Explainable Recommendations

Rose Catherine Kanjirathinkal

CMU-LTI-18-014

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, USA
www.lti.cs.cmu.edu

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
in Language and Information Technologies

To my loving parents and my beloved husband

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 serve many different purposes. In general, providing an explanation has been shown to build user's trust in the recommender system.

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. 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 in dealing with natural language content.

In addition to KGs, free form text such as reviews are another 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 an oracle-student architecture, called TransNet, can be used to transform latent representations of user and item into that of their joint review to improve recommendation performance.

We also show how TransNet can be used to select a candidate review that is most similar to the joint review. Such a review could possibly serve as an explanation of why the user would potentially like the item. However, 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. Ideally, a user would be shown a personalized summary of all relevant reviews for that item. In the third part of this thesis, we adapt our TransNet model to generate the user's review for a given item and compare its performance against the state-of-the-art models extensively using both automatic and human evaluations.

Although predicting a user's review for an item provides valuable information, it is not structured like an explanation. The main difficulty in training models to generate explanations is the lack of ground truth explanation data. In the final part of this thesis, we present a new gold standard dataset, the first of its kind, for personalized explanations that can be used to train neural models to generate detailed personalized recommendations in natural language.

Contents

List of Figures

List of Tables

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 from 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 [154]. Below are some of the important ones:

1. Transparency: Explanations that describe how a recommendation was chosen makes the system transparent to the user. Such explanations are called interpretable. 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: Explanations enable users to make good decisions by helping them understand why they would like or dislike certain aspects of a particular recommendation. We refer to them as *Explainable Recommendations that are Explanatory* or, *Explanatory Recommendations* for short.
5. Efficiency: Explanations help users make decisions quicker, especially 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 [48, 154]. 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 thesis are most in line with 'Effectiveness' in the above list of purposes of providing explanations. Our approaches strive to also explain why a user may 'dislike' an item. i.e. they are not limited to finding only positives or persuading the user to buy the item. A prominent use-case for our type of explanations would be as an explanation module in a virtual assistant like Amazon Alexa, Apple Siri, Google Home, Microsoft Cortana or CMU InMind SARA [125]. For example, the user may want to know how they would like a particular product — i.e. how does it measure in those aspects that the user cares about — and make an informed choice. 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 the Chapter 2 of this thesis , 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. We also propose an approach to generate explanations in this setting by jointly ranking the entities with the items. This work was published in the 10^{th} and the 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 easily. These free form texts 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 that is discussed in the Chapter 3 of this thesis, we use neural models to leverage reviews written by users. The set of reviews written by a user illustrates his or her

interests, and can be used to construct a representation for them in a latent vector space. Similarly, reviews written for an item can be used to learn its overall representation. Items to be recommended are scored using a neural regression model. Our model, called *Transformational Neural Network* (TransNet), generates a latent representation not of the entities, but of the user's predicted review. In this model, an explanation provided is via the most similar review written for the item, 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).

Selecting reviews to show as explanation suffers from a number of limitations. Most importantly, it is almost always impossible to find an existing review written for the item by other users, that covers all the aspects of the item that the user at hand cares about. Although users may be similar in their overall opinion about an item, each user's experience with the item is bound to be different. Therefore, in Chapter 4, we tackle the research problem of generating user's reviews for items. We adapt our previously proposed TransNet model for the purpose of review generation. The explanation in this setting is the generated review itself, that elucidates the properties of the item that the user cares about, whether they would like it or not, and why, in natural language text, all personalized to the user.

While showing to a user what their review would look like if they were to experience an item is valuable, it still is neither an explanation nor a detailed recommendation — it is a review. The main difficulty in generating 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 thesis in Chapter 5, we present the first ever dataset created for personalized explanations and detailed recommendations with human written text. The dataset was created using Amazon Mechanical Turk[1] and contains personalized recommendations and explanations for 1000 users and 900 businesses.

This thesis is organized as follows: Chapter 2 details techniques to improve recommendation accuracy and generate explanations using Knowledge Graphs. This is followed by Chapter 3 that discusses techniques for the same goals, but using neural networks. Subsequently, in Chapter 4, we delve into techniques for review generation. In Chapter 5, we present a new gold standard dataset for personalized explanations. Finally, in Chapter 6, we conclude with a discussion about the contributions of this thesis and some interesting directions of future research. Prior work relevant to the research reported in this thesis is presented in the corresponding chapters.

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

Chapter 2

Rating Prediction and Explanation using Knowledge Graphs

In this chapter, we first show how Knowledge Graphs can be leveraged to improve personalized recommendations, using a general-purpose probabilistic logic system called ProPPR[167]. 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 by [186]. 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) [23].

In the second part of this chapter, we shift our focus to generating explanations for Knowledge Graph (KG) -based recommendations. Although a number of explanation schemes have been proposed in the past, at the time this work was proposed, there were no existing approaches that produced explanations for KG-based recommenders. 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. This work was published in the 11^{th} ACM Conference on Recommender Systems (RecSys '17) as a Poster [26].

2.1 Background and Concepts: Knowledge Graphs

In this Thesis, we use the term *entity* as a generic term to denote a word or a phrase that can be mapped onto a knowledge base (KB) or an ontology. Since the datasets used in this Thesis already have links into a structured knowledge base, the mapping is straightforward. However, when using a generic knowledge base like Wikipedia[1], Yago [146] or NELL [108], one might require a wikifier or an entity linker [88]. 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

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

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). A HIN is in contrast to prior works that use graphs or networks of only one type of node, like say, a friend network; these are called homogeneous. A KG, as referred to in this Thesis, 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 unknown, then only some methods are applicable. However, if the knowledge graph is indeed an HIN, then all three methods apply.

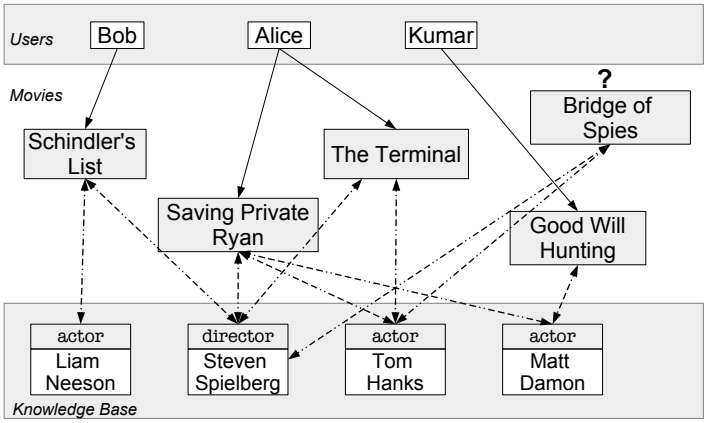

Figure 2.1: Example of Movie Recommendation with a Knowledge Graph

A typical movie recommendation example is depicted in Figure 2.1 where users watch and/or rate movies, and information about the movies is 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 information 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 particular 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, for example, `Tom Hanks` and `Best Actor Academy Award`.

2.2 Related Work

We have grouped the related work into two subsections. The first subsection discusses relevant literature for KG based recommendations and the second, details that for entity based explanations.

2.2.1 Knowledge Graphs for Recommendation

Recommendation systems have been popular for a long time now and are a well researched topic [124]. However, there has not been much effort directed at using external KGs for improving recommendations.

A recent method `HeteRec_p` [186], 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 2.2.1.1, since our approaches were compared against it. [185] is another link-based method proposed by the same authors but precedes [186]. It learns a global model of recommendation based on the KG, but does not attempt to personalize the recommendations. A similar method was proposed in [119], which used paths to find the top-N recommendations in a learning-to-rank framework. These methods use only the types of the nodes and do not track the actual entities that appear in the path. For example, their paths could be of the form `User` $\rightarrow$ `Movie` $\rightarrow$ `Actor` $\rightarrow$ `Movie` for movie ranking. The paths in our work contain the actual entities that appear in the data. A few methods such as [108, 111] 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. The ranking process does not involve a training step to learn to walk on the graph, unlike the methods proposed in this Thesis.

A recent work [164] that appeared after the publication of our methods, 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 [89] and LINE [150]. The recommendations are scored using their similarity in the embedded space. There have been other related efforts in the KG space that postdates our work. For example, [120] and [121] enhance an existing graph embedding method called `node2vec`, to account for the different kinds of properties/relations. [178] 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 [175], where their model learned a manifold-based embedding of the graph. Yet another approach proposed in [67] uses attention over multiple paths connecting two nodes to learn a representation for that entity pair.

