in this proposal, is therefore a relaxed version of a HIN where the types of entities and links may or may not be known. We assume that the nodes are typically heterogenous even if their type information is missing. If the types are

[1] https://en.wikipedia.org

unknown, then only some methods are applicable. However, if the knowledge graph is indeed an HIN, then all three methods apply.

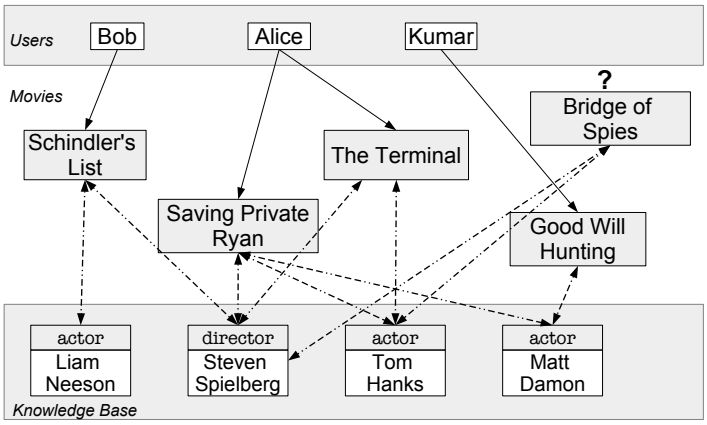

Figure 2.1: Example of Movie Recommendation with a Knowledge Graph

A typical movie recommendation example is depicted in Figure 2.1 where users watch and/or rate movies, and content about the movies are available in a database. For example, consider tracking three users `Bob`, `Alice` and `Kumar`. From usage records, we know that `Alice` has watched `Saving Private Ryan` and `The Terminal`, both of which have `Steven Spielberg` as the `Director` and `Tom Hanks` as an `Actor`, as specified by the knowledge base. The knowledge base may also provide additional content like plot keywords, language and country of release, awards won etc. Similarly, we also know the movies that were watched in the past by `Bob` and `Kumar`. In addition to watching, we could also include user's actions such as "reviewing" or "liking", if available. Given the past viewing history of users, we may want to know the likelihood of them watching a new movie, say `Bridge of Spies`. This scenario is graphically represented in Figure 2.1. Although in this particular case, the movie-entity graph is bipartite, it is also common to have links between movies themselves like say, `Finding Nemo` and `Finding Dory` where the latter is a sequel to the former, or between entities themselves like for example, `Tom Hanks` and `Best Actor Academy Award`.

2.2 Related Work: Knowledge Graphs for Recommendation

Recommendation systems have been popular for a long time now and are a well researched topic. However, there has not been much effort directed at using external KGs for improving recommendations.

A recent method [142], `HeteRec_p`, proposed the use of KGs for improving recommender performance. This method was the state-of-the-art at the time the work in this chapter was performed. We detail it in Section 2.2.1, since our approaches were compared against it. Another link-based method [141] proposed by the same authors precedes [142], and learns a global model of recommendation based on the KG, but does not attempt to personalize the recommendations. A similar method was proposed in [92], which used paths to find the top-N recommendations in a learning-to-rank framework. Since their method uses only the types of the nodes, they do not track the actual

entities that appear in the path. A few methods such as [85, 87] rank items using Personalized PageRank. In these methods, the entities present in the text of an item (e.g. a news article) are first mapped to entities in a knowledge graph.

A recent work [124] studies the problem of medicine recommendation using a Medical Knowledge Graph. This heterogeneous graph connects medicines, diseases and patients, and the edges encode relationships such as drug-drug interactions. Their method first embeds the entities and relationships of the graph into a low dimensional space using TransR [69] and LINE [112] . The recommendations are scored using their similarity in the embedded space. In [93], the authors enhance an existing graph embedding method called `node2vec` , to account for the different kinds of properties/relations. [132] is a recent work that uses Reinforcement Learning over KGs to reason over relations. Their method trains a policy-based agent that learns to choose a relation at every step to extend its current path on the graph. They apply their model to link and fact prediction tasks. Another method for the same task was proposed in [131] , where their model learned a manifold-based embedding of the graph.

A recent effort at using multiple sources of information is the HyPER system [57], where the authors show how to recommend using a Probabilistic Soft Logic framework [8]. They formulate rules to simulate collaborative filtering (CF) style recommendation. For instance, in the case of user-based CF, the rule is of the form, $SimilarUsers(u_1, u_2) \wedge Rating(u_1, i) \Rightarrow Rating(u_2, i)$, where $SimilarUsers$ indicate if the users u_1 and u_2 are similar using a k-nearest neighbor algorithm, computed offline using different similarity measures like Cosine, Pearson etc. If the friend-network of users is available, then they leverage it using the rule $Friends(u_1, u_2) \wedge Rating(u_1, i) \Rightarrow Rating(u_2, i)$. If other rating prediction algorithms like Matrix Factorization (MF) are available, then they induce an ensemble recommender using the rules $Rating_{MF}(u, i) \Rightarrow Rating(u, i)$ and $\neg Rating_{MF}(u, i) \Rightarrow \neg Rating(u, i)$. Eventually, during the training phase, they learn a weight per rule using the PSL framework, which is later used for predicting the ratings in the test set. Similar to HyPER is the approach proposed in [49] that uses Markov Logic Networks, and that proposed in [30] that uses Bayesian networks to create a hybrid recommender. Like these methods, we also base our methods on a general-purpose probabilistic reasoning system. However, we differ from these methods in our focus on using external knowledge in recommendations.

Prior research has previously proposed using various kinds of special purpose or domain–specific knowledge-graphs. In [51], the authors proposed to use a trust-network connecting the users especially for making recommendations to the cold-start users. The latter is matched up against the network to locate their most trusted neighbors, whose ratings are then used to generate the predictions. Another popularly used network is the social network of the users. Prior work like [41, 55, 75] among various other similar approaches use the social connection information of the users to find similar users or "friends" in this case, and use their ratings to generate recommendations for the former. In [13] , the authors use a graph representation of song playlists and impose a regularization constraint on NMF such that their similarity in the low dimensional space obtained by NMF reflects their distance in the graph.

2.2.1 HeteRec_p

`HeteRec_p` [142] aims to find user's affinity to items that they have not rated using *metapaths*. Metapaths describe paths in a graph through which two items may be connected. In the example of Figure

2.1, an example metapath would be `User` $\rightarrow$ `Movie` $\rightarrow$ `Actor` $\rightarrow$ `Movie`. Given a graph/network schema $G_T = (A, R)$ of a graph G where A is the set of node types and R is the set of relations between the node types A, then, metapaths are described in the form of $P = A_0 \xrightarrow{R_1} A_1 \xrightarrow{R_2} A_2 \ldots \xrightarrow{R_k} A_k$ and represent a path in G_T, which can be interpreted as a new composite relation $R_1 R_2 \ldots R_k$ between node-type A_0 and A_k, where $A_i \in A$ and $R_i \in R$ for $i = 0, \ldots k$, $A_0 = dom(R_1) = dom(P)$, $A_k = range(R_k) = range(P)$ and $A_i = range(R_i) = dom(R_{i+1})$ for $i = 1, \ldots, k-1$. For the specific purpose of recommending on user-item graphs, `HeteRec_p` uses metapaths of the form $user \rightarrow item \rightarrow * \rightarrow item$. Given a metapath P, they use a variant of PathSim [110] to measure the similarity between user i and item j along paths of the type P, a method the authors refer to as *User Preference Diffusion*. For each P, they use the user preference diffusion to construct the diffused user-item matrix $\tilde{R}_P$. Let $\tilde{R}^{(1)}, \tilde{R}^{(2)}, \ldots \tilde{R}^{(L)}$ be the diffused matrices corresponding to L different metapaths. Each such $\tilde{R}^{(q)}$ is then approximated as $\hat{U}^{(q)} \cdot \hat{V}^{(q)}$ using a low-rank matrix approximation technique. Then, the global recommendation model is expressed as: $r(u_i, v_j) = \sum_{q \in L} \theta_q \, \hat{U_i}^{(q)} \cdot \hat{V_j}^{(q)}$ where, θ_q is a learned weight for the path q. To personalize recommendations, they first cluster the users according to their interests. Then, the recommendation function is defined as: $r^*(u_i, v_j) = \sum_{k \in C} sim(C_k, u_i) \sum_{q \in L} \theta_q^{(k)} \, \hat{U_i}^{(q)} \cdot \hat{V_j}^{(q)}$ where, C represents the user clusters and $sim(C_k, u_i)$ gives a similarity score between the k^{th} cluster center and user i. Note that the θ_q is now learned for each of the clusters. This formulation of the rating prediction function is similar to [29]. Although `HeteRec_p` performed well on the recommendation tasks, the algorithm needs several hyper-parameters that need to be determined or tuned, like choosing the specific L metapaths from a potentially infinite number of metapaths, and the number of clusters. It also requires a rich KB with types for entities and links.

2.3 Method

2.3.1 Recommendation as Personalized PageRank

Consider the example in Figure 2.1. Similar to the Topic Sensitive PageRank proposed in [43] and the weighted association graph walks proposed in [16], imagine a random walker starting at the node `Alice` in the graph of Figure 2.1 (ignore the direction of the edges). At every step, the walker either moves to one of the neighbors of the current node with probability $1 - \alpha$ or jumps back to the start node (`Alice`) with probability α (the reset parameter). If repeated for a sufficient number of times, this process will eventually give an approximation of the steady-state probability of the walker being in each of the nodes. However, since we need only the ranking of the movies and not the other entities like actors and directors, we consider only those nodes corresponding to the movie nodes being tested, and sort them according to their probability to produce a ranked list of movie recommendations.

In the above method, there is no control over the walk. The final outcome of the walk is determined by the link structure and the start node alone. However, recent research has proposed methods to learn how to walk. Backstrom et. al in [9] showed how a random walker could be trained to walk on a graph. This is done by learning a weight vector w, which given a feature vector ϕ_{uv} for an edge in the graph $u \rightarrow v$, computes the edge strength as $f(w, \phi_{uv})$, a function of the weight

and the feature vectors, that is used in the walk. During the training phase, learning w is posed as an optimization problem with the constraint that the PageRank computed for the positive example nodes is greater than that of the negative examples. In our case, the positive examples would be those movies that the user watched, and negative examples would be those that the user did not watch or give an explicit negative feedback.

2.3.2 Learning to Recommend using ProPPR

ProPPR [126], which stands for **Pro**gramming with **P**ersonalized **P**age**R**ank, is a first-order probabilistic logic system which accepts a *program* similar in structure and semantics to a logic program [73] and a set of *facts*, and outputs a ranked list of entities that *answers* the program with respect to the facts. ProPPR scores possible answers to a query based on a Personalized PageRank process in the proof graph (explained below) for the query. Below, we show how it can be used for the task of learning to recommend.

For the recommendation task, the first step is to find a set of entities that each user is interested in, from their past behaviors. We call this set the `seedset` of the user because it will later seed the random walk for that user. For this, we use the ProPPR program of Figure 2.2. The first rule states that the entity `E` belongs to the `seedset` of user `U` if `U` has reviewed `M` which is linked to entity `X` and `X` is related to `E`. Further, two entities are defined to be related if they are the same (Rule 2), or if there is a link between `X` and another entity `Z` which is related to `E` (Rule 3). This last rule is recursive. The `link` and the *type* (`isEntity`, `isItem` and `isUser`) information forms the knowledge graph in our case. Sample entries from the knowledge graph in the ProPPR format are shown in Figure 2.3. To find the seed set for `Alice`, we would issue the query $Q =$ `seedset(Alice, E)` to ProPPR.

$$\texttt{seedset}(\texttt{U},\texttt{E}) \leftarrow \texttt{reviewed}(\texttt{U},\texttt{M}), \texttt{link}(\texttt{M},\texttt{X}), \texttt{related}(\texttt{X},\texttt{E}), \texttt{isEntity}(\texttt{E}). \tag{2.1}$$

$$\texttt{related}(\texttt{X},\texttt{X}) \leftarrow \texttt{true}. \tag{2.2}$$

$$\texttt{related}(\texttt{X},\texttt{E}) \leftarrow \texttt{link}(\texttt{X},\texttt{Z}), \texttt{related}(\texttt{Z},\texttt{E}). \tag{2.3}$$

Figure 2.2: Seed Set generation

`link(Bridge of Spies, Tom Hanks)`	`isEntity(Tom Hanks)`
`link(Tom Hanks, Saving Private Ryan)`	`isEntity(Matt Damon)`
`link(Saving Private Ryan, Matt Damon)`	`isItem(Bridge of Spies)`

Figure 2.3: Example entries from the knowledge graph

ProPPR performs inference as a graph search. Given a program LP like that of Figure 2.2 and a query Q, ProPPR starts constructing a graph G, called the *proof graph*. This procedure is called "grounding". Each node in G represents a list of conditions that are yet to be proved. The root vertex v_0 represents Q. Then, it recursively adds nodes and edges to G as follows: let u be a node of the form $[R_1, R_2, ..., R_k]$ where R_* are its predicates. If ProPPR can find a fact

in the database that matches R_1, then the corresponding variables become bound and R_1 is removed from the list. Otherwise, ProPPR looks for a rule in LP of the form $S \leftarrow S_1, S_2, ..., S_l$, where S matches R_1. If it finds such a rule, it creates a new node with R_1 replaced with the body of S as, $v = [S_1, S_2, ..., S_l, R_2, ..., R_k]$, and links u and v. In the running example, v_0 is `seedset(Alice, E)` which is then linked to the node v_1 = `[reviewed(Alice,M), link(M,X), related(X,E), isEntity(E)]` obtained using Rule 1. Then, ProPPR would use the training (historical) data for `reviewed` to substitute `Saving Private Ryan` and `The Terminal` for `M` creating two nodes v_2 and v_3 as `[link(Saving Private Ryan,X), related(X,E), isEntity(E)]` and `[ link(The Terminal,X), related(X,E), isEntity(E)]` respectively. ProPPR would proceed by substituting for `X` from the knowledge graph and `related(X,E)` using the rules and so on until it reaches a node whose predicates have all been substituted. These nodes are the answer nodes because they represent a complete proof of the original query. The variable bindings used to arrive at these nodes can be used to answer the query. Examples would be:

$$\texttt{seedSet}(\texttt{Alice}, \texttt{E} = \texttt{TomHanks})$$
$$\texttt{seedSet}(\texttt{Alice}, \texttt{E} = \texttt{StevenSpielberg})$$

Note that such a graph construction could be potentially infinite. Therefore, ProPPR uses an approximate grounding mechanism to construct an approximate graph in time $O(\frac{1}{\alpha\epsilon})$, where ϵ is the approximation error and α is the reset parameter. Once such a graph has been constructed, ProPPR runs a Personalized PageRank algorithm with the start node as v_0 and ranks the answer nodes according to their PageRank scores.

The output of the program of Figure 2.2 is a ranked list of entities for the user `U` and the first K of these will be stored as `U`'s seed set. Note that the Personalized PageRank scoring will rank high those entities that are reachable from the movies that the user reviewed through multiple short paths, and rank low the entities that are either far and/or do not have multiple paths leading to them.

$$\texttt{reviewed}(\texttt{U},\texttt{M}) \leftarrow \texttt{seedset}(\texttt{U},\texttt{E}), \texttt{likesEntity}(\texttt{U},\texttt{E}), \texttt{related}(\texttt{E},\texttt{X}), \texttt{link}(\texttt{X},\texttt{M}), \texttt{isApplicable}(\texttt{U},\texttt{M}). \quad (2.4)$$
$$\texttt{likesEntity}(\texttt{U},\texttt{E}) \leftarrow \{\texttt{l}(\texttt{U},\texttt{E})\}. \quad (2.5)$$

Figure 2.4: `EntitySim`: ProPPR program for finding movies that a user may like using similarity measured using the graph links

After generating the seed set for each user, the next step in recommendation is to train a model and then use it to make predictions. As one method, we use the ProPPR program of Figure 2.4. It states that the user `U` may like a movie `M` if there is an entity `E` belonging to `U`'s seed set, and `U` likes `E`, and `E` is related to another entity `X`, which appears in the movie `M` (Rule 4). The predicate `related` is defined recursively as before. For the definition of the predicate `likesEntity`, note the term `{l(U,E)}`. This corresponds to a feature that is used to annotate the edge in which that rule is used. For example, if the rule is invoked with `U = Alice` and `E = Tom Hanks`, then the feature would be `l(Alice, Tom Hanks)`. In the training phase, ProPPR learns the weight of that feature from

the training data. During the prediction phase, ProPPR uses the learned weight of the feature as the weight of the edge. Note that these learned weights for each user-entity pair are not related to the ranking obtained from the seed set generation program of Figure 2.2, because these weights are specific to the prediction function.