Another effort at using multiple sources of information is the HyPER system [75], where the authors show how to recommend using a Probabilistic Soft Logic framework [12]. 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 [62] that uses Markov Logic Networks, and that proposed in [39] 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 [65], 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 [53, 73, 95] 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. Our approachs use external information about the items in contrast to the link graph of users employed by these methods. Nodes in our KG are not necessarily of the same type, unlike that of a social network. In [17], 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, but they do not use any PageRank style methods.

2.2.1.1 Heterogeneous Recommender - Personalized (HeteRec_p)

The Heterogeneous Recommender - Personalized (`HeteRec_p`) method proposed in [186] 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 $i = 1, \ldots, k-1$. In the above example, this path would be of the form $P = \texttt{User} \xrightarrow{\texttt{viewed}} \texttt{Movie} \xrightarrow{\texttt{starredIn}^{-1}} \texttt{Actor} \xrightarrow{\texttt{starredIn}} \texttt{Movie}$

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 [148] 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} clus-

ter 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 [37]. 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.

2.2.2 Knowledge Graphs for Explanation

[59] 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 [3] 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 [2] 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 [18] simply displayed keyword matches between the user's profile and the books being recommended. Similarly, [157] 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, [140] 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. [189] extracted phrases and sentiments expressed in the reviews and used them to generate explanations. [99] 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.

[155] 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.

Another work [76], which is contemporary with our work, evaluated different ways of showing explanations for recommendations produced using HyPER [75], a recommender system that uses Probabilistic Soft Logic (PSL) [12]. 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 [167]. The user study in [76] 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.

2.3 Proposed Approach for Knowledge Graph based Recommendations

2.3.1 Recommendation as Personalized PageRank

Consider the example in Figure 2.1. Similar to the Topic Sensitive PageRank proposed in [55] and the weighted association graph walks proposed in [20], 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 [13] 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 [167], 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 [92] 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 conjectured to be 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.

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

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

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

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 higher

those entities that are reachable from the movies that the user reviewed through multiple short paths, and rank lower entities only reached by fewer, longer paths.

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

$$\texttt{likesEntity}(\texttt{U},\texttt{E}) \leftarrow \{\texttt{l}(\texttt{U},\texttt{E})\}. \quad (2.5)$$

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 that uses the seeds for a user 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.

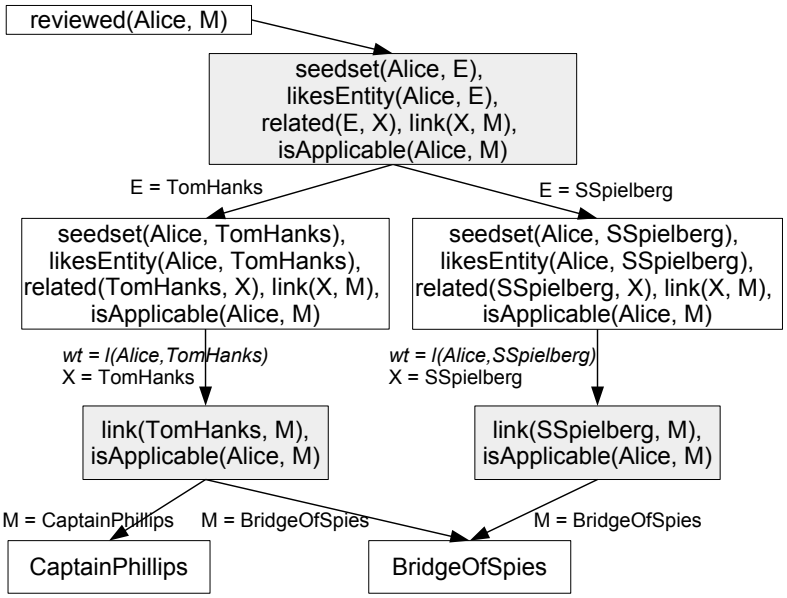

Figure 2.5: Sample grounding of the `EntitySim` ProPPR program

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 2.10, 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 [13], 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 [13]. To learn $\mathbf{w}$, we use AdaGrad [43] instead of the quasi-Newton method used in [13] and SGD used in [167]. 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 [167] and [166].

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.

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

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

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

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

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

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

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 $\mathtt{p}(\mathtt{E})$. 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 [74]. 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 [74] 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.

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

$$\mathtt{related}(\mathtt{U},\mathtt{E}) \leftarrow \mathtt{seedset}(\mathtt{U},\mathtt{E}), \mathtt{simLF}(\mathtt{U},\mathtt{E}). \tag{2.14}$$

$$\mathtt{related}(\mathtt{X},\mathtt{X}) \leftarrow. \tag{2.15}$$

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

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

$$\mathtt{val}(\mathtt{X},\mathtt{D}) \leftarrow \{\mathtt{v}(\mathtt{X},\mathtt{D})\}. \tag{2.18}$$

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 for Recommendation

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 [186], 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 [186], 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 [186], and have 4.8% more reviews than [186].

For all the datasets, similar to [186], 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 3.3.1.

2.4.2 Experimental Setup

We evaluate the performance using standard metrics, Mean Reciprocal Rank (MRR) and Precision at 1, 5 and 10 (P@1, P@5 and P@10) [96].

[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/

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

In our experiments, we found that any reasonable choice for the seed set size worked well enough. A small 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 [186] 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 more computationally intensive algorithms [123]. 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 [8] package. Since HeteRec_p was shown to be better than Popularity (which shows the globally popular items to users), Co-Click (which uses the co-click probabilities to find similar items), Non–Negative Matrix Factorization[42] and Hybrid-SVM (which uses a Ranking SVM[68] on metapaths) in [186], 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 the precision metrics.

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

2.4.4 Performance Comparison on IM100K

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.

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 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 [186]: in IM100K dataset, each user has rated at least 20 movies.

To quantitatively compare their differences, we define the Density of a dataset as $\frac{\#\texttt{reviews}}{\#\texttt{users} \times \#\texttt{items}}$, 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.

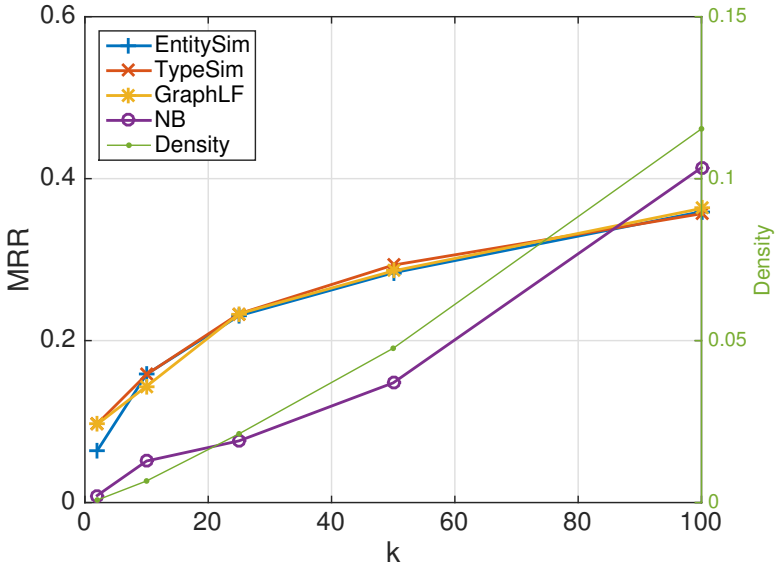

Figure 2.8: Performance of different methods with varying graph densities on Yelp

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 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 Proposed Approach for Entity-based Explanations

In this section, we shift our focus to generating entity-based explanations in a KG based recommender system. 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, earlier in this chapter. 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 2.9), and a similar set for movies that the user would not like. In Figure 2.9, Rule 1 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 1 invokes the predicate likes(U,E), which holds for an entity E if the user has explicitly stated that they like it (Rule 2), 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 3). The method for finding movies and entities that the user will dislike is similar to the above, except 'like' is replaced with 'dislike'.

$$\texttt{willLike}(\texttt{U},\texttt{E},\texttt{M}) \leftarrow \texttt{likes}(\texttt{U},\texttt{E}), \texttt{sim}(\texttt{E},\texttt{M}), \texttt{isMovie}(\texttt{M}). \quad \text{(Rule 1)}$$

$$\texttt{likes}(\texttt{U},\texttt{E}) \leftarrow \texttt{likesEntity}(\texttt{U},\texttt{E}). \quad \text{(Rule 2)}$$

$$\texttt{likes}(\texttt{U},\texttt{E}) \leftarrow \texttt{likesMovie}(\texttt{U},\texttt{M}), \texttt{link}(\texttt{M},\texttt{E}). \quad \text{(Rule 3)}$$

Figure 2.9: 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 as discussed in Section 2.3.2. A sample grounding when queried for a user alice who likes tom_hanks and the movie da_vinci_code is shown in Figure 2.10.

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

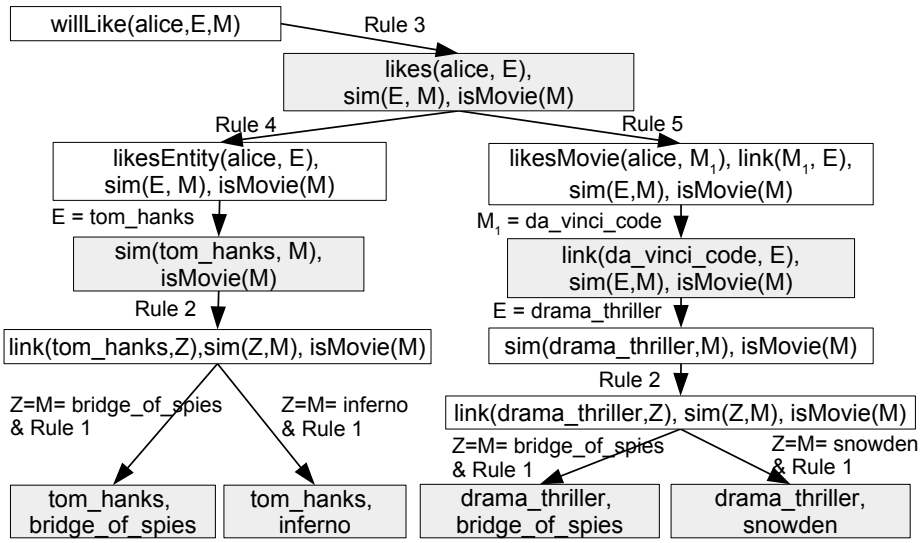

Figure 2.10: Sample grounding for predicting likes

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 }

2.6 Real World Deployment and Evaluation of Explanations

The technique presented in this Chapter is currently being used as the backend of a personal agent running on mobile devices for recommending movies [9] 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 required to give a feedback about the explanations after *every* interaction due to concerns that it might reduce the usability of the app.

2.7 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.

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 also presented a method to produce a ranked list of entities as explanations by jointly ranking them with the corresponding movies.

Chapter 3

Rating Prediction and Review Selection using TransNets

Using review text for predicting ratings has been shown to greatly improve the performance of recommender systems [15, 90, 100], compared to Collaborative Filtering (CF) techniques that use only past ratings [74, 136]. 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 [14, 44, 69, 84, 162] have focussed on the *content* associated with the user and the item, which is used to construct their latent representations. Content associated with a user could include their demographic information, socioeconomic characteristics, their product preferences and the like. Content linked to an item could include their price, appearance, usability and similar attributes in the case of products, food quality, ambience, service and wait times in the case of restaurants, or actors, director, genre, and similar metadata in the case of movies.

Since review text describes a user's reaction to an item, it 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* [4] feature. Only a few neural net models [6, 138, 194] have been proposed to date that use review text for predicting the rating. Of these, one recent model, *Deep Cooperative Neural Networks* (DeepCoNN) [194], 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) [24].

3.1 Related Work: Recommendation using Reviews

3.1.1 Non-Neural Models

The *Hidden Factors as Topics* (HFT) model [100] 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 [15], 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 [90] where the rating is sampled from a Gaussian mixture. A recent model [31] that postdates our work uses topic modeling and attention mechanism to model user's preferences over aspects of items.

The *Collaborative Topic Regression* (CTR) model proposed in [161] is a content based approach, as opposed to a context / review based approach. It uses LDA [19] 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 [149] 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. The *Explicit Factor Models* (EFM) proposed in [190] aims to generate explainable recommendations by extracting explicit product features (aspects) and users' sentiments towards these aspects using phrase-level sentiment analysis. Such explanations are discrete entity based ones like those discussed in Chapter 2 and not in fully constructed natural language sentences.

3.1.2 Neural Net Models

One recent model to successfully employ neural networks at scale for rating prediction is the *Deep Cooperative Neural Networks* (DeepCoNN) [194], which will be discussed in detail in Section 3.2. Prior to that work, [6] 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 [61], was computed using the item factors. In [138], contemporary to our work, the authors 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. Similar to DeepCoNN, it trains its regression layer directly unlike TransNet, which uses an intermediate layer that is trained to mimic another network. Another recent model, [63] that postdates our work, uses cross-domain information transfer to improve rating prediction. The rating prediction results reported in this chapter were also independently verified by some of the later publications such as [173] and [153].

Prior research has used deep neural nets for learning latent factors from ratings alone, i.e., with-

out using any content or review. The *Collaborative De-noising Auto-Encoder* model (CDAE) [174] learns to reconstruct user's feedback from a corrupted version of the same. In [152], 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 $||u + r - i|| \approx 0$.

Some of the other past research use neural networks in a CF setting with content, but not reviews. The *Collaborative Deep Learning* (CDL) model [162] uses a Stacked De-noising Auto Encoder (SDAE) [158] to learn robust latent representations of items from their content, which is then fed into a CTR model [161] for predicting the ratings. A very similar approach to CDL is the *Deep Collaborative Filtering* (DCF) method [84] which uses Marginalized De-noising Auto-Encoder (mDA) [28] instead. The *Convolutional Matrix Factorization* (ConvMF) model [69] 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 [44] represents users using their search queries and clicked urls, and items (news articles) using their title, named entities in the text and categories. These features are then processed using a deep neural net to map into a shared latent space such that their similarity in that space is maximized. [118] 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. [14] also proposed a similar technique but adapted to recommending scientific-articles. [34] used a deep neural net to learn a latent representation from video content which is then fed into a deep ranking network. [57] is an extension to MF with additional non-linear layers instead of the the dot product.

3.1.3 Comparison to Related Architectures and Tasks

3.1.3.1 Student-Teacher Models

The model proposed in this chapter resembles a Student-Teacher model in certain aspects. Student-Teacher models [21, 60] also have two networks; A Teacher Network (similar to the *Oracle Network* in our model), which is large and complex, and typically an ensemble of different models, and a much simpler Student Network (similar to the *Student Network* in our model), which learns to emulate the output of the Teacher Network.

However, 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 Oracle, there is only one input – the review by $user_A$ for an $item_B$ designated as rev_{AB}. But, in the Student, 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 more 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 Oracle is lean while the Student is heavy-weight, often processing large pieces of text using twice the number of parameters as the Oracle. Thirdly, in Student-Teacher models, the Teacher is pre-trained whereas in TransNets, the Oracle is trained simultaneously with the Student. A recently proposed Student-Teacher model in [64] does train both the Student and the Teacher simultaneously. Also, in Student-Teacher models, the emphasis is on

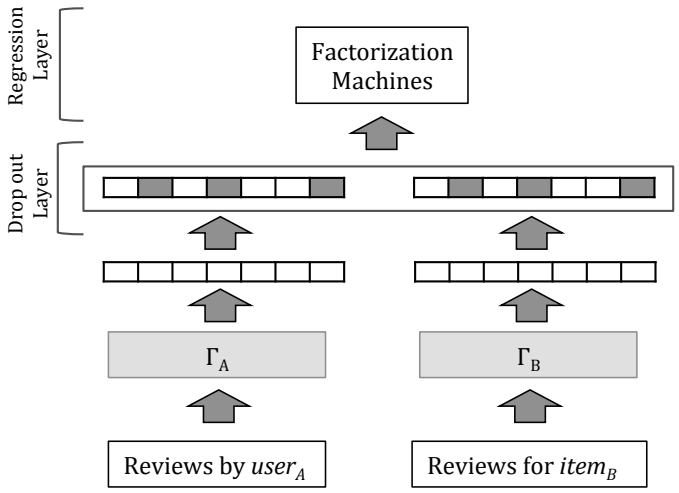

Figure 3.1: DeepCoNN model for predicting rating

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 [70].