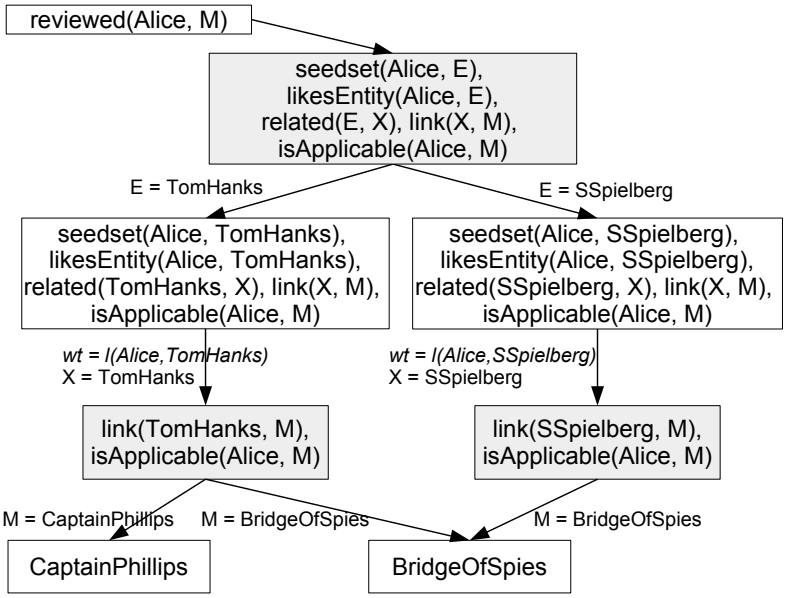

Figure 2.5: Sample grounding of the `EntitySim` ProPPR program

During the training phase, ProPPR grounds the program, similar to the process for the seed set generation discussed earlier. A sample grounding for `EntitySim` is depicted in Figure 3.2, where `Bridge Of Spies` and `Captain Phillips` are in the set of test examples for `Alice`. We may have other test examples for `Alice`, but if they are not provable using the rules (beyond a certain approximation error), they will not be present in the grounding. ProPPR then follows a procedure similar to that proposed by Backstrom et. al in [9], to train the random walker. This is done by learning a weight vector $\mathbf{w}$, which given a feature vector $\mathbf{\Phi}_{uv}$ for an edge in the graph $u \rightarrow v$, computes the edge strength as $f(\mathbf{w}, \mathbf{\Phi}_{uv})$, a function of the weight and the feature vectors. i.e. the probability of traversing the edge $P(v|u) \propto f(\mathbf{w}, \mathbf{\Phi}_{uv})$. Our method uses $f(\mathbf{w}, \mathbf{\Phi}_{uv}) = e^{\mathbf{w} \cdot \mathbf{\Phi}_{uv}}$.

During the training phase, learning of $\mathbf{w}$ is posed as an optimization problem as given below:

$$-\sum_{k=1}^{m} \Big(\sum_{i=1}^{I_m} \log \mathbf{p}[u_i^{k+}] + \sum_{j=1}^{J_m} \log(1 - \mathbf{p}[u_j^{k-}]) \Big) + \mu \|\mathbf{w}\|_2^2 \tag{2.6}$$

where, $\mathbf{p}$ is the PageRank vector computed with the edge weights obtained with $\mathbf{w}$. The optimization function of Equation 2.6 used by ProPPR is the standard positive-negative log loss function instead of the pairwise loss function used in [9]. To learn $\mathbf{w}$, we use AdaGrad [34] instead of the quasi-Newton method used in [9] and SGD used in [126]. The initial learning rate used by AdaGrad and the regularization parameter μ are set to 1. For a thorough description of ProPPR, we refer the reader to [126] and [125].

2.3.3 Approach 2: TypeSim

The EntitySim method uses only the knowledge graph links to learn about user's preferences. However, recall that we are in fact using a heterogenous information network where in addition to the link information, we also know the "type" of the entities. For example, we know that New York City is of type City and Tom Hanks is of type Actor. To leverage this additional type information, we extend the EntitySim method to TypeSim method as shown in Figure 2.6.

$$\texttt{reviewed}(\texttt{U},\texttt{R}) \leftarrow \texttt{seedset}(\texttt{U},\texttt{E}), \texttt{likesEntity}(\texttt{U},\texttt{E}), \texttt{popularEntity}(\texttt{E}), \texttt{related}(\texttt{E},\texttt{X}), \texttt{link}(\texttt{X},\texttt{R}), \texttt{isApplicable}(\texttt{U},\texttt{R}). \quad (2.7)$$

$$\texttt{likesEntity}(\texttt{U},\texttt{E}) \leftarrow \{\texttt{l}(\texttt{U},\texttt{E})\}. \quad (2.8)$$

$$\texttt{popularEntity}(\texttt{E}) \leftarrow \texttt{entityOfType}(\texttt{E},\texttt{T}), \texttt{popularType}(\texttt{T})\{\texttt{p}(\texttt{E})\}. \quad (2.9)$$

$$\texttt{popularType}(\texttt{T}) \leftarrow \{\texttt{p}(\texttt{T})\}. \quad (2.10)$$

$$\texttt{typeAssoc}(\texttt{X},\texttt{Z}) \leftarrow \texttt{entityOfType}(\texttt{X},\texttt{S}), \texttt{entityOfType}(\texttt{Z},\texttt{T}), \texttt{typeSim}(\texttt{S},\texttt{T}). \quad (2.11)$$

$$\texttt{typeSim}(\texttt{S},\texttt{T}) \leftarrow \{\texttt{t}(\texttt{S},\texttt{T})\}. \quad (2.12)$$

Figure 2.6: TypeSim method for recommendations

TypeSim models the general popularity of each of the node types in Rule 10 by learning the overall predictability offered by the type of the entity using $\texttt{p}(\texttt{T})$. For example, nodes of the type Actor may offer more insight into users' preferences than those of type Country. Note that, the learned weight is not specific to the user and hence its weight is shared by all the users. Similar to Rule 10, in Rule 9, the model learns the overall predictability offered by the entity itself, independent of the user using $\texttt{p}(\texttt{E})$. For example, it could be that the movies directed by Steven Spielberg are more popular than those by other lesser known directors. TypeSim also models a general traversal probability between two types using Rules 11 and 12. For example, Actor $\rightarrow$ Movie is generally a more predictive traversal on the graph compared to Country $\rightarrow$ Movie. These weights are incorporated into the prediction rule of EntitySim as shown in Rule 7.

2.3.4 Approach 3: GraphLF

One of the most successful types of Collaborative Filtering (CF) are Latent Factor (LF) models [56]. They try to uncover hidden dimensions that characterize each of the objects thus mapping users and items onto the same feature space for an improved recommendation performance. Koren et al. note in [56] that for movies, latent factors might measure obvious dimensions such as comedy versus drama, amount of action, or orientation to children, as well as less well-defined dimensions such as depth of character development, or quirkiness, or even uninterpretable dimensions. For users, each factor measures how much the user likes movies that score high on the corresponding factor. Singular Value Decomposition (SVD) is one of the more popular methods of generating LF models for recommendation. An SVD method assigns users and items with values along each of the hidden dimensions while minimizing a loss function over the predicted and actual rating matrix.

The main attraction of Collaborative Filtering methods is that they do not require any knowledge about the users or items and predict solely based on the rating matrix. Similarly, the main attraction of Latent Factor based CF models is that they develop a general representation of users and items based on the ratings data that are more generalizable and often indiscernible in the raw data.

$$\texttt{reviewed}(\texttt{U},\texttt{R}) \leftarrow \texttt{related}(\texttt{U},\texttt{E}), \texttt{related}(\texttt{E},\texttt{X}), \texttt{link}(\texttt{X},\texttt{R}), \texttt{isApplicable}(\texttt{U},\texttt{R}). \quad (2.13)$$

$$\texttt{related}(\texttt{U},\texttt{E}) \leftarrow \texttt{seedset}(\texttt{U},\texttt{E}), \texttt{simLF}(\texttt{U},\texttt{E}). \quad (2.14)$$

$$\texttt{related}(\texttt{X},\texttt{X}) \leftarrow. \quad (2.15)$$

$$\texttt{related}(\texttt{X},\texttt{Y}) \leftarrow \texttt{link}(\texttt{X},\texttt{Z}), \texttt{simLF}(\texttt{X},\texttt{Z}), \texttt{related}(\texttt{Z},\texttt{Y}). \quad (2.16)$$

$$\texttt{simLF}(\texttt{X},\texttt{Y}) \leftarrow \texttt{isDim}(\texttt{D}), \texttt{val}(\texttt{X},\texttt{D}), \texttt{val}(\texttt{Y},\texttt{D}). \quad (2.17)$$

$$\texttt{val}(\texttt{X},\texttt{D}) \leftarrow \{\texttt{v}(\texttt{X},\texttt{D})\}. \quad (2.18)$$

Figure 2.7: GraphLF method for recommendations

Given that we have access to a KG that connects the items via different entities, the third approach that we present in this chapter, `GraphLF`, integrates latent factorization and graph-based recommendation. The overall ruleset is defined in Figure 2.7. Its principal rule is the definition of Latent Factor Similarity `simLF` in Rules (17) and (18). Essentially, `simLF` of two input entities `X` and `Y` is measured by first picking a dimension `D`, and then measuring the values of `X` and `Y` along `D`. If there are many dimensions `D` along which the values of both `X` and `Y` are high, then probabilistically, their similarity scores will also be high. The value of an entity `X` along dimension `D`, `val(X,D)` is learned from the data during the training phase, as defined in Rule (18).

Note how the recursive definition of the relatedness of two entities `related(X,Y)` in Rule (16) has now changed to account for their latent factor similarity in addition to the presence of a link between them. Also, the original prediction rule has changed in Rule (13) to use a new relatedness score between the user and the entity. Essentially, the definition of `related(U,E)` in Rule (14) replaces the earlier predicate `likesEntity(U,E)` with the latent factor similarity `simLF(U,E)`, between the user and an entity belonging to their seedset. Therefore, the model no longer learns a weight for each user-entity pair, and instead learns weights for the users and entities separately along each of the dimensions.

It is also important to note that `GraphLF` is type-agnostic unlike `TypeSim` and `HeteRec_p`. Types are not always available, especially for general-purpose graphs like the Wikipedia. Therefore, being type-agnostic is a desirable property and increases its applicability to a wide range of data domains.

2.4 Experiments and Results

2.4.1 Datasets

To test our methods, we use two well known large datasets:

- **Yelp2013**: This is the 2013 version of the Yelp Dataset Challenge[2] released by Yelp[3], available now at Kaggle[4]. We use this earlier version instead of the latest version from the Yelp Dataset Challenge for the purposes of comparing with the published results of the `HeteRec_p` algorithm. Similar to [142], we discard users with only 1 review entry.
- **IM100K**: This dataset is built from the MovieLens-100K dataset[5] unified with the content — director, actor, studio, country, genre, tag — parsed out from their corresponding IMDb pages[6]. We could not obtain the dataset used in [142], which we will refer to as `IM100K-UIUC`. Our dataset `IM100K*` is a close approximation to it, created from the same MovieLens-100K dataset, but we have recovered the content of 1566 movies of the total 1682 movies compared to 1360 in [142], and have 4.8% more reviews than [142].

For all the datasets, similar to [142], we sort the reviews in the order of their timestamps, and use the older 80% of the reviews as training examples and the newer 20% of the reviews as the test examples. The overall statistics of these datasets, viz. the number of users, the number of items and the total number of reviews/ratings, are listed in Table 4.3.1.

Dataset	#Users	#Items	#Ratings
`Yelp`	43,873	11,537	229,907
`IM100K-UIUC`	943	1360	89,626
`IM100K*`	940	1566	93,948

Table 2.1: Dataset Statistics

2.4.2 Experimental Setup

We evaluate the performance using the standard metrics, Mean Reciprocal Rank (MRR), and Precision at 1, 5 and 10 (P@1, P@5 and P@10) [76].

In our experiments, we found that any reasonable choice for the seed set size worked well enough. A fixed size serves to constrain the number of parameters learned and hence, the complexity of the model.

In the following sections, we compare our methods to the state-of-the-art method `HeteRec_p` proposed in [142] on the two datasets. We also compare the performance to a Naïve Bayes (`NB`) baseline, which represents a recommender system that uses only the content of the item without the knowledge graph and link information, to make predictions. Naïve Bayes classifiers have been previously shown to be as effective as certain computationally intensive algorithms [95]. For each user, the `NB` system uses the entities of the items in that user's training set as the features to train the model. These are the same entities used by our methods. We use the probabilities output by the classifier to rank the pages. The implementation used is from the Mallet [83] package. Since `HeteRec_p` was shown to be better than Popularity (which shows the globally popular items to users), Co-Click

[2] https://www.yelp.com/dataset_challenge
[3] http://www.yelp.com/
[4] https://www.kaggle.com/c/yelp-recsys-2013/data
[5] http://grouplens.org/datasets/movielens/
[6] http://www.imdb.com/

(which uses the co-click probabilities to find similar items), Non–Negative Matrix Factorization[33] and Hybrid-SVM (which uses a Ranking SVM[52] on metapaths) in [142], we refrain from repeating those comparisons again.

2.4.3 Performance Comparison on Yelp

Method	P@1	P@5	P@10	MRR	Settings
HeteRec_p	0.0213	0.0171	0.0150	0.0513	*published results*
EntitySim	0.0221	0.0145	0.0216	0.0641	$n = 20$
TypeSim	0.0444	**0.0188** [↑ 10%]	**0.0415** [↑ 176%]	**0.0973** [↑ 89%]	$n = 20$
GraphLF	**0.0482** [↑ 126%]	0.0186	0.0407	0.0966	$n = 20, dim = 10$
NB	0	0.0012	0.0013	0.0087	

Table 2.2: Performance comparison on Yelp: The best score for each metric is highlighted in blue and the lowest score in red. [↑ x%] gives the percent increase compared to the corresponding HeteRec_p score

The performance of the algorithms presented in this chapter as well as the baselines on the Yelp data are tabulated in Table 2.2. It can be seen from the results that our methods outperform HeteRec_p by a large margin. For example, GraphLF is 126% better on P@1 and TypeSim is 89% better on MRR.

Also, we can note that using the type information (TypeSim) improves the performance drastically compared to EntitySim. For example, P@1 improves by 118% and MRR by 51%. Similarly, we can note that discovering the latent factors in the data (GraphLF) also improves the performance tremendously compared to its simpler counterpart (EntitySim). For example, P@1 improves by 115% and MRR by 37%.

However, there is no clear winner when comparing TypeSim and GraphLF: while the former scores better on MRR, the latter is better on P@1.

The NB baseline's performance is poor, but that was expected since the dataset is extremely sparse.

2.4.4 Performance Comparison on IM100K

The performance of HeteRec_p on the IM100K-UIUC dataset, and that of the algorithms presented in this chapter and the Naïve Bayes baseline on the IM100K* dataset, are tabulated in Table 2.3.

As noted in Section 2.4.1, the IM100K-UIUC dataset and the IM100K* dataset are slightly different from each other. Therefore, we cannot compare the performance of the methods directly; the most that can be said is that the methods appear to be comparable.

A more interesting and surprising result is that the simplest of the methods, NB and EntitySim, perform as well or better than the more complex TypeSim and GraphMF. In fact, NB outperforms

Method	P@1	P@5	P@10	MRR	Settings
HeteRec_p (on IM100K-UIUC)	0.2121	**0.1932**	0.1681	**0.553**	*published results*
EntitySim	0.3485	0.1206	**0.2124** [↑ 26.3%]	0.501 [↓ −9.4%]	$n = 10$
TypeSim	**0.353** [↑ 66.4%]	0.1207 [↓ −37.5%]	0.2092	0.5053	$n = 10$
GraphLF	0.3248	0.1207 [↓ −37.5%]	0.1999	0.4852	$n = 10, dim = 10$
NB	0.312	0.1202	0.1342	0.4069	

Table 2.3: Performance comparison on IM100K (IM100K-UIUC & IM100K*): The best score for each metric is highlighted in blue and the lowest score in red. [↑ x%] gives the percent increase compared to the corresponding HeteRec_p score and [↓ x%], the percent decrease.

HeteRec_p on the P@1 metric. This leads us to speculate that when there are enough training examples per user and enough signals per item, simple methods suffice. We explore this conjecture more fully below.

2.4.5 Effect of Dataset Density on Performance

In the previous two sections, we saw how on the Yelp dataset TypeSim and GraphLF performed extremely well in comparison to the EntitySim method, whereas on the IM100K dataset, the latter was as good as or better than the former two. In this section, we explore this phenomenon further.

An important difference between the two datasets is that Yelp is a complete real world dataset with review frequencies exhibiting a power law distribution, while IM100K is a filtered version of a real world dataset counterpart, as noted by the authors of [142]: in IM100K dataset, each user has rated at least 20 movies.

To quantitatively compare their differences, we define the Density of a dataset as $\frac{\#\text{reviews}}{\#\text{users} \times \#\text{items}}$, which is the ratio of filled entries in the rating matrix to its size. Using this definition, the density of Yelp was found to be only 0.00077 whereas that of IM100K* was 0.06382.

To study this further, we created 4 additional datasets from Yelp by filtering out users and businesses that have less than k reviews, where k =10, 25, 50 and 100. The MRR scores of our methods and the NB baseline with varying k is plotted in Figure 2.8 (with the left y axis). These are with a seedset of size 20. The graph densities at the different k are also plotted in the same Figure 2.8 (in green with the right y axis). Note that the density increases to 0.11537 at $k = 100$, which is 148 times higher than the density at $k = 2$.

From the figure, we can see that when the dataset is the least dense ($k = 2$), the more complex methods TypeSim and GraphLF perform much better than the simple EntitySim. However, as the density increases with larger k, we can observe that EntitySim eventually equals TypeSim and comes within 1% of that of GraphLF, at $k = 100$. Therefore, we can deduce that, when we have enough training examples and a dense graph connecting them, a simple method like EntitySim

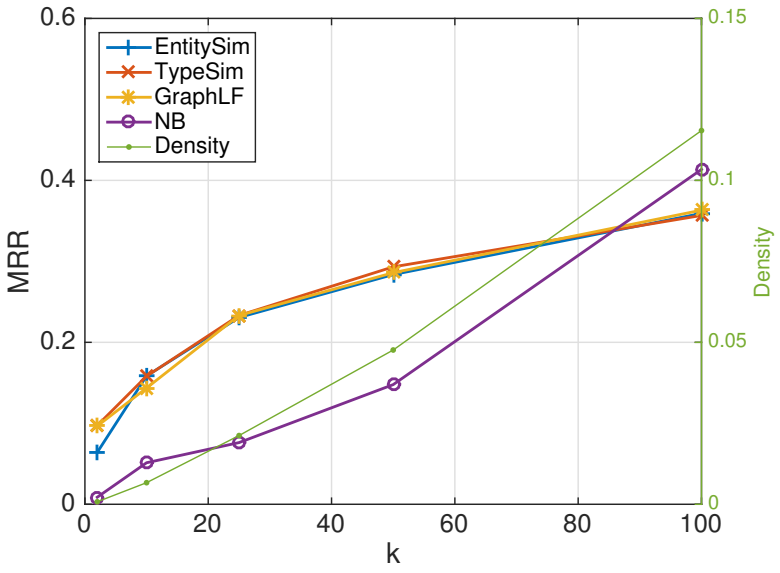

Figure 2.8: Performance of different methods with varying graph densities on `Yelp`

that predicts based on just the links in the graph can give good results.

Another observation from Figure 2.8 is that the `NB` recommender, whose performance is poor at low graph densities — 633% worse than `EntitySim` — slowly improves with increasing k to eventually better all the KG based methods at $k = 100$ (14% better than `GraphLF`). From this, we conjecture that when there are enough training examples per user, we can produce accurate predictions using a simple classifier based recommender like `NB`. However, at slightly lower densities, like at $k = 50$, the knowledge graph is a valuable source of information for making recommendations, as can be seen from the figure, where `NB` is 92% below `EntitySim` at $k = 50$.

2.5 Contributions

In this chapter, we presented three methods for performing knowledge graph based recommendations using a general-purpose probabilistic logic system called ProPPR. Our methods use the link structure of the knowledge graph as well as type information about the entities to improve predictions. The more complex of the models discussed in this chapter combined the strengths of latent factorization with graphs, and is type agnostic. By comparing our methods to the published results of the state-of-the-art method that uses knowledge-graphs in generating recommendations, we showed that our methods were able to achieve a large improvement in performance.

We also studied the behavior of the methods with changing dataset densities and showed that at higher densities, just the graph link structure sufficed to make accurate recommendations and the type information was redundant. In addition, we showed that in sparse datasets, the knowledge graph is a valuable source of information, but its utility diminishes when there are enough training examples per user.

Chapter 3

Ongoing Work: Entity based Explanations using Knowledge Graphs

The focus of this chapter is a system to generate explanations for Knowledge Graph (KG) -based recommendation. Although a number of explanation schemes have been proposed in the past, there has been no work which produces explanations for KG-based recommenders. In this chapter, we present a method to jointly rank items and entities in the KG such that the entities can serve as an explanation for the recommendation.

Our technique can be run without training, thereby allowing faster deployment in new domains. Once enough data has been collected, it can then be trained to yield better performance. It can also be used in a dialog setting, where a user interacts with the system to refine its suggestions. The main difficulty in evaluating an entity-based explanation system is the unavailability of a gold standard data. In almost all datasets available for this task, only the user-item interaction is known, This work was published in the 11^{th} ACM Conference on Recommender Systems (RecSys '17) as a Poster[21].

3.1 Related Work: Knowledge Graphs for Explanation

[46] was an early work that assessed different ways of explaining recommendations in a CF-based recommender system. They reported that using a histogram of ratings by the user's neighbors as well as specifying if any of their favorite actors appear in the movie were perceived well by the users. A recent work [2] proposed to constrain MF such that it favors recommendations that are explainable. In their work, a recommendation is explainable if it there are enough known examples to reason the recommendation as "x other people like you have liked this item in the past" (user-based neighbor style) or "you have liked y other items like this in the past" (item-based neighbor style). A similar method was proposed by the same authors in [1] for a CF method that uses Restricted Boltzmann Machines (RBM).

In content-based recommenders, the explanations revolve around the profile or content associated with the user and the item. The system of [14] simply displayed keyword matches between the user's profile and the books being recommended. Similarly, [117] proposed a method called 'Tagsplanations', which showed the degree to which a tag is relevant to the item, and the sentiment of the user towards the tag.

With the advent of social networks, explanations that leverage social connections have also gained attention. For example, [105] produced explanations that showed whether a good friend of the user has liked something, where friendship strength was computed from their interactions on Facebook.

More recent research has focused on providing explanations that are extracted from user written reviews for the items. [144] extracted phrases and sentiments expressed in the reviews and used them to generate explanations. [77] uses topics learned from the reviews as aspects of the item, and uses the topic distribution in the reviews to find useful or representative reviews.

Knowledge Graphs have been shown to improve the performance of recommender systems in the past. [140] proposed a meta-path based method that learned paths consisting of node types in a graph. Similarly, [92] used paths to find the top-N recommendations in a learning-to-rank framework. A few methods such as [85, 87] rank items using Personalized PageRank. In these methods, the entities present in the text of an item are first mapped to entities in a knowledge graph. [19] proposed probabilistic logic programming models for recommendation on knowledge graphs. A recent work [124] studies the problem of medicine recommendation using a Medical Knowledge Graph by embedding the graph into a low dimensional space. [132] and [131] use KGs for reasoning over relations. None of the above KG-based recommenders attempted to generate explanations.

[116] is a related work on graphs that proposed an algorithm to find a node in the graph that is connected directly or indirectly to a given input of Q nodes. Although they have similar elements to our approach, their method is not intended for recommendations.

A recent work [58] evaluated different ways of showing explanations for recommendations produced using HyPER [57], a recommender system that uses Probabilistic Soft Logic (PSL) [8]. PSL is similar to ProPPR since both use probabilistic logic but unlike PSL, ProPPR uses a "local" grounding procedure, which leads to small inference problems, even for large databases [126]. The user study in [58] showed that visualizing explanations using Venn diagrams was preferred by the users over other visualizations like pathways between nodes. Their system generates explanations that only show why a user would like the item, and not why they may not like it. Also, the graph used in their method is a friend network and not a Knowledge Graph that connects users and items using the content associated with them.

3.2 Explanation Method

The method presented in this Chapter builds on the work described in Chapter 2 by using ProPPR [126] for learning to recommend. Our technique proceeds in two main steps. First, it uses ProPPR to jointly rank items and entities for a user. Second, it consolidates the results into recommendations and explanations.

To use ProPPR to rank items and entities, we reuse the notion of similarity between graph nodes defined in Equations 2.2 and 2.3, from Chapter 2. The model has two sets of rules for ranking: one set for joint ranking of movies that the user would like, together with the most likely reason (Figure 3.1), and a similar set for movies that the user would not like. In Figure 3.1, Rule 3 states that a user `U` will like an entity `E` and a movie `M` if the user likes the entity, and the entity is related (`sim`) to the movie. The clause `isMovie` ensures that the variable `M` is bound to a movie, since `sim` admits all types of entities. Rule 3 invokes the predicate `likes(U,E)`, which holds for an entity `E` if the user has explicitly stated that they like it (Rule 4), or if they have provided positive feedback (e.g.

clicked, thumbs up, high star rating) for a movie `M` containing (via `link(M,E)`) the entity (Rule 5). The method for finding movies and entities that the user will dislike is similar to the above, except 'like' is replaced with 'dislike'.

$$\texttt{willLike}(\texttt{U},\texttt{E},\texttt{M}) \leftarrow \texttt{likes}(\texttt{U},\texttt{E}), \texttt{sim}(\texttt{E},\texttt{M}), \texttt{isMovie}(\texttt{M}). \tag{3.1}$$

$$\texttt{likes}(\texttt{U},\texttt{E}) \leftarrow \texttt{likesEntity}(\texttt{U},\texttt{E}). \tag{3.2}$$

$$\texttt{likes}(\texttt{U},\texttt{E}) \leftarrow \texttt{likesMovie}(\texttt{U},\texttt{M}), \texttt{link}(\texttt{M},\texttt{E}). \tag{3.3}$$

Figure 3.1: Predicting likes

To jointly rank the items and entities, we use ProPPR to query the `willLike(U,E,M)` predicate with the user specified and the other two variables free. Then, the ProPPR engine will ground the query into a proof graph by replacing each variable recursively with literals that satisfy the rules from the KG [19, 126]. A sample grounding when queried for a user `alice` who likes `tom_hanks` and the movie `da_vinci_code` is shown in Figure 3.2.

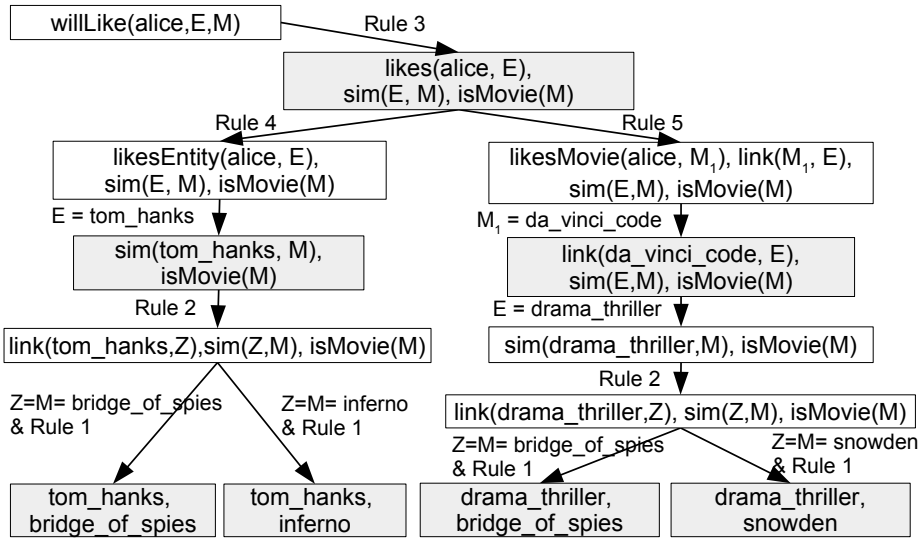

Figure 3.2: Sample grounding for predicting likes

After constructing the proof graph, ProPPR runs a Personalized PageRank algorithm with `willLike(alice, E, M)` as the start node. In this simple example, we will let the scores for `(tom_hanks, bridge_of_spies)`, `(tom_hanks, inferno)`, `(drama_thriller, bridge_of_spies)`, and `(drama_thriller, snowden)`, be 0.4, 0.4, 0.3 and 0.3 respectively.

Now, let us suppose that `alice` has also specified that she dislikes `crime` movies. If we follow the grounding procedure for dislikes and rank the answers, we may obtain `(crime, inferno)` with score 0.2. Our system then proceeds to consolidate the recommendations and the explanations by grouping by movie names, adding together their 'like' scores and deducting their 'dislike' scores. For each movie, the entities can be ranked according to their joint score. The end result is a list of reasons which can be shown to the user:

1. `bridge_of_spies`, score = 0.4 + 0.3 = 0.7, reasons = { tom_hanks, drama_thriller }
2. `snowden`, score = 0.3, reasons = { drama_thriller }
3. `inferno`, score = 0.4 - 0.2 = 0.2, reasons = { tom_hanks, (-ve) crime }

3.3 Real World Deployment and Evaluation

The technique presented in this Chapter is currently being used as the backend of a personal agent running on mobile devices for recommending movies [6] undergoing Beta testing. The knowledge graph for recommendations is constructed from the weekly dump files released by `imdb.com`. The personal agent uses a dialog model of interaction with the user. In this setting, users are actively involved in refining the recommendations depending on what their mood might be. For example, for a fun night out with friends, a user may want to watch an action movie, whereas when spending time with her significant other, the same user may be in the mood for a romantic comedy.

Qualitative feedback is being collected from the users of the Beta testing, where they are requested to answer the following questions on a Likert-type scale:

1. Did you like that the app explained why a recommendation was made?
2. Did you think that the explanation was reasonable?

However, the users are not requested to give a feedback about the explanations after *every* interaction due to concerns that it might reduce the usability of the app.

3.4 Contributions

Knowledge graphs have been shown to improve recommender system accuracy in the past. However, generating explanations to help users make an informed choice in KG-based systems has not been attempted before. In this chapter, we presented a method to produce a ranked list of entities as explanations by jointly ranking them with the corresponding movies.

Chapter 4

Completed Work: Rating Prediction from Reviews using TransNets

Using review text for predicting ratings has been shown to greatly improve the performance of recommender systems [11, 70, 78], compared to Collaborative Filtering (CF) techniques that use only past ratings [56, 101]. Recent advances in Deep Learning research have made it possible to use Neural Networks in a multitude of domains including recommender systems, with impressive results. Most neural recommender models [10, 35, 53, 65, 122] have focussed on the *content* associated with the user and the item, which are used to construct their latent representations.