3.1.3.2 Sentiment Analysis

Part of the model proposed in this chapter, referred to as the *Oracle Network*, resembles a sentiment analyzer because it inputs a user's review for an item and predicts that user's rating for that item. In [141], the authors proposed a Recursive Neural Tensor Network that could achieve high accuracy levels for sentence and phrase level sentiment prediction. [77] proposed a tree-structured deep NN for predicting sentiment from the discourse structure. [170] uses a character-level CNN for Twitter sentiment analysis whereas [127] uses an LSTM on utterances in a video to compute its sentiment. In [187], 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. [10] computes an aspect-level sentiment using an aspect graph and a rhetorical structure tree. Compared to the above models, only one of the subnetworks of TransNet resembles a sentiment analyzer. The other subnetwork performs rating prediction from user and item representations, which is a different task.

3.2 The TransNet Method

In this section, we describe our model called TransNet. But, before we delve into its details, we first discuss the state-of-the-art method at the time, DeepCoNN, and its limitations.

3.2.1 DeepCoNN model

To compute the rating r_{AB} that $user_A$ would assign to $item_B$, the DeepCoNN model of [194] uses two text processing modules, Γ_A and Γ_B side by side as shown in Figure 3.1. These modules, called *CNN Text Processors*, are described in the next subsection. 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 [144]. 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) [133]. 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 [133] (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[183].

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 [194] also uses L_1 loss. Therefore, in this work, all FMs are trained using L_1 loss, defined as:

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

3.2.2 CNNs to process text

We process text using the same approach as our competitive baseline for rating prediction, *DeepCoNN* [194]. The basic building block, referred to as a *CNN Text Processor* in the rest of this chapter, is a Convolutional Neural Network (CNN) [81] 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 $\Gamma : [w_1, w_2, ..., w_T] \rightarrow \mathbb{R}^n$. Figure 3.2 gives the architecture of the *CNN Text Processor*. In the first layer, a word embedding function $f : M \rightarrow \mathbb{R}^d$ maps each word in the review that are also in its M-sized vocabulary into a d dimensional vector. The embedding can be any pre-trained embedding

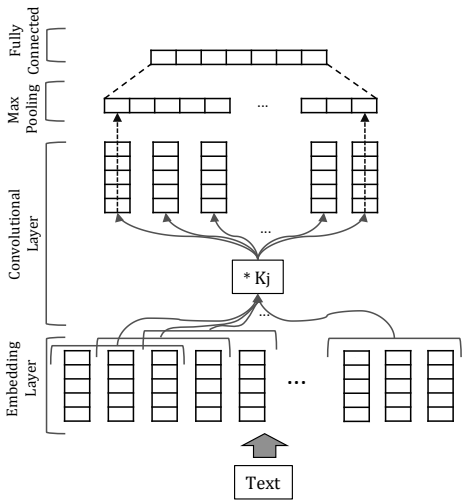

Figure 3.2: The *CNN Text Processor* architecture

like those trained on the GoogleNews corpus using *word2vec*[1][107], or on Wikipedia using GloVe[2] [126]. These word vectors are held fixed throughout the training process.

Following the embedding layer is the Convolutional Layer, adapted to text processing [32]. It consists of m neurons each associated with a filter $K \in \mathbb{R}^{t \times d}$, where t is a window size, typically 2 – 5. The filter processes t-length windows of d-dimensional vectors to produce features. Let $V_{1:T}$ be the embedded matrix corresponding to the T-length input text. Then, j^{th} neuron produces its features as:

$$z_j = \alpha(V_{1:T} * K_j + b_j)$$

where, b_j is its bias, $*$ is the convolution operation and α is a non-linearity like Rectified Linear Unit (ReLU) [112] or $tanh$.

Let $z_j^1, z_j^2, ...z_j^{(T-t+1)}$ be the features produced by the j^{th} neuron on the sliding windows over the embedded text. Then, the final feature corresponding to this neuron is computed using a max-pooling operation, defined as:

$$o_j = \max\{z_j^1, z_j^2, ...z_j^{(T-t+1)}\}$$

The max-pooling operation provides location invariance to the neuron, i.e., the neuron is able to detect the features in the text regardless of where it appears.

The final output of the Convolutional Layer is the concatenation of the output from its m neurons, denoted by:

$$O = [o_1, o_2, ...o_m]$$

This output is then passed to a fully connected layer consisting of a weight matrix $W \in \mathbb{R}^{m \times n}$ and a bias $g \in \mathbb{R}^n$, which computes the final representation of the input text as:

$$x = \alpha(W \times O + g)$$

[1]`https://code.google.com/archive/p/word2vec`
[2]`https://nlp.stanford.edu/projects/glove`

3.2.3 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 [100], Collaborative Topic Regression (CTR) [161] and Collaborative Deep Learning (CDL) [162], as well as Collaborative Filtering techniques that use only the rating information like Matrix Factorization (MF) [74] and Probabilistic Matrix Factorization (PMF) [136].

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. This issue is discussed in more detail below.

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 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.

3.2.4 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. Our model, TransNet, exploits rev_{AB} by using it to regularize a model that uses only data available at test time.

TransNet consists of two networks, as shown in the architecture diagram of Figure 3.3. An *Oracle Network* that processes the target review rev_{AB} and a *Student Network* that processes the texts of the $(user_A, item_B)$ pair that excludes the joint review, rev_{AB}. Given a review text rev_{AB}, the Oracle Network uses a CNN Text Processor, Γ_O, and a Factorization Machine, FM_O, to predict the rating

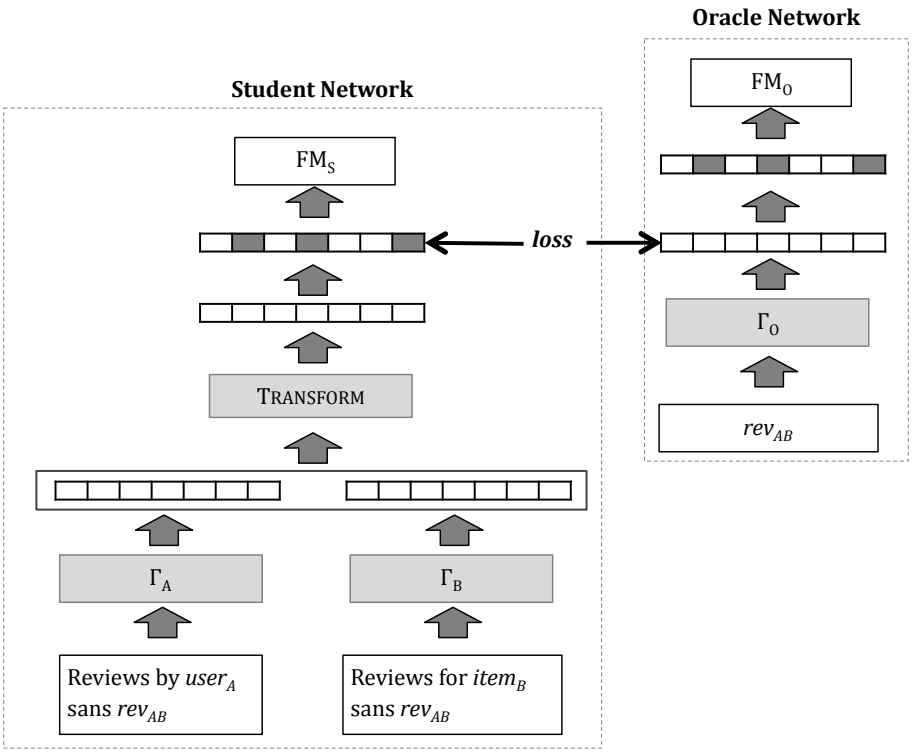

Figure 3.3: The TransNet architecture

as:

$$\begin{aligned} x_O &= \Gamma_O(rev_{AB}) \\ \bar{x}_O &= \delta(x_O) \\ \hat{r}_O &= FM_O(\bar{x}_O) \end{aligned}$$

Since the Oracle Network uses the actual review, its task is similar to sentiment analysis [79, 141].

The Student 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 Student 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. z_0 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. In our experiments, we set σ to *tanh*. 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 $\Gamma_O(rev_{AB})$, the latent representation of the target review. The Student 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 Student Network's representation z_L to be similar to the encoding of rev_{AB} produced by the Oracle Network, as we discuss below.

3.2.5 Training TransNets

TransNet is trained using three 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 Oracle Network, denoted by θ_O, which includes those of Γ_O and FM_O, are updated to minimize a L_1 loss computed between the actual rating r_{AB} and the rating $\hat{r}_O$ predicted from the actual review text rev_{AB}.

To teach the Student Network how to generate an approximation of the latent representation of the original review rev_{AB} generated by the Oracle 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_O 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 Student 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.