Review text, unlike content, is not a property of only the user or only the item; it is a property associated with their joint interaction. In that sense, it is a *context* [3] feature. Only a few neural net models [5, 103, 146] have been proposed to date that use review text for predicting the rating. Of these, one recent model, *Deep Cooperative Neural Networks* (DeepCoNN) [146] uses neural nets to learn a latent representation for the user from the text of all reviews written by her and a second latent representation for the item from the text of all reviews that were written for it, and then combines these two representations in a regression layer to obtain state-of-the-art performance on rating prediction. However, as we will show, much of the predictive value of review text comes from reviews of the target user for the target item, which is not available at test time. We introduce a way in which this information can be used in training the recommender system, such that when the target user's review for the target item is not available at the time of prediction, an approximation for it is generated, which is then used for predicting the rating. Our model, called ***Trans****formational Neural* ***Net****works* (TransNets), extends the DeepCoNN model by introducing an additional latent layer representing an approximation of the review corresponding to the target user-target item pair. We then regularize this layer, at training time, to be similar to the latent representation of the actual review written by the target user for the target item. Our experiments illustrate that TransNets and its extensions give substantial improvements in rating prediction.

This work was published in the 11^{th} ACM Conference on Recommender Systems (RecSys '17) [20].

4.1 Related Work: Recommendation using Reviews

4.1.1 Non-Neural Models

The *Hidden Factors as Topics* (HFT) model [78] aims to find topics in the review text that are correlated with the latent parameters of users. They propose a transformation function which converts user's latent factors to the topic distribution of the review, and since the former exactly defines the latter, only one of them is learned. A modified version of HFT is the *TopicMF* model [11], where the goal is to match the latent factors learned for the users and items using MF with the topics learned on their joint reviews using a Non-Negative Matrix Factorization, which is then jointly optimized with the rating prediction. In their transformation function, the proportion of a particular topic in the review is a linear combination of its proportion in the latent factors of the user and the item, which is then converted into a probability distribution over all topics in that review. Unlike these two models, TransNet computes each factor in the transformed review from a non-linear combination of any number of factors from the latent representations of either the user or the item or both. Another extension to HFT is the *Rating Meets Reviews* (RMR) model [70] where the rating is sampled from a Gaussian mixture.

The *Collaborative Topic Regression* (CTR) model proposed in [121] is a content based approach, as opposed to a context / review based approach. It uses LDA [15] to model the text of documents (scientific articles), and a combination of MF and content based model for recommendation. The *Rating-boosted Latent Topics* (RBLT) model of [111] uses a simple technique of repeating a review r times in the corpus if it was rated r, so that features in higher rated reviews will dominate the topics. *Explicit Factor Models* (EFM) proposed in [145] aims to generate explainable recommendations by extracting explicit product features (aspects) and users' sentiments towards these aspects using phrase-level sentiment analysis.

4.1.2 Neural Net Models

One recent model to successfully employ neural networks at scale for rating prediction is the *Deep Cooperative Neural Networks* (DeepCoNN) [146], which will be discussed in detail in Section 4.2. Prior to that work, [5] proposed two models: *Bag-of-Words regularized Latent Factor* model (BoWLF) and *Language Model regularized Latent Factor* model (LMLF), where MF was used to learn the latent factors of users and items, and likelihood of the review text, represented either as a bag-of-words or an LSTM embedding [48], was computed using the item factors. [103] proposed a CNN based model identical to DeepCoNN, but with attention mechanism to construct the latent representations, the inner product of which gave the predicted ratings.

Some of the other past research uses neural networks in a CF setting with content, but not reviews. The *Collaborative Deep Learning* (CDL) model [122] uses a Stacked De-noising Auto Encoder (SDAE) [118] to learn robust latent representations of items from their content, which is then fed into a CTR model [121] for predicting the ratings. A very similar approach to CDL is the *Deep Collaborative Filtering* (DCF) method [65] which uses Marginalized De-noising Auto-Encoder (mDA) [22] instead. The *Convolutional Matrix Factorization* (ConvMF) model [53] uses a CNN to process the description associated with the item and feed the resulting latent vectors into a PMF model for rating prediction. The *Multi-View Deep Neural Net* (MV-DNN) model [35] uses a deep neural net to map

user's and item's content into a shared latent space such that their similarity in that space is maximized. [91] proposed to generate the latent factors of items – music in this case— from the content, audio signals. The predicted latent factors of the item were then used in a CF style with the latent factors of the user. [10] also proposed a similar technique but adapted to recommending scientific-articles. [27] used a deep neural net to learn a latent representation from video content which is then fed into a deep ranking network. [45] is a simple extension to MF with additional non-linear layers instead of the the dot product.

Prior research has also used deep neural nets for learning latent factors from ratings alone, i.e., without using any content or review. The *Collaborative De-noising Auto-Encoder* model (CDAE) [130] learns to reconstruct user's feedback from a corrupted version of the same. In [114], the authors propose a TransE like framework for the user-item (non-graph) data, where the goal is to learn to generate a relation r that would translate the user embedding u to that of the item embedding i by minimizing $||u + r - i|| \approx 0$.

4.1.3 Comparison to Related Architectures and Tasks

Student-Teacher Models

The model proposed in this Chapter is a type of Student-Teacher model, albeit not in their conventional setting. Student-Teacher models [17, 47] also have two networks: a Teacher Network (similar to Target Network in our model), which is large and complex, and typically an ensemble of different models, is first trained to make predictions, and a much simpler Student Network (similar to Source Network), which learns to emulate the output of the Teacher Network, is trained later.

There are substantial differences between Student-Teacher models and TransNets in how they are structured. Firstly, in Student-Teacher models, the input to both the student and the teacher models are the same. For example, in the case of digit recognition, both networks input the same image of the digit. However, in TransNets, the inputs to the two networks are different. In the Target, there is only one input – the review by $user_A$ for an $item_B$ designated as rev_{AB}. But, in the Source, there are two inputs: all the reviews written by $user_A$ sans rev_{AB} and all the reviews written for $item_B$ sans rev_{AB}. Secondly, in Student-Teacher models, the Teacher is considerably complex in terms of width and depth, and the Student is more light-weight, trying to mimic the Teacher. In TransNets, the complexities are reversed. The Target is lean while the Source is heavy-weight, often processing large pieces of text using twice the number of parameters as the Target. Thirdly, in Student-Teacher models, the Teacher is pre-trained whereas in TransNets the Target is trained simultaneously with the Source. A recently proposed Student-Teacher model in [50] does train both the Student and the Teacher simultaneously. Also, in Student-Teacher models, the emphasis is on learning a simpler and easier model that can achieve similar results as a very complex model. But in TransNets, the objective is to learn how to transform a source representation to a target representation. Recently, Student-Teacher models where shown to work on sequence-level tasks as well[54].

Sentiment Analysis

Part of the model proposed in this Chapter, referred to as *Target Network* later, resembles a sentiment analyzer because it inputs a user's review for an item and predict that user's rating for that item. In [106], the authors proposed a Recursive Neural Tensor Network that could achieve high accuracy

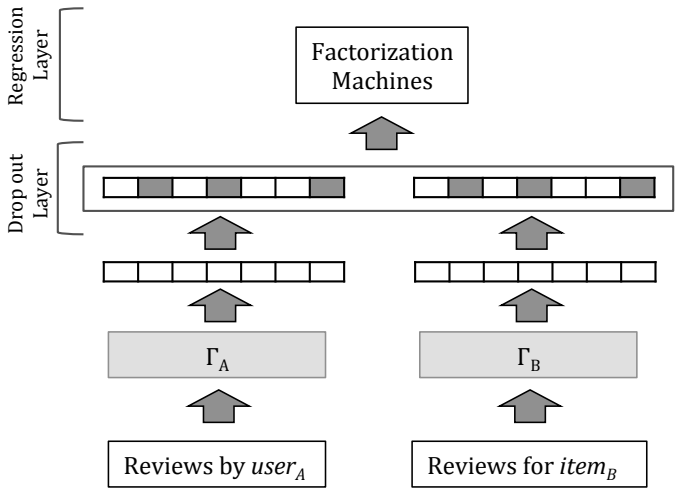

Figure 4.1: DeepCoNN model for predicting rating

levels for sentence and phrase level sentiment prediction. [59] proposed a tree-structured deep NN for predicting sentiment from the discourse structure. [128] uses a character-level CNN for Twitter sentiment analysis whereas [96] uses an LSTM on utterances in a video to compute its sentiment. In [143], the authors also consider other inputs such as a user's facial expressions or tone of voice along with the sentences to predict the sentiment. [7] computes an aspect-level sentiment using an aspect graph and a rhetorical structure tree.

4.2 The TransNet Method

4.2.1 CNNs to process text

We process text using the same approach as the current state-of-the-art method for rating prediction, *DeepCoNN* [146]. The basic building block, referred to as a *CNN Text Processor* in the rest of this chapter, is a Convolutional Neural Network (CNN) [62] that inputs a sequence of words and outputs a n-dimensional vector representation for the input, i.e., the *CNN Text Processor* is a function $\Gamma : [w_1, w_2, ..., w_T] \rightarrow \mathbb{R}^n$. For more details, please refer to [20].

4.2.2 The DeepCoNN model

To compute the rating r_{AB} that $user_A$ would assign to $item_B$, the DeepCoNN model of [146] uses two *CNN Text Processors* side by side as shown in Figure 4.1. The first one processes the text labeled $text_A$, which consists of a concatenation of all the reviews that $user_A$ has written and produces a representation, x_A. Similarly, the second processes the text called $text_B$, which consists of a concatenation of all the reviews that have been written about $item_B$ and produces a representation, y_B. Both outputs are passed through a dropout layer [108]. Dropout is a function $\delta : \mathbb{R}^n \rightarrow \mathbb{R}^n$, that suppresses the output of some of the neurons randomly and is a popular technique for regularizing a network. We let $\bar{x}_A = \delta(x_A)$ and $\bar{y}_B = \delta(y_B)$ denote the output of the dropout layer applied on

x_A and y_B.

The model then concatenates the two representations as $z = [\bar{x}_A, \bar{y}_B]$ and passes it through a regression layer consisting of a *Factorization Machine* (FM) [99]. The FM computes the second order interactions between the elements of the input vector as:

$$\hat{r}_{AB} = w_0 + \sum_{i=1}^{|z|} w_i z_i + \sum_{i=1}^{|z|} \sum_{j=i+1}^{|z|} \langle \mathbf{v}_i, \mathbf{v}_j \rangle z_i z_j$$

where $w_0 \in \mathbb{R}$ is the global bias, $\mathbf{w} \in \mathbb{R}^{2n}$ weights each dimension of the input, and $\mathbf{V} \in \mathbb{R}^{2n \times k}$ assigns a k dimensional vector to each dimension of the input so that the pair-wise interaction between two dimensions i and j can be weighted using the inner product of the corresponding vectors $\mathbf{v}_i$ and $\mathbf{v}_j$. Note that the FM factorizes the pair-wise interaction, and therefore requires only $O(nk)$ parameters instead of $O(n^2)$ parameters which would have been required otherwise, where k is usually chosen such that $k \ll n$. This has been shown to give better parameter estimates under sparsity [99] (sparsity means that in the user-item rating matrix, most cells are empty or unknown). FMs have been used successfully in large scale recommendation services like online news[139].

FMs can be trained using different kinds of loss functions including least squared error (L_2), least absolute deviation (L_1), hinge loss and logit loss. In our experiments, L_1 loss gave a slightly better performance than L_2. DeepCoNN [146] also uses L_1 loss. Therefore, in this work, all FMs are trained using L_1 loss, defined as:

$$loss = \sum_{(u_A, i_B, r_{AB}) \in D} |r_{AB} - \hat{r}_{AB}|$$

4.2.3 Limitations of DeepCoNN

DeepCoNN model has achieved impressive performance surpassing that of the previous state-of-the-art models that use review texts, like the Hidden Factors as Topics (HFT) model [78], Collaborative Topic Regression (CTR) [121] and Collaborative Deep Learning (CDL) [122], as well as Collaborative Filtering techniques that use only the rating information like Matrix Factorization (MF) [56] and Probabilistic Matrix Factorization (PMF) [101].

However, it was observed in our experiments that DeepCoNN achieves its best performance only when the text of the review written by the target user for the target item is available at test time. In real world recommendation settings, an item is always recommended to a user **before** they have experienced it. Therefore, it would be unreasonable to assume that the target review would be available at the time of testing.

Let rev_{AB} denote the review written by $user_A$ for an $item_B$. At training time, the text corresponding to $user_A$, denoted as $text_A$, consists of a concatenation of all reviews written by her in the training set. Similarly, the text for $item_B$, denoted by $text_B$, is a concatenation of all reviews written for that item in the training set. Both $text_A$ and $text_B$ includes rev_{AB} for all $(user_A, item_B)$ pairs in the training set. At test time, there are two options for constructing the test inputs. For a test pair $(user_P, item_Q)$, their pairwise review, rev_{PQ} in the test set, could be included in the texts corresponding to the user, $text_P$, and the item, $text_Q$, or could be omitted. In one of our datasets, the

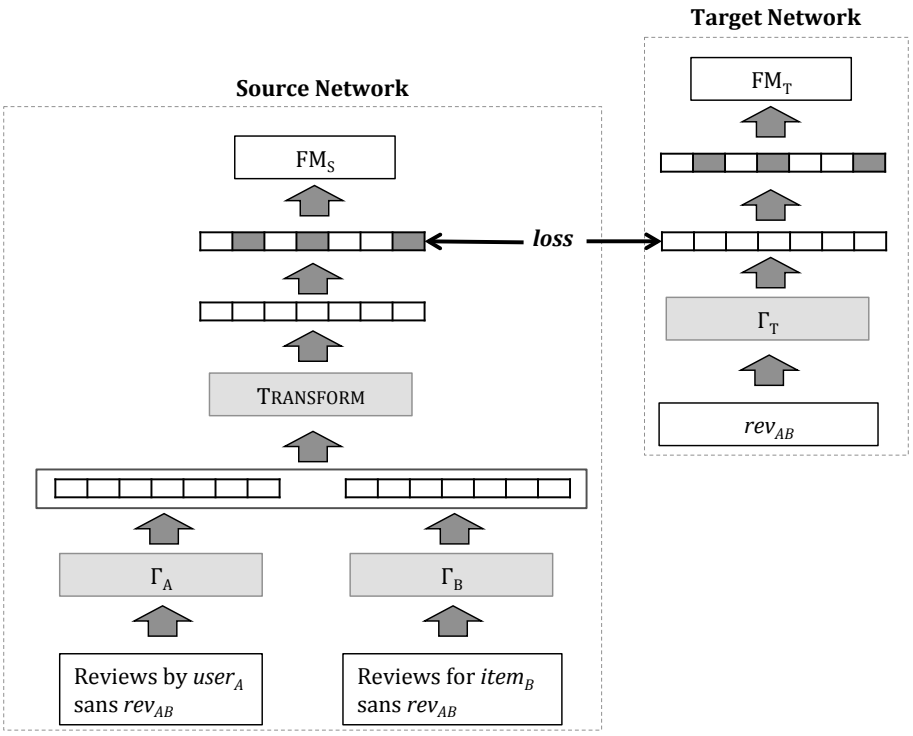

Figure 4.2: The TransNet architecture

MSE obtained by DeepCoNN if rev_{PQ} is included in the test inputs is only 1.21. However, if rev_{PQ} is omitted, then the performance degrades severely to 1.89. This is lower than Matrix Factorization applied to the same dataset, which has an MSE of 1.86. If we train DeepCoNN in the setting that mimics the test setup, by omitting rev_{AB} in the texts of all $(user_A, item_B)$ pairs in the training set, the performance is better at 1.70, but still much higher than when rev_{AB} is available in both training and testing.

In the setting used in this work, reviews in the validation and the test set are never accessed at any time, i.e., assumed to be unavailable — both during training and testing — simulating a real world situation.

4.2.4 TransNets

As we saw in the case of DeepCoNN, learning using the target review rev_{AB} at train time inadvertently makes the model dependent on the presence of such reviews at test time, which is unrealistic. However, as shown by the experiment above, rev_{AB} gives an insight into what $user_A$ thought about their experience with $item_B$, and can be an important predictor of the rating r_{AB}. Although unavailable at test time, rev_{AB} is available during training.

TransNet consists of two networks as shown in the architecture diagram of Figure 4.2, a *Target Network* that processes the target review rev_{AB} and a *Source Network* that processes the texts of the $(user_A, item_B)$ pair that does not include the joint review, rev_{AB}. Given a review text rev_{AB}, the Target Network uses a CNN Text Processor, Γ_T, and a Factorization Machine, FM_T, to predict the

rating as:

$$\begin{aligned} x_T &= \Gamma_T(rev_{AB}) \\ \bar{x}_T &= \delta(x_T) \\ \hat{r}_T &= FM_T(\bar{x}_T) \end{aligned}$$

Since the Target Network uses the actual review, its task is similar to sentiment analysis [60, 106].

The Source Network is like the DeepCoNN model with two CNN Text Processors, Γ_A for user text, $text_A - rev_{AB}$, and Γ_B for item text, $text_B - rev_{AB}$, and a Factorization Machine, FM_S, but with an additional Transform layer. The goal of the Transform layer is to **transform** the user and the item texts into an approximation of rev_{AB}, denoted by $\hat{rev}_{AB}$, which is then used for predicting the rating. The Source Network predicts the rating as given below:

First, it converts the input texts into their latent form as:

$$\begin{aligned} x_A &= \Gamma_A(text_A - rev_{AB}) \\ x_B &= \Gamma_B(text_B - rev_{AB}) \\ z_0 &= [x_A x_B] \end{aligned}$$

The last step above is a concatenation of the two latent representations. This is then input to the Transform sub-network, which is a L-layer deep non-linear transformational network. Each layer l in Transform has a weight matrix $G_l \in \mathbb{R}^{n \times n}$ and bias $g_l \in \mathbb{R}^n$, and transforms its input z_{l-1} as:

$$z_l = \sigma(z_{l-1} G_l + g_l)$$

where σ is a non-linear activation function. Since the input to the first layer, z_0, is a concatenation of two vectors each of n dimensions, the first layer of Transform uses a weight matrix $G_1 \in \mathbb{R}^{2n \times n}$.

The output of the L^{th} layer of Transform, z_L is the approximation constructed by the TransNet for rev_{AB}, denoted by $\hat{rev}_{AB}$. Note that we do not have to generate the surface form of rev_{AB}; It suffices to approximate $\Gamma_T(rev_{AB})$, the latent representation of the target review. The Source Network then uses this representation to predict the rating as:

$$\begin{aligned} \bar{z}_L &= \delta(z_L) \\ \hat{r}_S &= FM_S(\bar{z}_L) \end{aligned}$$

During training, we will force the Source Network's representation z_L to be similar to the encoding of rev_{AB} produced by the Target Network, as we discuss below.

4.2.5 Training TransNets

TransNet is trained using 3 sub-steps as shown in Algorithm 1. In the first sub-step, for each training example (or a batch of such examples), the parameters of the Target Network, denoted by θ_T, which includes those of Γ_T and FM_T, are updated to minimize a L_1 loss computed between the actual rating r_{AB} and the rating $\hat{r}_T$ predicted from the actual review text rev_{AB}.

To teach the Source Network how to generate an approximation of the latent representation of the original review rev_{AB} generated by the Target Network, in the second sub-step, its parameters,

denoted by θ_{trans}, are updated to minimize a L_2 loss computed between the transformed representation, $\bar{z}_L$, of the texts of the user and the item, and the representation x_T of the actual review. θ_{trans} includes the parameters of Γ_A and Γ_B, as well as the weights W_l and biases g_l in each of the transformation layers. θ_{trans} does not include the parameters of FM_S.

In the final sub-step, the remaining parameters of the Source Network, θ_S, which consists of the parameters of the FM_S are updated to minimize a L_1 loss computed between the actual rating r_{AB} and the rating $\hat{r}_S$ predicted from the transformed representation, $\bar{z}_L$. Note that each sub-step is repeated for each training example (or a batch of such examples), and not trained to convergence independently. The training method is detailed in Algorithm 1.

Algorithm 1 Training TransNet

procedure Train(D_{train})
  while not converged **do**
    for $(text_A, text_B, rev_{AB}, r_{AB}) \in D_{train}$ **do**
      #Step 1: Train Target Network on the actual review
      $x_T = \Gamma_T(rev_{AB})$
      $\hat{r}_T = FM_T(\delta(x_T))$
      $loss_T = |r_{AB} - \hat{r}_T|$
      update θ_T to minimize $loss_T$
      #Step 2: Learn to Transform
      $x_A = \Gamma_A(text_A)$
      $x_B = \Gamma_B(text_B)$
      $z_0 = [x_A x_B]$
      $z_L =$ Transform(z_0)
      $\bar{z}_L = \delta(z_L)$
      $loss_{trans} = ||\bar{z}_L - x_T||_2$
      update θ_{trans} to minimize $loss_{trans}$
      #Step 3: Train a predictor on the transformed input
      $\hat{r}_S = FM_S(\bar{z}_L)$
      $loss_S = |r_{AB} - \hat{r}_S|$
      update θ_S to minimize $loss_S$
  return θ_{trans}, θ_S

Algorithm 2 Transform the input

procedure Transform(z_0)
  for layer $l \in L$ **do**
    $z_l = \sigma(z_{l-1}G_l + g_l)$
  return z_L

At test time, TransNet uses only the Source Network to make the prediction as shown in Algorithm 3.

Algorithm 3 Testing using TransNet

procedure Test(D_{test})
  for $(text_P, text_Q) \in D_{test}$ **do**
    #Step 1: Transform the input
    $x_P = \Gamma_A(text_P)$
    $x_Q = \Gamma_B(text_Q)$
    $z_0 = [x_P x_Q]$
    $z_L = \text{Transform}(z_0)$
    $\bar{z}_L = \delta(z_L)$
    #Step 2: Predict using the transformed input
    $\hat{r}_{PQ} = FM_S(\bar{z}_L)$

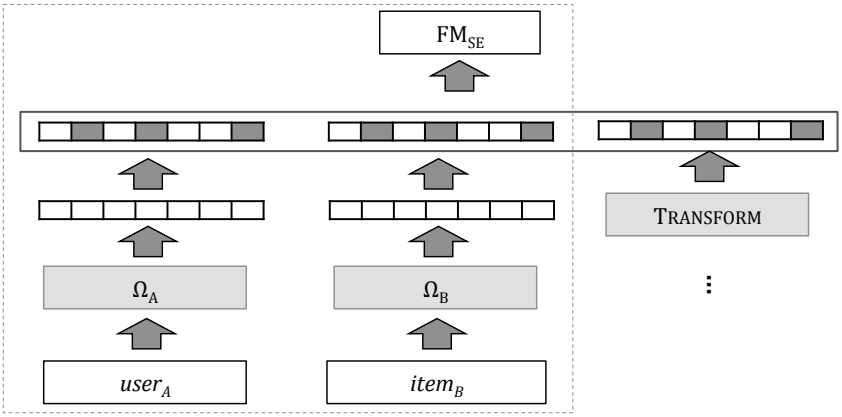

Figure 4.3: The Extended TransNet sub-architecture

4.2.6 Extended TransNets

TransNet uses only the text of the reviews and is user/item identity-agnostic, i.e., the user and the item are fully represented using the review texts, and their identities are not used in the model. However, in most real world settings, the identities of the users and items are known to the recommender system. In such a scenario, it is beneficial to learn a latent representation of the users and items, similar to Matrix Factorization methods. The *Extended TransNet* (TransNet-Ext) model achieves that by extending the architecture of TransNet as shown in Figure 4.3.

The Source Network now has two embedding matrices Ω_A for users and Ω_B for items, which are functions of the form, $\Omega : id \rightarrow \mathbb{R}^n$. These map the string representing the identity of $user_A$ and $item_B$ into a n-dimensional representation. These latent representations are then passed through a dropout layer and concatenated with the output of the Transform layer before being passed to the FM regression layer. Therefore, given $user_A$ and $item_B$, TransNet-Ext computes the rating as:

$$\begin{aligned} \omega_A &= \Omega(user_A) \\ \omega_B &= \Omega(item_B) \\ \bar{z} &= [\delta(\omega_A)\ \delta(\omega_B)\ \bar{z}_L] \\ \hat{r}_{SE} &= FM_{SE}(\bar{z}) \end{aligned}$$

Dataset	Category	#Users	#Items	#Ratings & Reviews
Yelp17		1,029,432	144,072	4,153,150
AZ-Elec	Electronics	4,200,520	475,910	7,820,765 (7,824,482)
AZ-CSJ	Clothing, Shoes and Jewelry	3,116,944	1,135,948	5,748,260 (5,748,920)
AZ-Mov	Movies and TV	2,088,428	200,915	4,606,671 (4,607,047)

Table 4.1: Dataset Statistics

Computation of the loss in *Step 3* of Algorithm 1, $loss_{SE}$ is same as earlier: $loss_{SE} = |r_{AB} - \hat{r}_{SE}|$. But the parameter θ_S updated at the end now contains the embedding matrices Ω_A and Ω_B.

4.3 Experiments and Results

4.3.1 Datasets

We evaluate the performance of our approach on four large datasets. The first one, Yelp17, is from the latest Yelp dataset challenge[1], containing about 4M reviews and ratings of businesses by about 1M users. The rest are three of the larger datasets in the latest release of Amazon reviews[2] [80, 81] containing reviews and ratings given by users for products purchased on amazon.com, over the period of May 1996 - July 2014. We use the aggressively de-duplicated version of the dataset and also discard entries where the review text is empty. The statistics of the datasets are given in Table 4.3.1. The original size of the dataset before discarding empty reviews is given in brackets when applicable.

4.3.2 Evaluation Procedure and Settings

Each dataset is split randomly into train, validation and test sets in the ratio $80 : 10 : 10$. After training on every 1000 batches of 500 training examples each, MSE is calculated on the validation and the test datasets. We report the MSE obtained on the test dataset when the MSE on the validation dataset was the lowest, similar to [78]. All algorithms, including the competitive baselines, were implemented in Python using TensorFlow[3], an open source software library for numerical computation, and were trained/tested on NVIDIA GeForce GTX TITAN X GPUs. Training TransNet on Yelp17 takes approximately 40 minutes for 1 epoch ($\sim$6600 batches) on 1 GPU, and gives the best performance in about 2–3 epochs.

4.3.3 Competitive Baselines

We compare our method against the current state-of-the-art, DeepCoNN [146]. DeepCoNN was previously evaluated against the then state-of-the-art models like Hidden Factors as Topics (HFT)

[1] https://www.yelp.com/dataset_challenge
[2] http://jmcauley.ucsd.edu/data/amazon
[3] https://www.tensorflow.org

model [78], Collaborative Topic Regression (CTR) [121], Collaborative Deep Learning (CDL) [122] and Probabilistic Matrix Factorization (PMF) [101], and shown to surpass their performance by a wide margin. We also consider some variations of DeepCoNN.

Our competitive baselines are:

1. **DeepCoNN**: The model proposed in [146]. During training, $text_A$ and $text_B$ corresponding to the $user_A$-$item_B$ pair contains their joint review rev_{AB}, along with reviews that $user_A$ wrote for other items and what other users wrote for $item_B$ in the training set. During testing, for a $user_P$-$item_Q$ pair, $text_P$ and $text_Q$ are constructed from only the training set and therefore, does not contain their joint review rev_{PQ}.
2. **DeepCoNN-rev$_{\mathbf{AB}}$**: The same DeepCoNN model (1) above, but trained in a setting that mimics the test setup, i.e., during training, $text_A$ and $text_B$ corresponding to the $user_A$-$item_B$ pair **does not contain** their joint review rev_{AB}, but only the reviews that $user_A$ wrote for other items and what other users wrote for $item_B$ in the training set. Testing procedure is the same as above: for a $user_P$-$item_Q$ pair, $text_P$ and $text_Q$ are constructed from only the training set and therefore, does not contain their joint review rev_{PQ} which is present in the test set.
3. **MF**: A neural net implementation of Matrix Factorization with $n = 50$ latent dimensions.

We also provide the performance numbers of DeepCoNN in the setting where the test reviews are available at the time of testing. i.e. the same DeepCoNN model (1) above, but with the exception that at test time, for a $user_P$-$item_Q$ pair, $text_P$ and $text_Q$ are constructed from the training set as well as the test set, and therefore, contains their joint review rev_{PQ} from the test set. This is denoted as **DeepCoNN + Test Reviews**, and its performance is provided for the sole purpose of illustrating how much better the algorithm could perform, had it been given access to the test reviews.

4.3.4 Evaluation on Rating Prediction

Like prior work, we use the Mean Square Error (MSE) metric to evaluate the performance of the algorithms. Let N be the total number of datapoints being tested. Then MSE is defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(r_i - \hat{r}_i)^2$$

where, r_i is the ground truth rating and $\hat{r}_i$ is the predicted rating for the i^{th} datapoint. Lower MSE indicates better performance.

The MSE values of the various competitive baselines are given in Table 4.2. For each dataset, the best score is highlighted in blue.

As can be seen from the Table, it is clear that TransNet and its variant TransNet-Ext perform better at rating prediction compared to the competitive baselines on all the datasets (*p-value* ≤ 0.05). It can also be seen that learning a user and item embedding using only the ratings in addition to the text helps TransNet-Ext improve the performance over the vanilla TransNet (*p-value* ≤ 0.1), except in the case of one dataset (`AZ-CSJ`).

It is also interesting to note that training DeepCoNN mimicking the test setup (DeepCoNN-rev$_{AB}$) gives a large improvement in the case of `Yelp`, but does not help in the case of the `AZ` datasets.

Table 4.2: Performance comparison using MSE metric

Dataset	DeepCoNN + Test Reviews	**MF**	**DeepCoNN**	**DeepCoNN-rev$_{AB}$**	**TransNet**	**TransNet-Ext**
Yelp17	1.2106	1.8661	1.8984	1.7045	1.6387	1.5913
AZ-Elec	0.9791	1.8898	1.9704	2.0774	1.8380	1.7781
AZ-CSJ	0.7747	1.5212	1.5487	1.7044	1.4487	1.4780
AZ-Mov	0.9392	1.4324	1.3611	1.5276	1.3599	1.2691

4.4 Contributions

Using reviews for improving recommender systems is an important task and is gaining a lot of attention in the recent years. A recent neural net model, *DeepCoNN*, uses the text of the reviews written by the user and for the item to learn their latent representations, which are then fed into a regression layer for rating prediction. However, its performance is dependent on having access to the user-item pairwise review, which is unavailable in real-world settings.

We presented a new model called *TransNets* which extends *DeepCoNN* with an additional Transform layer. This additional layer learns to transform the latent representations of user and item into that of their pair-wise review so that at test time, an approximate representation of the target review can be generated and used for making the predictions. We also showed how *TransNets* can be extended to learn user and item representations from ratings only which can be used in addition to the generated review representation. Our experiments showed that *TransNets* and its extended version can improve the state-of-the-art substantially.

Chapter 5

Preliminary Work: User Review Prediction using TransNets

Our evaluation of TransNets in Chapter 4 was quantitative, using MSE of predicted ratings. We would also like to investigate whether the learned representation is qualitatively useful—i.e., does it capture interesting high-level properties of the user's review. One possible use of the learning representation would be to give the user information about her predicted reaction to the item that is more detailed than a rating. In this section, we show how TransNets could be used to find the most similar reviews, personalized to each user. For example, the most similar review for a user who is more concerned about the quality of service and wait times would be different from the most similar review for another user who is sensitive to the price. For a test $user_P - item_Q$ pair, we run the Source Network with the text of their reviews from the training set to construct z_L, which is an approximation for the representation of their actual joint review. Candidate reviews are all the reviews rev_{CQ} in the training set written for $item_Q$ by other users. We pass each of them separately through the Target Network to obtain their latent representation $x_{CQ} = \Gamma_T(rev_{CQ})$. If rev_{CQ} had been most similar to what $user_P$ would write for $item_Q$, then x_{CQ} would be most similar to z_L. Therefore, to find the most similar review, we simply choose the rev_{CQ} whose x_{CQ} is closest to z_L in Euclidean distance.