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

3.2.6 Design Decisions and Other Architectural Choices

In this section, we describe some of the choices we have made in designing the TransNet architecture and why alternatives did not give good results in our preliminary experiments.

1. **Training with sub-steps vs. jointly:** While training TransNets using Algorithm 1, in each iteration (or batch), we could choose to jointly minimize a total loss, $loss_{total} = loss_O + loss_{trans} + loss_S$. We observed in our experiments that the joint training gave a lower performance. This is probably because joint training will result in parameter updates to the Oracle network, Γ_O, resulting from $loss_{trans}$, in addition to those from $loss_O$, i.e., the Oracle Network will get penalized for producing a representation that is different from that produced

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 Oracle Network on the actual review
      $x_O = \Gamma_O(rev_{AB})$
      $\hat{r}_O = FM_O(\delta(x_O))$
      $loss_O = |r_{AB} - \hat{r}_O|$
      update θ_O to minimize $loss_O$
      #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)
      $\bar{z}_L = \delta(z_L)$
      $loss_{trans} = ||\bar{z}_L - x_O||_2$
      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

Algorithm 3 Testing using TransNet

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

by the Student Network. This results in both networks learning to produce sub-optimal representations and converging to a lower performance in our experiments. Therefore, it was important to separate the Oracle Network's parameter updates so that it learns to produce the best representation which will enable it to make the most accurate rating predictions from the review text.

2. **Training the Oracle Network to convergence independently:** We could choose to first train the Oracle Network to convergence and then train the Student Network to emulate the trained Oracle Network. However, note that the Oracle Network's input is the actual review, which is unavailable for testing its performance, i.e., we do not know when the Oracle Network has converged with good generalization vs. when it is overfitting. The only way to measure the performance at test time is to check the output of the Student Network. Therefore, we let the Student and the Oracle Networks learn simultaneously and stop when the Student Network's test performance is good.
3. **Using the same convolutional model to process text in both the Student and Oracle networks:** We could choose to use the Γ_O that was trained in the Oracle Network to generate features from the user and the item text in the Student Network, instead of learning separate Γ_A and Γ_B. After all, we are learning to transform the latter's output into the former. However, in that case, TransNet would be constrained to generate generic features similar to topics. By providing it with separate feature generators, it can possibly learn to transform the occurrence of different features in the user text and the item text of the Student Network to another feature in the Oracle Network. For example, it could learn to transform the occurrence of features corresponding to say, 'love indian cuisine' & 'dislike long wait' in the user profile, and 'lousy service' & 'terrible chicken curry' in the item (restaurant) profile, to a feature corresponding to say, 'disappointed' in the Oracle review, and subsequently predict a lower rating. Having separate feature generators in the Student Network gives TransNets more expressive power and gave a better performance compared to an architecture that reuses the Oracle Network's feature generator.
4. **Training the Transform without the dropout:** We could choose to match the output z_L of Transform with x_O instead of its dropped out version $\bar{z}_L$ in Step 2 of Algorithm 1. However, this makes the Transform layer unregularized, leading it to overfit, thus giving poor performance.

3.2.7 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 3.4.

The Student 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

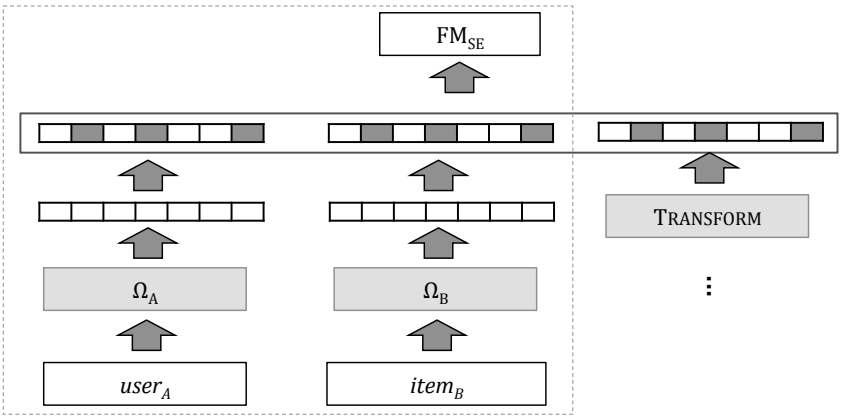

Figure 3.4: The Extended TransNet sub-architecture

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:

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

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.

3.3 Experiments and Results

3.3.1 Datasets

We evaluate the performance of our approach on four large datasets. The first one, `Yelp17`, is from the latest Yelp dataset challenge[3], 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[4] [102, 103] 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 3.3.1. The original size of the dataset before discarding empty reviews is given in paranthesis when applicable.

3.3.2 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

[3] `https://www.yelp.com/dataset_challenge`
[4] `http://jmcauley.ucsd.edu/data/amazon`

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 3.1: Dataset Statistics

the test datasets. We report the MSE obtained on the test dataset when the MSE on the validation dataset was the lowest, similar to [100]. All algorithms, including the competitive baselines, were implemented in Python using TensorFlow[5], 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 (∼6600 batches) on 1 GPU, and gives the best performance in about 2–3 epochs.

Below are the details of the text processing and the parameter settings used in the experiments:

1. **Text Pre-Processing and Embedding:** All reviews are first passed through a Stanford Core NLP Tokenizer [97] to obtain the tokens, which are then lowercased. Stopwords (the, and, is etc.) as well as punctuations are considered as separate tokens and are retained. A 64-dimensional $word2vec$[6] [107] embedding using the Skip-gram model is pre-trained on the 50,000 most frequent tokens in each of the training corpora.
2. **CNN Text Processor:** We reuse most of the hyper-parameter settings reported by the authors of DeepCoNN [194] since varying them did not give any perceivable improvement. In all of the CNN Text Processors Γ_A, Γ_B and Γ_O, the number of neurons, m, in the convolutional layer is 100, the window size t is 3, and n, the dimension of the output of the CNN Text Processor, is 50. The maximum length of the input text, T, is set to 1000. If there are many reviews, they are randomly sorted and concatenated, and the first T tokens of the concatenated version are used. In our experiments, the word embedding dimension, d, is 64, and the vocabulary size, $|M|$ is 50,000. Also, the non-linearity, σ, is *tanh*.
3. **Dropout Layer and Factorization Machines:** All dropout layers have a keep probability of 0.5. In all of the factorization machines, FM_O, FM_S and FM_{SE}, the pair-wise interaction is factorized using a $k = 8$ dimensional matrix, V. Since FM_O processes a n-dimensional input, its parameters are $\mathbf{w}_O \in \mathbb{R}^n$ and $\mathbf{V}_O \in \mathbb{R}^{n \times k}$. Similarly, since FM_{SE} processes a $3n$-dimensional input, its parameters are $\mathbf{w}_{SE} \in \mathbb{R}^{3n}$ and $\mathbf{V}_{SE} \in \mathbb{R}^{3n \times k}$. All $\mathbf{w}$'s are initialized to 0.001, and all $\mathbf{V}$'s are initialized from a truncated normal distribution with 0.0 mean and 0.001 standard deviation. All FMs are trained to minimize an L_1 loss.
4. **Transform:** The default setting for the number of layers, L, is 2. We show the performance for different values of L in Section 3.3.5. All weight matrices G_l are initialized from a truncated normal distribution with 0.0 mean and 0.1 standard deviation, and all biases g_l are initialized to 0.1. The non-linearity, σ, is *tanh*.

[5] https://www.tensorflow.org

[6] https://www.tensorflow.org/tutorials/word2vec#the_skip-gram_model

5. **TransNet-Ext:** The user (item) embedding matrices, Ω, are initialized from a random uniform distribution (-1.0, 1.0), and map users (items) that appear in the training set to a $n = 50$ dimensional space. New users (items) in the validation and test sets are mapped to a random vector.
6. **Training:** All optimizations are learned using Adam [71], a stochastic gradient-based optimizer with adaptive estimates, at a learning rate set to 0.002. All gradients are computed by automatic differentiation in TensorFlow.

3.3.3 Competitive Baselines

We compare our method against the previous state-of-the-art, DeepCoNN [194]. DeepCoNN was previously evaluated against the then state-of-the-art models like Hidden Factors as Topics (HFT) model [100], Collaborative Topic Regression (CTR) [161], Collaborative Deep Learning (CDL) [162] and Probabilistic Matrix Factorization (PMF) [136], 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 [194]. 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. The 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.