Some examples of such predicted most similar reviews on the `Yelp17` dataset are listed in Table 5.1. Here, the column Original Review is the actual review that $user_P$ wrote for $item_Q$, and the column Predicted Review gives the most similar of the candidate reviews predicted by TransNet. The examples show how the predicted reviews talk about particulars that the original reviews also emphasize. These are highlighted in the Table (highlighting was done manually).

Our model of explanation seeks to show why a user might like an item in terms of what aspects of that item the user would like as well as dislike. Reviews are examples of explanations of this sort – they discuss why a particular rating was given. So, it makes sense that the latent representation for the review would have information relevant to the explanation. One of the ways to test if the approximate latent representation computed by TransNet contains the relevant information is by checking if reviews with similar latent representations offer similar explanations. The evaluation of review similarity in this chapter is purely qualitative. However, this method will serve as a baseline for the work proposed in Chapter 6 and will be evaluated thoroughly, as described there.

Original Review	Predicted Review
my laptop flat lined and i did n't know why , just one day it did n't turn on . i cam here based on the yelp reviews and happy i did . although my laptop could n't be revived due to the fried motherboard , they did give me a full explanation about what they found and my best options . i was grateful they did n't charge me for looking into the problem , other places would have . i will definitely be coming back if needed . .	my hard drive crashed and i had to buy a new computer . the store where i bought my computer could n't get data off my old hard drive . neither could a tech friend of mine . works could ! they did n't charge for the diagnosis and only charged $ 100 for the transfer . very happy .
this is my favorite place to eat in south charlotte . great cajun food . my favorite is the fried oysters with cuke salad and their awesome mac 'n' cheese (their mac 'n' cheese is not out of a box) . their sweet tea would make my southern grandma proud . to avoid crowds , for lunch i recommend arriving before 11:30 a.m. or after 1 p.m. and for dinner try to get there before 6 p.m. is not open on sundays .	always !! too small location so wait line can be long . been going to for 13 years .
this place is so cool . the outdoor area is n't as big as the fillmore location , but they make up for it with live music . i really like the atmosphere and the food is pretty spot on . the sweet potato fry dip is really something special . the vig was highly recommended to me , and i 'm passing that recommendation on to all who read this .	like going on monday 's . happy hour for drinks and apps then at 6pm their burger special . sundays are cool too , when they have live music on their patio .
i have attempted at coming here before but i have never been able to make it in because it 's always so packed with people wanting to eat . i finally came here at a good time around 6ish ... and not packed but by the time i left , it was packed ! the miso ramen was delicious . you can choose from add on 's on your soup but they charge you , i dont think they should , they should just treat them as condiments . at other ramen places that i have been too i get the egg , bamboo shoot , fire ball add on 's free . so i am not sure what their deal is .	hands down top three ramen spots on the west coast , right up there with , and the line can be just as long .
this place can be a zoo !! however , with the produce they have , at the prices they sell it at , it is worth the hassle . be prepared to be pushed and shoved . this is much the same as in asia . my wife (from vietnam) says that the markets in asia are even more crowded . i agree as i have seen vietnam with my own eyes .	i enjoy going to this market on main street when i am ready to can ... the prices are great esp for onions . . broccoli and bell peppers ... a few times they have had bananas for $ 3.00 for a huge box like 30 lbs ... you can freeze them or cover in ... or make banana bread if they begin to go dark ... and ripe . the employees will talk if you say hello first ...
great spot for outdoor seating in the summer since it 's sheltered early from the sun . good service but americanos sometimes are not made right	this is my " go to " starbucks due to the location being close to where i live . i normally go through the drive-thru , which more likely than not , has a long line . . but does n't every starbucks ? i have always received great customer service at this location ! there has been a couple times that my order was n't correct - which is frustrating when you are short on time & depend on your morning coffee ! but overall you should have a good experience whether you drive-thru or dine in !

Table 5.1: Example of predicted similar reviews

Original Review	Predicted Review
excellent quality korean restaurant . it 's a tiny place but never too busy , and quite possibly the best korean dumplings i 've had to date .	for those who live near by islington station you must visit this new korean restaurant that just opened up . the food too good to explain . i will just say i havent had a chance to take picture since the food was too grat .
very overpriced food , very substandard food . wait staff is a joke . if you like being ignored in the front room of a one story house and being charged for it , by all means . otherwise , go to freaking mcdonald 's .	i want this place to be good but it fails me every time . i brought my parents here for dinner and was totally embarrassed with my dining choice . i tried it two more times after that and continue to be disappointed . their menu looks great but what is delivered is a total let down . to top it off , the service is painfully slow . the only thing this place has going for it is the dog friendly patio and craft beers . i hope someone reads these reviews as the poor service piece continues to be brought up as an issue .
ok the first time i came here , i was very disappointed in the selection of items , especially after reading previous review . but , then i realized that i went at a bad time , it was the end of the day and they sold out of everything ! i recently went back at the store opening time and a lot happier with the market . they sell freshly made bentos , made in house , and they are perfect for microwaving at home or in the market for a cheap and satisfying meal . the key is to get there early , bc they are limited and run out quick , but they have a good variety of bentos . one draw back is that it is smaller than expected , so if you come from a place like socal , where japanese markets were like large grocery stores with mini stores and restaurants located inside , you might not be too happy .	the main reason i go here is for the bento boxes -LRB- see example pic -RRB- . made fresh every day , and when they 're gone , they 're gone . on my way home from work it 's a toss up whether there will be any left when i get there at 5:30 . i would by no means call them spectacular , but they 're good enough that i stop in every weeks i like to pick up some of the nori maki as well -LRB- see pic -RRB- one thing i wish they had more often is the spam and egg onigiri -LRB- see pic -RRB- . very cool . i 'm told you can order them in advance , so may have to do that
holey moley - these bagels are delicious ! i 'm a bit of a bagel connoisseur . (note : the bagels at dave 's grocery in ohio city are currently my favs) . these bagels had me floored . thankfully , cleveland bagel pops up at festivals and flea markets so there are plenty of opportunities to put them in your mouth (though rising star coffee is a regular option) . their are also amazing ! though they are n't the cheapest bagels in the world , you can taste the love that goes into them . they 're perfectly crisp , yet doughy in the middle . the add an added flavor - honestly , it 's a bagel experience .	i had heard from a colleague at work about cleveland bagel company 's bagels and how they were , " better than new york city bagels . " naturally , i laughed at this colleague and thought he was a for even thinking such a thing . so , a few weeks later i happened to be up early on a saturday morning and made the trek up to their storefront -(located across from the harp .) when i arrived was around 8:15 am ; upon walking in i found most bagel bins to be empty and only a few poppyseed bagels left . i do n't like poppyseed bagels so i asked them what was going on with the rest and when they 'd have more . to my surprise i found out that they only stay open as long as they have bagels to sell . once they sell out , they close up shop and get going for the next day . i ordered a poppyseed bagel even though i do n't like them as i was curious as to what was up with these bagels and can tell you that they are in fact better than new york city bagels . i ca n't even believe i 'm saying that , but it 's true . you all need to do what you can to get over there to get some of these bagels . they 're unbelievable . i ca n't explain with words exactly why they 're so amazing , but trust me , you will love yourself for eating these bagels . coffee is n't that great , but it does n't matter . get these bagels ?!

Table 5.2: Example of predicted similar reviews (continued from previous page)

Chapter 6

Proposed Work: User Review Generation

In this chapter, we look more closely into how a textual explanation could be 'generated'. In the previous chapter, the textual explanation was selected from existing reviews written by other users. Each user's overall experience with the item is bound to be unique even if they share common opinions about certain aspects of the item with other users. For example, consider the set of reviews for the movie 'Inferno' by users Alice and Kumar in Figure 6.1. Alice who likes the actor Tom Hanks and good movie plots, talks about those aspects in her review. That review echoes some part of other user's reviews for the same movie, like that by Bob who disapproves of the plot and that by UserC who is all praise for the Hanks' character. But Alice's review is not exactly the same as any one of these other reviews, but a combination of different aspects from multiple reviews. Similar is the case of Kumar who loves action sequences and film scores. His review has elements from UserD's review but also additional aspects not mentioned by UserD.

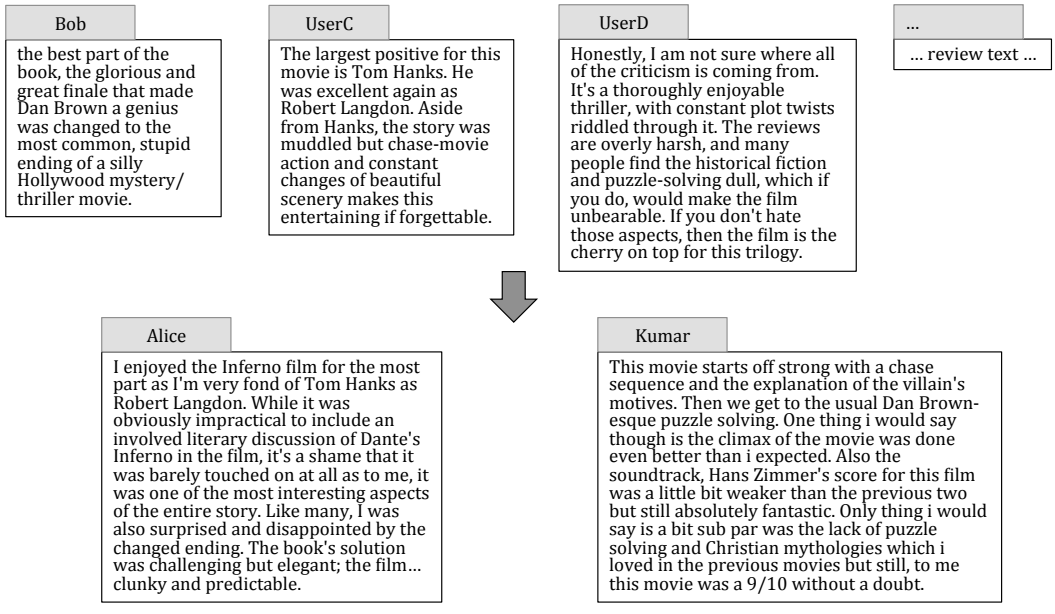

Figure 6.1: Sample Movie Reviews

The above example is a typical real world scenario where different users care about different aspects of the same item. An ideal explanation of why they would like or dislike the item typically cannot be extracted from any one review, but needs to be synthesized from multiple reviews.

6.1 Related Work: Explanation using Reviews

6.1.1 Non-Personalized Summarization

A multitude of models have been proposed in the past that can create a generic summary of reviews. [100] is a recent method for abstractive text summarization using a neural language model with a beam-search decoder. However, the task is limited to generating a headline from the first sentence of a news article, compared to reviews that typically include multiple long sentences. A similar work is the [123], which uses an attention based neural network to generate one-sentence summaries from multiple opinions. In [40], the authors incorporate a copy mechanism to a `seq2seq` learning model to improve text summarization task. [37] is an abstractive graph-based technique that constructs summaries from highly redundant opinions.

[90] is an extractive method that selects sentences from multiple reviews using measures of informativeness and readability, to construct a summary review. There has been a number of recent work that proposed how neural models could be used for extractive summarization. For example, [89] uses side information to construct news headlines from the text of the article. [24] is another model for the same task, but uses a hierarchical reader and a sentence level attention extractor for selecting informative sentences. [107] is a recently proposed extractive method that uses the principles of sparse-coding and Maximal Marginal relevance (MMR) to select the sentences to be included in the summary. The RNN-based method proposed in [88] has been shown to be comparable or better than the state-of-the-art algorithms for extractive summarization. [138] is a graph based neural model for the same task.

[12] learns an LSTM model that can generate spurious reviews for restaurants. This character level model only generates review-like text and has no notion of user or item nor is it summarizing anything. An almost identical method was also proposed in [137]. The reviews generated by both these models were evaluated by human judges and found to be indistinguishable from authentic reviews.

6.1.2 Extractive / Non-Neural Models for personalized explanations

[97] proposed an extractive review generation by selecting sentences that are most similar to the user's profile but diverse from the already selected sentences. This greedy approach is same as the MMR criteria proposed for search engines [18]. [145] proposed an Explicit Factor Model for generating explanations using phrases extracted from reviews. They identify the sentiment that each user has about the aspects of the item and use them to provide discrete (not in fully formed natural language sentences) explanations. [44] is another method that ranks aspects for each user on a tripartite graph consisting of the users, items and aspects. In [129], the authors proposed a joint Bayesian model for rating and review prediction. Although they learn a generative model for the review text, they evaluate it using only perplexity and do not explicitly generate the text. Similarly, in [32], the

authors proposed a user specific aspect sentiment model to predict ratings where they evaluated the reviews using only perplexity. Although [79] models the sentiment associated with each of the aspects of the item from the reviews, they do not provide any explanations to the user while predicting ratings for new items. [86] jointly models reviews and ratings using a Hidden Markov Model – Latent Dirichlet Allocation (HMM–LDA) to provide interpretable explanations using words from latent word clusters that would explain important aspects of the item.

6.1.3 Abstractive / Neural Models for personalized explanations

[71] proposed a character-level model called Generative Concatenative Net (GCN), which is a regular character-level LSTM with additional information concatenated to its input. The additional information includes that of the user, item, overall rating and the category of the item. [26] is a minor extension to [71], where ratings of individual aspects of the item are also concatenated to the auxiliary information. [113] is a similar work that produces the text of the review from the item id and sentiment/rating as input. [127] applies Dynamic Memory Networks (DMN) to generate a personalized latent representation of the joint review. First, it uses LSTMs to separately process reviews for the product and those written by the user for other products to generate latent representations for the product and the user. Then, the product representation is iteratively refined using a DMN with the user representation driving the attention model, to construct a representation of the product that is personalized to the user. That is then passed through an LSTM to generate the words of the review. Although the experiments were conducted on very small datasets, the authors do show that the method is capable of producing reviews with better ROUGE scores than competitive summarization algorithms. [64] is another recent model that generates tips instead of explanations. Their model jointly optimizes prediction of ratings along with the generation of a short one sentence suggestion like for example "You need to reserve a table in advance". The tip generation process is quite similar to the source network of the model proposed in this chapter.

6.1.4 Comparison to Related Tasks

Sections of the model proposed in this Chapter bare resemblance to models that have been proposed for different but related tasks such as text generation from structured inputs and caption generation.

Text Generation from Structured Data

In [61], the authors proposed a model for generating one-sentence biographies from data structured in the form of tables like the Infobox of Wikipedia. The main improvement compared to earlier models was to produce an identifier corresponding to the location of the desired data in the table, which could then be copied into the generated text. [104] proposed an improvement of this model which also considers the order of facts in the table. [4] is another method for a similar task which combines information from a knowledge graph of facts with a neural language model while generating text. [42] is a simpler model which poses this as a `seq2seq` problem by flattening the table to a sequence of words. The model proposed in [84], is similar, except they introduce an additional aligning step before the decoding stage.

Caption Generation

In the model proposed in this Chapter, the input to the decoder that generated the natural language text could be from a CNN, in which case, the sequence information present in the input text will not be preserved. This setting is similar to the popular task of caption generation for images. [119, 120] embed the image using a deep vision CNN which is input to a language RNN that produces the caption. This was extended with attention to improve the text generation in [133]. This is further improved by using a sentinel for non-visual words in [74]. In [72], the authors train the model to explicitly learn attention maps from alignment information. In [136], there is an additional review network that inputs the output of an encoder to produce multiple thought vectors using attention. These are then passed as input to the decoder.

Autoencoder

In one of the networks of the proposed model, the input and the output are the same, making it an Autoencoder. In [63], the authors proposed a hierarchical LSTM that could encode and decode a long paragraph showing that the model could preserve syntactic, semantic and discourse coherence. A recent work [36] proposed a similar model which encodes using a CNN and decodes using a hierarchical LSTM. An earlier model proposed in [28] auto-encoded sentences.

6.2 Paradigms and Models for Improving Review Generation

In this section, we consider some of the existing techniques that have been used for other (similar) tasks previously with success, and which could be used to enhance review generation.

6.2.1 Extractive vs. Abstractive Text

In Section 6.1.2, we discussed techniques proposed in the past that extracted sentences or phrases from different reviews and stitched together, an explanation for the target user. While such an ad-hoc explanation may suffice for the purpose, they are known to suffer from poor readability.

With the recent advances in the field of neural networks, it has been shown that generating text that is both grammatical and readable is now feasible. e.g. [71]. Therefore, we propose to generate reviews in an abstractive manner.

6.2.2 Character and Word Level Models

A few attempts have been already made to generate personalized reviews that can serve as an explanation. Two of the models proposed in the past used character-level models [26, 71] and a more recent approach used word-level models [127]. Learning at character-level is helpful in scaling to large vocabulary because at each step of the prediction, the model is only predicting which of the characters will occur next, which is several orders of magnitude smaller than the typical vocabulary sizes in large corpora. In contrast, word-level models are known to have trouble with increasing vocabulary sizes. However, a character-level model will most likely need to learn more model parameters since it has to also learn how to combine characters together to form meaningful words,

before learning to order the words to form meaningful sentences. i.e. both models have different strengths and weaknesses, and it is not obvious which one would perform better. In [134], the authors proposed a model that could combine word and character level representations dynamically for a reading comprehension task. An earlier model proposed in [135] concatenated the two representations for a sequence tagging task.

6.2.3 Beam Search

In the field of text generation, summarization where the focus is not on personalizing the generated text, performance improvements have been reported by using a beam-search style decoding mechanism [25, 100]. In a beam search decoder, the system maintains a set of K most probable output at each step in the decoding. Such a mechanism can perform better than a greedy decoding mechanism which selects the next most probable word at each step.

6.2.4 GAN style training

Generative Adversarial Networks (GAN) [39, 98] are a recent development in deep learning for training generative models, by an adversarial process. This framework has two sub networks – a generative network that is responsible for generating the output, and an adversarial network which tries to differentiate if a sample is an original one from the training data or a synthetic one from the generative network. The generative network is trained to maximize the chances of the adversarial network making a mistake. Such a framework has been shown to give good performance gains in the computer vision community [23, 39, 102].

A recent work, [66], used it for visual paragraph generation. To improve the quality of the generated paragraph, [66] employs two discriminators – one for measuring the plausibility of each of the sentences, and the other for measuring the smoothness of transition of topic between the sentences. They produce sentences that are semantically relevant and grammatically correct, even though they do not use a beam search decoder.

6.3 Proposed Work

I propose to study different character and word-level models, with and without beam search optimization and an adversarial training, for the purposes of review generation and compare their performances, both automatically and using human judgements. The proposed work is detailed in the below subsections.

6.3.1 Models to be compared

I plan to implement (or reuse existing implementations of) the following models for comparing their relative performances:

1. TransNets for selecting the most similar review (described in Chapter 5)
2. The character level model proposed in [71]
3. The extended character level model proposed in [26]

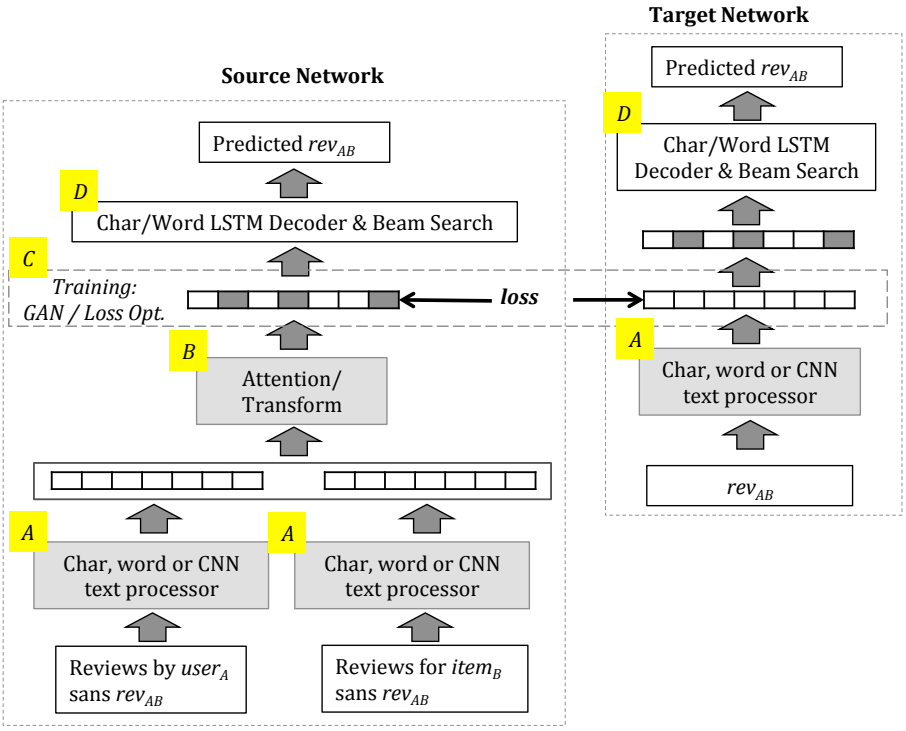

Figure 6.2: Proposed architecture

4. The word level model proposed in [127]
5. Character-level models (2 & 3) with a beam-search decoder
6. Character-level models (2 & 3) with a GAN-style training
7. Word-level model (4) with a beam-search decoder
8. Word-level model (4) with a GAN-style training

The proposed architecture is shown in Figure 6.2. The above variants correspond to changing one or more of the modules A, B, C or D in the figure. A corresponds to the module that converts a piece of text into its latent representation. Following the prior work, we could use a Character or Word level LSTM. It could also be the CNN text processor detailed in Section 4.2.1. B is the module that converts the two latent representations into the one that corresponds to the latent representation of their joint review. It could be modeled using an attention mechanism as shown in [127] or using the Transform as in the TransNets. C corresponds to the training process for learning how to generate the approximate latent representation. It could either follow the process used in TransNets by minimizing the L_1 or L_2 distance between the two latent vectors, or it could use an adversarial training mechanism that has been very popular for training a number of different powerful neural models. D corresponds to the decoder module that inputs a latent representation and any side information where applicable, and produces the actual text of the review. This could be implemented using a plain LSTM that uses either a character or a word level decoder. However, adding a beam search could possibly improve its performance.

Dataset	Category	#Users	#Items	#Ratings & Reviews
Yelp17		1,029,432	144,072	4,153,150
AZ-Elec	Electronics	4,200,520	475,910	7,820,765 (7,824,482)
AZ-CSJ	Clothing, Shoes and Jewelry	3,116,944	1,135,948	5,748,260 (5,748,920)
AZ-Mov	Movies and TV	2,088,428	200,915	4,606,671 (4,607,047)
RateBeer	Rate Beer	40,213	110,419	2,924,127
BeerAdv	Beer Advocate	33,387	66,051	1,586,259

Table 6.1: Dataset Statistics

6.3.2 Datasets

We will evaluate the performance of the models on six large datasets. The first one, Yelp17, is from the latest Yelp dataset challenge[1], containing about 4M reviews and ratings of businesses by about 1M users. The next three are three of the larger datasets in the latest release of Amazon reviews[2] [80, 81] containing reviews and ratings given by users for products purchased on amazon.com. The next two are beer rating datasets [79, 82]. The statistics of the datasets are given in Table 6.3.2.

The reviews in BeerAdv and RateBeer are relatively well formed in the sense that the reviews show more structure. Since the reviews focus entirely on the aspects of the beer, the chance for variance is reduced. For example, in BeerAdv, most reviews discuss the following aspects of the beer in order: appearance, smell, taste, mouthfeel and drinkability, although each user has their own way of describing the beer [71]. A sample review is below:

Poured from 12oz bottle into half-liter Pilsner Urquell branded pilsner glass. Appearance: Pours a cloudy golden-orange color with a small, quickly dissipating white head that leaves a bit of lace behind. Smell: Smells HEAVILY of citrus. By heavily, I mean that this smells like kitchen cleaner with added wheat. Taste: Tastes heavily of citrus- lemon, lime, and orange with a hint of wheat at the end. Mouthfeel: Thin, with a bit too much carbonation. Refreshing. Drinkability: If I wanted lemonade, then I would have bought that.

The markers "Appearance", "Smell", etc. are not always used. Some users abbreviate it as "A", "S", etc. and others omit them entirely. In these two datasets, the users also provide separate ratings for these aspects along with their overall rating.

Compared to the beer reviews, reviews in Yelp17 are completely free form text as users describe their unique experience with the item. The reviews do not always focus on only the aspects of the item, but also the circumstances leading to the user trying the item and many of them tend to ramble on. For example, a sample review is given below:

*I had heard from a colleague at work about Cleveland bagel company's bagels and how they were, "better than new york city bagels." Naturally, i laughed at this colleague and thought he was a ** for even thinking such a thing. So, a few weeks later I happened to be up early on a saturday morning and made the trek up to their storefront -(located across from the harp.) When i arrived was around 8:15 am; upon walking in I found most bagel bins to be empty and only a few poppyseed bagels left. i don't like poppyseed bagels so i asked them what was going on with the rest and when they'd have more. To my surprise I found out that they*

[1] https://www.yelp.com/dataset_challenge
[2] http://jmcauley.ucsd.edu/data/amazon

only stay open as long as they have bagels to sell. Once they sell out, they close up shop and get going for the next day. I ordered a poppyseed bagel even though I don't like them as I was curious as to what was up with these bagels and can tell you that they are in fact better than new york city bagels. I can't even believe I'm saying that, but it's true. You all need to do what you can to get over there to get some of these bagels. They're unbelievable. I can't explain with words exactly why they 're so amazing, but trust me, you will love yourself for eating these bagels. Coffee isn't that great, but it doesn't matter. Get these bagels ?!

The Amazon reviews are similar to the Yelp reviews, with no discernible structure. Therefore, we expect most of the models to be better at generating the beer reviews than those from the Yelp and Amazon datasets.

6.3.3 Automatic Evaluation

To measure the performance of the models, we will evaluate the generated text to the original review written by the target user for the target item, using the following metrics:

1. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [67] is a recall oriented metric for evaluating automatically generated summaries with human generated ones and was used by some of the previous work to evaluate their approaches. The commonly used version of this metric is the ROUGE-N, which computes N-gram overlap between the system and reference summaries. N is typically 1,2 or 3. This will help us determine how closely the generated text resembles the original. It also accounts for the different word usage of different users.
2. BLEU (BiLingual Evaluation Understudy) [94] is a metric used for evaluating the quality of machine-translated text. It uses a modified form of precision computed between the system generated text and the reference.
3. METEOR (Metric for Evaluation of Translation with Explicit ORdering) [31] was proposed to overcome some of the limitations of BLEU. It uses a harmonic mean of precision and recall to calculate the score, and was found to correlate highly with human judgements.
4. MSE (Mean Squared Error) using a classifier (rating prediction / regression) trained on the original reviews for rating prediction, applied to the generated summaries. This will help us determine if the overall sentiment conveyed by the generated text is the same as the original.
5. Perplexity: The negative log likelihood of the generated text under a language model trained on the original reviews. This will help us determine if the word sequence in the generated text follows a similar grammar as the original corpus. Although we could use a generic English language grammar checker, we should note that the generated text can be only as good as the corpus used to train the model. Like we saw in the examples, the training corpora is not exactly grammatically correct.
6. Predictability of aspect ratings: In the beer datasets, `RateBeer` and `BeerAdv`, in addition to the overall ratings, users also provide individual ratings for the different aspects of the item. That makes it possible to test if the generated text contains the same sentiment as the original, for each of the aspects, in these two datasets. We will train separate classifiers for predicting each of the aspect ratings from the original reviews. These classifiers can then be applied on the generated text.

6.3.4 Human Judgements

In certain tasks, it has been noted that a good performance on intrinsic metrics does not necessarily correlate with human ratings. Therefore, it is equally important to test the models' performances using external evaluation methods.

We propose a Mechanical Turk setup for evaluating the generated text. For each test item, the turker is shown the original user's rating and review for the item. Along side those, the generated text from a pair of the test systems is shown in a random order. The turker is asked to rate the generated text according to their similarity to the actual review and how closely they convey the overall sentiment as indicated by the actual rating given by the user. Such an evaluation screen is shown in Figure 6.3. The average scores by multiple MTurkers will help us compare the performance of the different algorithms with respect to how humans evaluate their performance.

6.4 Timeline

- Knowledge Graph based methods
  - Improving rating prediction - Published at RecSys 2016
  - Explanation generation - Published at RecSys 2017 (joint work with Kathryn Mazaitis)
- Neural Network based methods
  - Improving rating prediction - Published at RecSys 2017
  - Explanation generation:
    - Preliminary Experiments - Ongoing work - Fall 2017
    - MTurk pipeline - Fall 2017
  - Evaluation of Explanations - Spring 2018
- Thesis writing and defense - Summer 2018

Original Review	Test Review A	Test Review B
I enjoyed the Inferno film for the most part as I'm very fond of Tom Hanks as Robert Langdon. While it was obviously impractical to include an involved literary discussion of Dante's Inferno in the film, it's a shame that it was barely touched on at all as to me, it was one of the most interesting aspects of the entire story. Like many, I was also surprised and disappointed by the changed ending. The book's solution was challenging but elegant; the film... clunky and predictable.	The largest positive for this movie is Tom Hanks. He was excellent again as Robert Langdon. Aside from Hanks, the story was muddled but chase-movie action and constant changes of beautiful scenery makes this entertaining if forgettable. The best part of the book, the glorious and great finale that made Dan Brown a genius was changed to the most common, stupid ending of a silly Hollywood mystery/ thriller movie.	This movie starts off strong with a chase sequence and the explanation of the villain's motives. Then we get to the usual Dan Brown- esque puzzle solving. One thing i would say though is the climax of the movie was done even better than i expected. Only thing i would say is a bit sub par was the lack of puzzle solving which i loved in the previous movies but still, to me this movie was a 9/10 without a doubt.
This user gave the movie 4 out of 5 stars		

Which of the Test Reviews matches the Original Review better?			
O Both A & B	O Only A	O Only B	O Neither
Which of the Test Reviews conveys the same sentiment as the Original Review ?			
O Both A & B	O Only A	O Only B	O Neither
Which of the Test Reviews emphasizes the same likeable aspect of the item as the Original Review?			
O Both A & B	O Only A	O Only B	O Neither
Which of the Test Reviews emphasizes the same unlikeable aspect of the item as the Original Review?			
O Both A & B	O Only A	O Only B	O Neither

The English language and style of which of the Test Reviews resemble that of a human ?			
O Both A & B	O Only A	O Only B	O Neither
The English language and style of which of the Test Reviews resemble that of the Original Review ?			
O Both A & B	O Only A	O Only B	O Neither
Which of the Test Reviews is coherent ?			
O Both A & B	O Only A	O Only B	O Neither
Which of the Test Reviews is readable ?			
O Both A & B	O Only A	O Only B	O Neither

Figure 6.3: Sample evaluation screen for an Amazon M Turker

Bibliography

[1] Behnoush Abdollahi and Olfa Nasraoui. Explainable restricted boltzmann machines for collaborative filtering. *CoRR*, abs/1606.07129, 2016. URL `http://arxiv.org/abs/1606.07129`. 3.1

[2] Behnoush Abdollahi and Olfa Nasraoui. Using explainability for constrained matrix factorization. In *Proc. Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 79–83, 2017. 3.1

[3] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 335–336, 2008. 4

[4] Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. A neural knowledge language model. *CoRR*, abs/1608.00318, 2016. URL `http://arxiv.org/abs/1608.00318`. 6.1.4

[5] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 147–154, 2015. 4, 4.1.2

[6] Authors anonymized. Inmind movie agent - a platform for research. Under Review, 2017. 3.3

[7] Lukasz Augustyniak, Krzysztof Rajda, and Tomasz Kajdanowicz. Method for aspect-based sentiment annotation using rhetorical analysis. In *Intelligent Information and Database Systems - 9th Asian Conference, ACIIDS*, pages 772–781, 2017. 4.1.3

[8] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. arXiv:1505.04406 [cs.LG], 2015. 2.2, 3.1

[9] Lars Backstrom and Jure Leskovec. Supervised random walks: Predicting and recommending links in social networks. In *Proc. WSDM '11*, pages 635–644, 2011. 2.3.1, 2.3.2, 2.3.2

[10] Trapit Bansal, David Belanger, and Andrew McCallum. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 107–114, 2016. 4, 4.1.2

[11] Yang Bao, Hui Fang, and Jie Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 2–8, 2014. 4, 4.1.1

[12] A. Bartoli, A. d. Lorenzo, E. Medvet, D. Morello, and F. Tarlao. "best dinner ever!!! ": Automatic generation of restaurant reviews with lstm-rnn. In *2016 IEEE/WIC/ACM International*

Conference on Web Intelligence (WI), pages 721–724, 2016. 6.1.1

[13] Kirell Benzi, Vassilis Kalofolias, Xavier Bresson, and Pierre Vandergheynst. Song recommendation with non-negative matrix factorization and graph total variation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 2439–2443, 2016. 2.2

[14] Mustafa Bilgic and Raymond J. Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization Workshop*, January 2005. 3.1

[15] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. 4.1.1

[16] Matthew Brand. A random walks perspective on maximizing satisfaction and profit. In *Proc. SDM*, 2005. 2.3.1

[17] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 535–541, 2006. 4.1.3

[18] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998. 6.1.2

[19] R. Catherine and W. Cohen. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *Proc. RecSys '16*, pages 325–332, 2016. 2, 3.1, 3.2

[20] R. Catherine and W. Cohen. Transnets: Learning to transform for recommendation. In *Proc. RecSys '17*, 2017. 4, 4.2.1

[21] R. Catherine, K. Mazaitis, M. Eskenazi, and W. Cohen. Explainable entity-based recommendations with knowledge graphs. In *Proc. RecSys '17 Posters*, 2017. 3

[22] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1627–1634, 2012. 4.1.2

[23] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS 2016*, pages 2172–2180, 2016. 6.2.4

[24] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. In *Proc. 54th Annual Meeting of the Association for Computational Linguistics, ACL*, 2016. 6.1.1

[25] Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL HLT 2016*, pages 93–98, 2016. 6.2.3

[26] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. Automatic generation of natural language explanations. 2017. 6.1.3, 6.2.2, 3

[27] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 191–198, 2016. 4.1.2

[28] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In C. Cortes, N. D.

Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3079–3087. 2015. 6.1.4

[29] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proc. WWW '07*, pages 271–280, 2007. 2.2.1

[30] Luis de Campos, Juan Fernández-Luna, Juan Huete, and Miguel Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *Int. J. Approx. Reasoning*, 2010. 2.2

[31] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014. 3

[32] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 193–202, 2014. 6.1.2

[33] Chris Ding, Tao Li, and Michael Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE TPAMI*, 2010. 2.4.2

[34] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *JMLR '11*, pages 2121–2159, 2011. 2.3.2

[35] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 278–288, 2015. 4, 4.1.2

[36] Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. Learning generic sentence representations using convolutional neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 2380–2390, 2017. 6.1.4

[37] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 340–348, 2010. 6.1.1

[38] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. How should i explain? a comparison of different explanation types for recommender systems. *Int. J. Hum.-Comput. Stud.*, 72(4):367–382, April 2014. 1

[39] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 2672–2680, 2014. 6.2.4

[40] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proc. 54th Annual Meeting of the Association for Computational Linguistics, ACL*, 2016. 6.1.1

[41] Ido Guy, Naama Zwerdling, David Carmel, Inbal Ronen, Erel Uziel, Sivan Yogev, and Shila

Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *Proc. RecSys*, 2009. 2.2

[42] Ben Hachey, Will Radford, and Andrew Chisholm. Learning to generate one-sentence biographies from wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pages 633–642, 2017. 6.1.4

[43] Taher H. Haveliwala. Topic-sensitive pagerank. In *Proc. WWW '02*, pages 517–526, 2002. 2.3.1

[44] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1661–1670, 2015. 6.1.2

[45] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182, 2017. 4.1.2

[46] J. Herlocker, J. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *CSCW*, pages 241–250, 2000. 3.1

[47] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *Deep Learning and Representation Learning Workshop*, NIPS '14, 2014. 4.1.3

[48] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, November 1997. 4.1.2

[49] Julia Hoxha and Achim Rettinger. First-order probabilistic model for hybrid recommendations. In *Proc. ICMLA '13*, pages 133–139, 2013. 2.2

[50] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2410–2420, August 2016. 4.1.3

[51] Mohsen Jamali and Martin Ester. Trustwalker: A random walk model for combining trust-based and item-based recommendation. In *Proc. KDD*, 2009. (document), 2.2

[52] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proc. SIGKDD*, 2002. 2.4.2

[53] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 233–240, 2016. 4, 4.1.2

[54] Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1317–1327, 2016. 4.1.3

[55] Ioannis Konstas, Vassilios Stathopoulos, and Joemon M. Jose. On social networks and collaborative recommendation. In *Proc. SIGIR '09*, pages 195–202, 2009. 2.2

[56] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proc. KDD '08*, pages 426–434, 2008. 2.3.4, 4, 4.2.3

[57] Pigi Kouki, Shobeir Fakhraei, James Foulds, Magdalini Eirinaki, and Lise Getoor. Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In *Proc.*

RecSys '15, pages 99–106, 2015. 2.2, 3.1

[58] Pigi Kouki, James Schaffer, Jay Pujara, John O'Donovan, and Lise Getoor. User preferences for hybrid explanations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 84–88, 2017. 3.1

[59] Mathias Kraus and Stefan Feuerriegel. Sentiment analysis based on rhetorical structure theory: Learning deep neural networks from discourse trees. *CoRR*, abs/1704.05228, 2017. URL `http://arxiv.org/abs/1704.05228`. 4.1.3

[60] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014. 4.2.4

[61] Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1203–1213, 2016. 6.1.4

[62] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4.2.1

[63] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1106–1115, 2015. 6.1.4

[64] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. Neural rating regression with abstractive tips generation for recommendation. In *Proc. 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 345–354, 2017. 6.1.3

[65] Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 811–820, 2015. 4, 4.1.2

[66] Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan, and Eric P. Xing. Recurrent topic-transition GAN for visual paragraph generation. *CoRR*, abs/1703.07022, 2017. 6.2.4

[67] Chin-Yew Lin and Eduard H. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL 2003*, 2003. 1

[68] Thomas Lin, Mausam, and Oren Etzioni. Entity linking at web scale. In *Proc. AKBC-WEKEX '12*, pages 84–88, 2012. 2.1

[69] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proc. Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2181–2187, 2015. 2.2

[70] Guang Ling, Michael R. Lyu, and Irwin King. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 105–112, 2014. 4, 4.1.1

[71] Zachary Chase Lipton, Sharad Vikram, and Julian McAuley. Capturing meaning in product reviews with character-level generative text models. *CoRR*, abs/1511.03683, 2015. 6.1.3, 6.2.1,

6.2.2, 2, 6.3.2

[72] Chenxi Liu, Junhua Mao, Fei Sha, and Alan L. Yuille. Attention correctness in neural image captioning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4176–4182, 2017. 6.1.4

[73] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag New York, Inc., 1984. 2.3.2

[74] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via A visual sentinel for image captioning. *CoRR*, abs/1612.01887, 2016. URL `http://arxiv.org/abs/1612.01887`. 6.1.4

[75] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 287–296, 2011. 2.2

[76] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. 2.4.2

[77] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *RecSys '13*, pages 165–172. 3.1

[78] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 165–172, 2013. 4, 4.1.1, 4.2.3, 4.3.2, 4.3.3

[79] Julian McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ICDM '12, pages 1020–1025, 2012. 6.1.2, 6.3.2

[80] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 785–794, 2015. 4.3.1, 6.3.2

[81] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 43–52, 2015. 4.3.1, 6.3.2

[82] Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. In *Proc. WWW '13*, pages 897–908, 2013. 6.3.2

[83] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002. 2.4.2

[84] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, 2016. 6.1.4

[85] Tom Mitchell, William Cohen, Estevam Hruschka Jr., Partha Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei

Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. Never-ending learning. In *Proc. AAAI*, 2015. 2.1, 2.2, 3.1

[86] Subhabrata Mukherjee, Kashyap Popat, and Gerhard Weikum. Exploring latent semantic factors to find useful product reviews. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 480–488, 2017. 6.1.2

[87] C. Musto, P. Basile, M. de Gemmis, P. Lops, G. Semeraro, and S. Rutigliano. Automatic selection of linked open data features in graph-based recommender systems. In *CBRecSys 2015*. 2.2, 3.1

[88] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proc. Thirty-First AAAI Conference on Artificial Intelligence*, pages 3075–3081, 2017. 6.1.1

[89] Shashi Narayan, Nikos Papasarantopoulos, Mirella Lapata, and Shay B. Cohen. Neural extractive summarization with side information. *CoRR*, abs/1704.04530, 2017. URL `http://arxiv.org/abs/1704.04530`. 6.1.1

[90] Hitoshi Nishikawa, Takaaki Hasegawa, Yoshihiro Matsuo, and Gen-ichiro Kikui. Optimizing informativeness and readability for sentiment summarization. In *ACL Short Papers*, pages 325–330, 2010. 6.1.1

[91] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 2643–2651, 2013. 4.1.2

[92] V. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *RecSys '13*, pages 85–92. 2.2, 3.1

[93] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proc. Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 32–36, 2017. 2.2

[94] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, 2002. 2

[95] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification ofinteresting web sites. *Mach. Learn.*, 27(3):313–331, June 1997. 2.4.2

[96] Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 873–883, 2017. 4.1.3

[97] Mickaël Poussevin, Vincent Guigue, and Patrick Gallinari. Extended recommendation framework: Generating the text of a user review as a personalized summary. 6.1.2

[98] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1060–1069, 2016. 6.2.4

[99] Steffen Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 995–1000, 2010. 4.2.2

[100] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 379–389, 2015. 6.1.1, 6.2.3

[101] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264, 2007. 4, 4.2.3, 4.3.3

[102] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS 2016*, pages 2226–2234, 2016. 6.2.4

[103] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Representation learning of users and items for review rating prediction using attention-based convolutional neural network. In *3rd International Workshop on Machine Learning Methods for Recommender Systems (MLRec)*, SDM '17, 2017. 4, 4.1.2

[104] Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. Order-planning neural text generation from structured data. *CoRR*, abs/1709.00155, 2017. URL `http://arxiv.org/abs/1709.00155`. 6.1.4

[105] A. Sharma and D. Cosley. Do social explanations work?: Studying and modeling the effects of social explanations in recommender systems. In *WWW '13*, pages 1133–1144, 2013. 3.1

[106] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics, 2013. 4.1.3, 4.2.4

[107] Hongya Song, Zhaochun Ren, Shangsong Liang, Piji Li, Jun Ma, and Maarten de Rijke. Summarizing answers in non-factoid community question-answering. In *Proc. Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 405–414, 2017. 6.1.1

[108] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014. 4.2.2

[109] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proc. WWW*, 2007. 2.1

[110] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 2011. 2.2.1

[111] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. Rating-boosted latent topics: Understanding users and items with ratings and reviews. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2640–2646, 2016. 4.1.1

[112] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proc. 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, 2015. 2.2

[113] Jian Tang, Yifan Yang, Samuel Carton, Ming Zhang, and Qiaozhu Mei. Context-aware natural

language generation with recurrent neural networks. *CoRR*, abs/1611.09900, 2016. URL http://arxiv.org/abs/1611.09900. 6.1.3

[114] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Translational recommender networks. *CoRR*, abs/1707.05176, 2017. URL http://arxiv.org/abs/1707.05176. 4.1.2

[115] Nava Tintarev and Judith Masthoff. *Designing and Evaluating Explanations for Recommender Systems*, pages 479–510. Springer US, Boston, MA, 2011. 1, 1

[116] Hanghang Tong and Christos Faloutsos. Center-piece subgraphs: Problem definition and fast solutions. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 404–413, 2006. 3.1

[117] J. Vig, S. Sen, and J. Riedl. Tagsplanations: Explaining recommendations using tags. In *IUI '09*, pages 47–56. 3.1

[118] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010. 4.1.2

[119] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 6.1.4

[120] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):652–663, April 2017. 6.1.4

[121] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 448–456, 2011. 4.1.1, 4.1.2, 4.2.3, 4.3.3

[122] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1235–1244, 2015. 4, 4.1.2, 4.2.3, 4.3.3

[123] Lu Wang and Wang Ling. Neural network-based abstract generation for opinions and arguments. 2016. 6.1.1

[124] M. Wang, M. Liu, J. Liu, S. Wang, G. Long, and B. Qian. Safe Medicine Recommendation via Medical Knowledge Graph Embedding. *ArXiv e-prints*, October 2017. 2.2, 3.1

[125] William Yang Wang and William W. Cohen. Joint information extraction and reasoning: A scalable statistical relational learning approach. In *Proc. ACL 2015*, pages 355–364, 2015. 2.3.2

[126] William Yang Wang, Kathryn Mazaitis, and William W. Cohen. Programming with personalized pagerank: A locally groundable first-order probabilistic logic. In *Proc. CIKM '13*, pages 2129–2138, 2013. 2, 2.3.2, 2.3.2, 3.1, 3.2, 3.2

[127] Zhongqing Wang and Yue Zhang. Opinion recommendation using A neural model. In *Proc. Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1627–1638, 2017. 6.1.3, 6.2.2, 4, 6.3.1

[128] Jonatas Wehrmann, Willian Becker, Henry E. L. Cagnini, and Rodrigo C. Barros. A character-

based convolutional neural network for language-agnostic twitter sentiment analysis. In *2017 International Joint Conference on Neural Networks, IJCNN*, pages 2384–2391, 2017. 4.1.3

[129] Chao-Yuan Wu, Alex Beutel, Amr Ahmed, and Alexander J. Smola. Explaining reviews and ratings with paco: Poisson additive co-clustering. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, pages 127–128, 2016. 6.1.2

[130] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 153–162, 2016. 4.1.2

[131] Han Xiao, Minlie Huang, and Xiaoyan Zhu. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *Proc. Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 1315–1321, 2016. 2.2, 3.1

[132] Wenhan Xiong, Thien Hoang, and William Yang Wang. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Proc. 2017 Conference on Empirical Methods in Natural Language Processing EMNLP*, pages 575–584, 2017. 2.2, 3.1

[133] Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2048–2057, 2015. 6.1.4

[134] Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. Words or characters? fine-grained gating for reading comprehension. *CoRR*, abs/1611.01724, 2016. URL `http://arxiv.org/abs/1611.01724`. 6.2.2

[135] Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. Multi-task cross-lingual sequence tagging from scratch. *CoRR*, abs/1603.06270, 2016. URL `http://arxiv.org/abs/1603.06270`. 6.2.2

[136] Zhilin Yang, Ye Yuan, Yuexin Wu, William W. Cohen, and Ruslan Salakhutdinov. Review networks for caption generation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pages 2361–2369, 2016. 6.1.4

[137] Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y. Zhao. Automated crowdturfing attacks and defenses in online review systems. *CoRR*, abs/1708.08151, 2017. URL `http://arxiv.org/abs/1708.08151`. 6.1.1

[138] Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir R. Radev. Graph-based neural multi-document summarization. In *Proc. 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, 2017. 6.1.1

[139] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. Beyond clicks: Dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 113–120, 2014. 4.2.2

[140] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM '14*, pages 283–292. 3.1

[141] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Bran-

don Norick, and Jiawei Han. Recommendation in heterogeneous information networks with implicit user feedback. In *Proc. RecSys*, 2013. 2.2

[142] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proc. WSDM*, 2014. 2, 2.2, 2.2.1, 2.4.1, 2.4.2, 2.4.5

[143] Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1114–1125, 2017. 4.1.3

[144] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR '14*, pages 83–92. 3.1

[145] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 83–92, 2014. 4.1.1, 6.1.2

[146] Lei Zheng, Vahid Noroozi, and Philip S. Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 425–434, 2017. 4, 4.1.2, 4.2.1, 4.2.2, 4.3.3, 1