3.3.4 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:

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

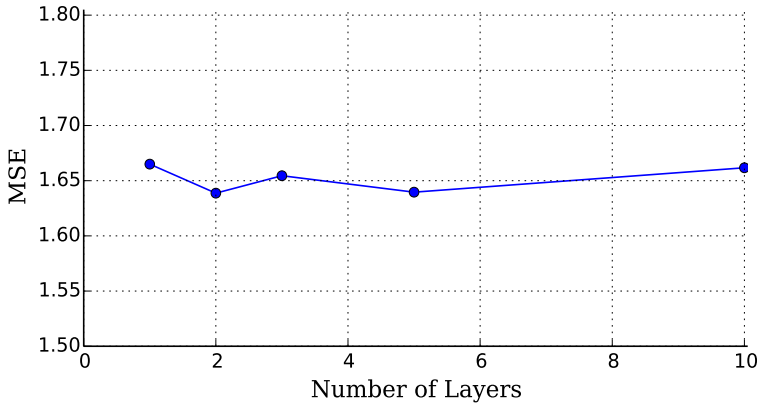

Figure 3.5: Variation in MSE with different Layers: TransNets on `Yelp` dataset

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 3.2. 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 3.2: 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

3.3.5 Number of Transform layers

The Transform network uses L fully connected layers, where L is a hyperparamter. In Figure 3.5, we plot the MSE of TransNet on the `Yelp17` dataset when varying L from 1 to 10. It can be seen from the figure that TransNets are quite robust to the choice of L, fluctuating only narrowly in its performance. Using only one layer gives the highest MSE, most probably because it doesn't have

enough parameters to learn how to transform the input. Using 10 layers also gives a high MSE, probably because it overfits or because it has too many parameters to learn. From the figure, using 2 or 5 layers gives the best MSE for this particular setting of TransNets. It is known that a 2 layer non-linear neural network is sufficient to represent all the logic gates including the XOR [49]. So, using 2 layers seems like a reasonable choice.

3.4 Explanation as Selection of the Most Similar Reviews

Our primary evaluation of TransNet is 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 $x_{CQ} = \Gamma_O(rev_{CQ})$. 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 3.4. 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. We also evaluate TransNet's selection of reviews quantitatively, with the help of human judges (*turkers*) on Amazon Mechanical Turk[7]. We use a pairwise setup that is described in detail in Section 4.7.4, where human judges are shown two test reviews side-by-side along with the original review written by a user for an item. After tagging liked (disliked) aspects in each of the reviews as well as any reason that the user has provided for their sentiment towards that particular aspect, the turkers were asked to select the test review that was most similar to the original review based on their tags.

We compare the TransNet method of review selection to a Matrix Factorization (`MF`) baseline as well as a random review selector baseline (`Random`). The MF baseline is the same competitive baseline used earlier in Section 3.3.3 to compare the rating prediction performance. We first calculate latent

[7] https://www.mturk.com

Dataset		TransNet > Random	TransNet > MF
Yelp17	likes	38.30 %†	33.33 %†
	dislikes	44.83 %	43.47 %

Table 3.3: Human Evaluation of Review Selection († denotes results that are statistically significant with *p-value* < 0.05)

representations of users using a neural implementation of matrix factorization. Subsequently, we find k neighbors of each of the users according to the cosine similarity computed on their latent representations. k is set to 40 in our experiments. For a test user-item pair, we find the most similar neighbor of the test user who has also written a review for that item. The selected review for a test user is that neighbor's review for that item. Even with 40 neighbors per user, only 2.7% of the test entries had any neighbor who rated the same item. For our comparison, we used only the valid test entries of MF. The Random baseline simply selects a random review for that item from the training set. Table 3.3 gives the percentage of instances where TransNet's review selection was judged to be strictly better than the corresponding baseline in those cases, where one of the competing methods was chosen. Statistical Significance is calculated using McNemar's Chi-Square test [105].

Contrary to the qualitative analysis and the rating prediction performance measured earlier, compared to the MF and Random selector baselines, TransNet's selection of reviews was found to be matching the original review better only in approximately 30% to 45% of the cases.

3.5 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. In addition, we showed how TransNets can be used to select reviews that are similar to the target user's review for the target item. Such reviews could be shown as an explanation for recommendations shown to the user.

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 3.4: 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 (see example pic) . 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 (see pic) one thing i wish they had more often is the spam and egg onigiri (see pic) . 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 3.5: Example of predicted similar reviews (continued from previous page)

Chapter 4

User Review Generation

Personalized Recommender Systems need to not only produce good recommendations that suit the taste of each user but also provide an explanation that shows why a recommendation would be interesting or useful to the user, to be more effective [154]. Such explanations would need to show the aspects of the item being recommended that the user would like or dislike, as well as a reason why they may or may not like it. The reason is consequential because different people perceive the same aspect differently according to their personal preferences and beliefs. For example, one user is happy that a restaurant is cheap and affordable, and therefore will like it, whereas another user is unhappy that it is cheap, worrying that it's of lower quality, and therefore does not like it. Many times, users also care about certain aspects of an item without having any explicit sentiment. For example, it would be good for a user to know beforehand if the parking is limited or if a movie theatre has an attached food court. These are usually purely for informational purposes and do not necessarily affect their sentiment towards the item itself. Such aspects, which we will refer to as 'neutral aspects', of the item are also important because it helps the user plan ahead. For example, they could plan to arrive early. Neutral aspects also need to be personalized because not everyone cares about parking especially if one regularly uses public transport. User's preferences are undeniably evident in the reviews that they have written in the past. This is true for aspects that are liked and disliked as well as neutral. If a user has mentioned about parking in many reviews in the past, it is obvious that they would like to know about the parking situation at a place that they are planning to visit.

Providing explanations is important for recommendations and is gaining popularity in the recommendation community [154]. The main obstacle in training a model to generate explanations is the lack of availability of training data. There is no public dataset available thus far that provides a ground-truth explanation that would be acceptable to a user for recommending an item. However, users' reviews for items are available in abundance. Using such review data, it is possible to train models that can predict or generate a review for a given user-item pair. Showing a user what their review would look like for an item if they were to try it is an important step towards providing a detailed personalized explanation. The predicted review can be shown to the user while recommending an item, in place of an explanation. To motivate this argument, consider an example scenario where we know that Alice wrote the following review after watching the movie, Inferno: "*I enjoyed the Inferno film for the most part as you are 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 disappointed by the changed ending. The book's solution was challenging but elegant; the film is clunky and predictable." If we knew that Alice would be writing this review, then in hindsight, a good explanation for why Alice may or may not like Inferno could be constructed from the review, like so: *"You will most likely enjoy the Inferno film for the most part as I'm very fond of Tom Hanks as Robert Langdon. You will most likely be disappointed by the changed ending. The book's solution was challenging but elegant; the film is clunky and predictable."* To construct such an explanation, we changed the first person past tense of some of the sentences in the original review to second person future tense. i.e. showing a user what they may think about an item if they were to use or experience it could serve as a type of explanation. In Chapter 5, we construct such a dataset of detailed explainable recommendations by formulating it as a review-rewriting task on Amazon Mechanical Turk. In the task, turkers are shown the original review written by a user for an item and asked to rewrite it in the form of a detailed recommendation by changing the first person past tense of the sentences into a second person future tense.

In the past, user reviews have been constructed by extractive techniques that select sentences from other reviews and concatenate them together. Some of those techniques are personalized to the user and others are standard extractive summarization techniques that are not specific to the user. A discussion of such methods is in the Related Works Section 4.1. One of the main limitations of extractive methods is that the resulting review does not sound natural. This is expected because stitching together sentences from different reviews takes them out of their contexts and therefore, does not jell well.

Another set of methods proposed in the past attempt to provide explanations by selecting reviews written by other users for the same item. This was also the case in the method proposed in the previous chapter. By selecting the whole review, the review sounds natural and does not suffer from the problems exhibited by the extractive approaches. However, there are obvious drawbacks to this approach. Almost always, each user's overall experience with an 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 4.1. 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 Charlie, 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 Dave's review, but also additional aspects not mentioned by the latter.

The above example is a typical scenario where different users care about different aspects of the same item. Explanation by review selection suffers from the disadvantage that it is almost always impossible to find one review that would discuss all aspects of the item that the user at hand would like to know about. 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.

In this chapter, our goal is to generate reviews using abstractive methods. Recurrent Neural Networks (RNN)s have been shown to be very good at modeling language [188]. Once trained, they can generate samples from the language model. Controlling the content that is generated is usually achieved by providing a context vector to the RNN. A recent work called Collaborative Filtering with Generative Concatenative Networks (CF-GCN) [116] proposed to concatenate latent embeddings of a user-item pair to the input in each timestep of the RNN in an effort to force it to generate their joint

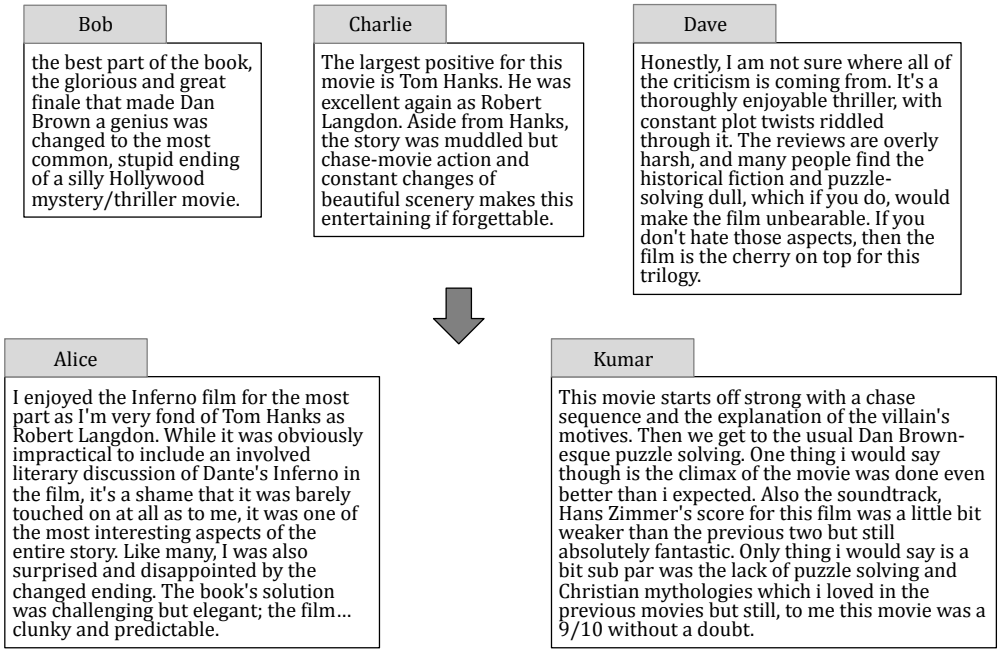

Figure 4.1: Sample Movie Reviews

review. Albeit a simple strategy, the authors showed that by providing the user-item information to the RNN at each step, they could achieve very low perplexity scores. Yet another recent work [169] proposed a review generator model that uses a Memory Network to refine and customize an item's representation before using it to produce the rating and generate a review. A discussion of other review generation methods is in the Related Works Section 4.1. An important consideration for neural models that generate natural language text is the availability of copious amounts of training data. In this chapter, we show how the Transformational Neural Network (TransNet) architecture that was proposed in the previous Chapter 3 for the task of rating prediction can be used for review generation. By introducing an additional latent layer that represents the review of a user for an item, which at training time is regularized to be similar to the actual review's latent representation, we showed that it could improve the rating prediction performance substantially. In this chapter, we show that by using such an intermediate step and training a review generator *indirectly*, we are able to train a model that is especially good at generating reviews when the training data is scarce, and insufficient to train the state-of-the-art review generators that train their models *directly*.

The rest of this chapter is organized as follows. In Section 4.1, we discuss the past research that is relevant to explanation using reviews. In Section 4.2, we describe how the TransNet architecture could be adapted for review generation. In addition to this standard version, there are various choices for modifying the TransNet architecture for text generation, some of which are known to be better choices in other text generation settings. Therefore, it is imaginable that such choices should lead to a better model. After introducing the datasets, evaluation metrics, and experiment settings in Sections 4.3, 4.4 and 4.5 respectively, in Section 4.6, we describe different variations to the standard TransNet model and study their effect on the performance empirically. In Section 4.7 we perform

extensive experiments to compare the performance of TransNet to state-of-the-art methods for review generation. In addition to automatic evaluation, we also use human judgements on Amazon Mechanical Turk to test their performance. A preliminary version of this work was published in the 6^{th} International Conference on Learning Representations (ICLR '18) Workshop [25].

4.1 Related Work: Explanation using Reviews

4.1.1 Non-Personalized Summarization

A multitude of models have been proposed in the past that can create a generic summary of reviews. [135] 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 [163], which uses an attention based neural network to generate one-sentence summaries from multiple opinions. In [52], the authors incorporate a copy mechanism into a `seq2seq` learning model to improve text summarization task. [47] is an abstractive graph-based technique that constructs summaries from highly redundant opinions.

[117] 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, [114] uses side information to construct news headlines from the text of the article. [30] is another model for the same task, but uses a hierarchical reader and a sentence level attention extractor for selecting informative sentences. [142] 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 [113] has been shown to be comparable or better than the state-of-the-art algorithms for extractive summarization. [182] is a graph based neural model for the same task.

[16] 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 [181]. The reviews generated by both these models were evaluated by human judges and found to be indistinguishable from authentic reviews.

4.1.2 Extractive / Non-Neural Models for Personalized Review Generation

[128] 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 [22]. [190] 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. [56] is another method that ranks aspects for each user on a tripartite graph consisting of the users, items and aspects. In [172], 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 [41], the authors proposed a user specific aspect sentiment model to predict ratings where they evaluated the reviews using only perplexity. Although [101] 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. [110] 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.

4.1.3 Abstractive / Neural Models for Personalized Review Generation

[116] proposed a character-level model called Collaborative Filtering with Generative Concatenative Net (CF-GCN), which is a regular character-level LSTM with additional information concatenated to its input. The additional information includes that of the user, item and the category of the item. It is one of the most recent state-of-the-art models for review generation. Since it is one of our competitive baselines, we describe it in detail later in Section 4.7.1. [33] is a minor extension to [116], where ratings of individual aspects of the item are also concatenated to the auxiliary information. [151] is a similar work that produces the text of the review from the item id and sentiment/rating as input. [169] 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. The authors showed that the method is capable of producing reviews with better ROUGE scores than competitive summarization algorithms. This method is yet another recent state-of-the-art model and is one of our competitive baselines. We describe it in detail later in Section 4.7.2. [83] 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 Student network of the model proposed in this chapter. In [195], the authors proposed generating user reviews by using an attention mechanism over the attributes (metadata) of items as well as the user rating for that item. In our setting, ratings are not assumed to be known beforehand. Having access to attributes would certainly improve the performance of most methods considered in this thesis.

4.1.4 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.

4.1.4.1 Text Generation from Structured Data

In [80], 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. [139] proposed an improvement of this model which also considers the order of facts in the table. [5] is another method for a similar task which combines information from a knowledge graph of facts with a neural language model while generating text. [54] is a simpler model which poses this as a `seq2seq` problem by flattening the table to a sequence of words. The model proposed in [106], is similar, except they introduce an additional aligning step before the decoding stage.

4.1.4.2 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. [159, 160] 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 [179]. This is further improved by using a sentinel for non-visual words in [93]. In [91], the authors train the model to explicitly learn attention maps from alignment information. In [180], 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.

4.1.4.3 Autoencoders

In one of the networks of the proposed model, the input and the output are the same, making it an Autoencoder. In [82], 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 [45] proposed a similar model which encodes using a CNN and decodes using a hierarchical LSTM. An earlier model proposed in [36] auto-encoded sentences.

4.2 TransNets for Review Generation

In this Section, we show how the Oracle-Student architecture called ***Trans****formational Neural* ***Net****work* (TransNet) that we proposed in Chapter 3 for the task of rating prediction can be used for review generation as well. By introducing an additional latent layer that represents the review of a user for an item, which at training time is regularized to be similar to the actual review's latent representation, it could improve the rating prediction performance substantially. Such an intermediate step dramatically reduces the training time of the review generator for large datasets and the improves its performance especially at low data settings.

In our architecture for review generation, there are two networks as shown in Figure 4.2. An *Oracle Network* processes the text of $user_A$'s review for $item_B$, denoted as rev_{AB}, using a text processor Γ_O, which embeds the words and extracts sentence or phrase level features, to construct a latent representation for the review. This latent vector is then concatenated to each step of a decoder LSTM to reconstruct rev_{AB}. Since the input and the output of the Oracle network are the same, it is an autoencoder. By training the Oracle in this manner, it learns to generate a good encoding of the text to minimize the reconstruction error. Also, this network achieves a low perplexity without much difficulty because it is working with the actual review.

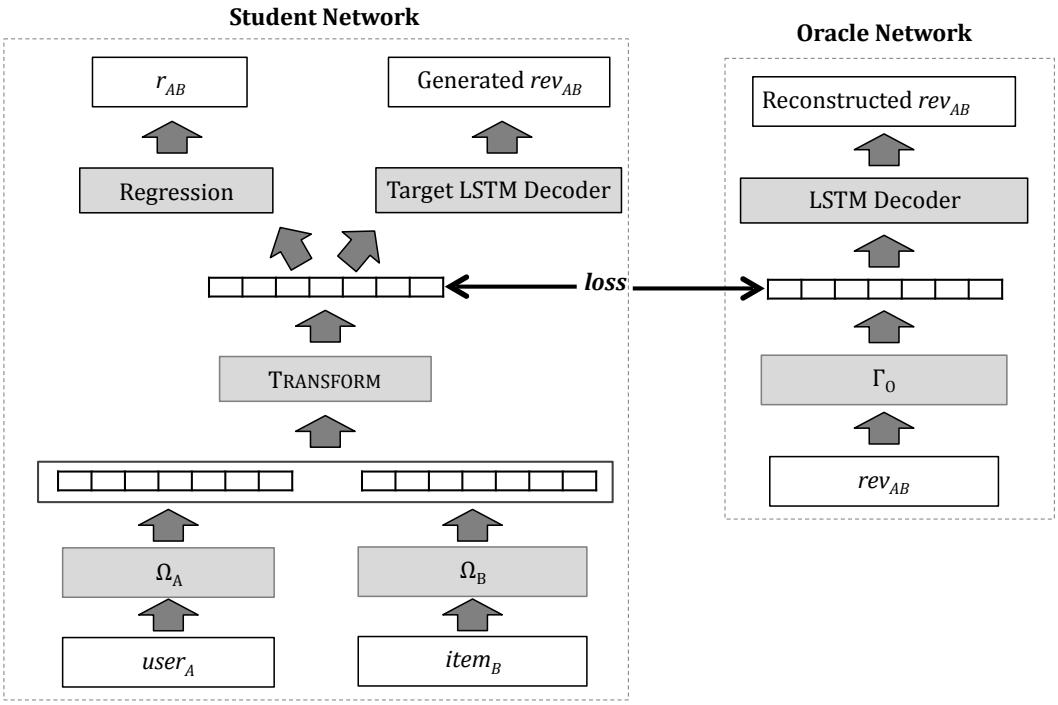

Figure 4.2: TransNet Architecture adapted for Review Generation

The proposed architecture has a second network called *Student Network* that first embeds $user_A$ and $item_B$ using embedding operations, Ω_A and Ω_B, and subsequently, learns how to transform the two latent representations into that of their joint review, rev_{AB}. The embedding operation could be as simple as a matrix lookup, or it could be the process of constructing the user/item representation from their historical reviews as described in Section 3.2. Learning to transform is achieved by using a Transform layer, which is a L-layer deep non-linear fully connected feed forward network. During training, we will force this layer to produce a representation that approximates the Oracle Network's encoding for rev_{AB}, by minimizing the L_2 loss computed between the two latent representations. The intuition behind the transformation step is that, while information about the user and the item for which the review is being generated is important, the information about the outcome of their interaction would be more informative. If we knew the aspects of $item_B$ that $user_A$ liked and those that they disliked, we could produce a review that $user_A$ would write for $item_B$, with high fidelity. The transformed representation is also used by a regression layer to predict $user_A$'s rating for $item_B$.

There are two obvious architectural variants for the Student's decoder network: Either it has a decoder of its own as shown in Figure 4.2 that is trained alongside the rest of the Student network, or, it reuses the Oracle's decoder network by plugging in the approximate representation that it constructed for the user's review for the item. The first option of training a separate decoder is useful if the Student wants to decode using a different input than that used by the Oracle. However, if the inputs are logically the same, i.e. they represent the joint interaction, it does not warrant a separate decoder, and the Student can simply reuse the Oracle's decoder. We experimented with both approaches.

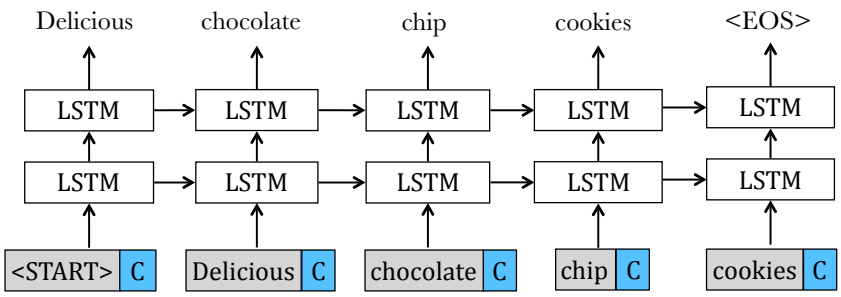

Figure 4.3: TransNet Decoder for Review Generation

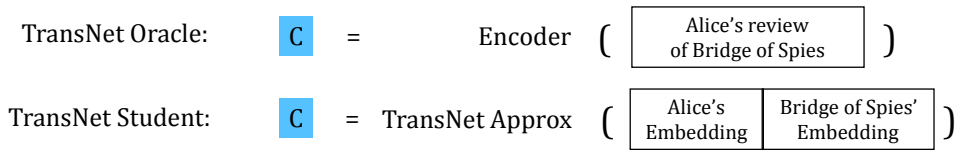

Figure 4.4: TransNet Decoder Contexts

At test time, for a test user-item pair, $user_P$ and $item_Q$, the Student Network is used to construct an approximate representation of their joint interaction, which is fed as a context vector to the decoder LSTM that was trained by the Oracle Network, to generate their joint review.

4.2.1 Encoder

The Oracle network uses an encoder to convert the text of a review to its latent representation. Ideally, this process should be able to capture the overall sentiment expressed by the reviewer about the item, specific aspects of the item that the user mentioned, sentiments expressed for those aspects along with reasons if any, word and phrase usages characteristic of the reviewer etc., and be able to represent them in a distributed vector representation. This context vector is the only information available to the decoder while re-generating the review. Such a constraint is needed because while it is possible to let the Oracle's decoder attend to the actual text of the review when decoding, there is no text to attend to when decoding with Student's context — the latter is created from user's and item's latent vectors.

For the standard version of TransNet for review generation, we will use the CNN-based encoder described in Section 3.2.2. However, there are other types of encoders that could be used in the Oracle's network. Performance comparison of different encoders and a discussion is in Section 4.6.2.

4.2.2 Decoder with Context

A review is an outcome of a user's interaction with an item. Therefore, it is influenced by both the user and the item. Review generation is a type of controlled text generation task, where the generated text is pertinent to the item being reviewed and matches the idiosyncrasies of the user.

Recurrent Neural Networks (RNNs) have been shown to be very good at modeling language [188]. Once trained, they can generate samples from the language model. Controlling the text that is generated can be achieved by providing a context vector to the RNN [116]. Figure 4.3 shows an example of such an unrolled LSTM network. The context vector `C` controls the overall content while the word generated at each step is influenced by the word generated at its prior step. `C` could be provided as an input to the first hidden state of the LSTM or it could be concatenated to the input at each step. As noted in [116], by concatenating the context with the input at each step, the context signal is preserved through hundreds of steps, allowing long and coherent sequences to be generated, whereas, treating the context as input to the hidden cell, the signal will quickly vanish or explode. Therefore, all decoders used in this Thesis concatenate the context to the input at each step.

The input at the first step to the decoder is a special start symbol `<START>`. The network is unrolled until it encounters an end-of-sequence symbol `<EOS>`, or reaches the maximum number of steps. In the case of TransNet, there are two kinds of context vectors — one that is created by the Oracle from the text of the actual review rev_{AB}, and the other created by the Student using the Transform operation on the latent representations of $user_A$ and $item_B$. This scenario is depicted in Figure 4.4. The decoder network of Figure 4.3 is fairly independent from the rest of the TransNet modules — it can be thought of as a black box language generator that produces text relevant to the supplied context vector. In the rest of this Thesis, unless specified otherwise, the Student network reuses Oracle network's decoder for generating the review, by providing its own context vector.

4.3 Datasets

We will evaluate the performance of the models on seven large datasets and their variants. The first two, `Yelp17` and `Yelp18`, are from the Yelp dataset challenges[1] released in the years 2017 and 2018 respectively, containing 4M to 5M reviews and ratings of businesses by more than 1M users. The next three, `AZ-Books`, `AZ-Elec` and `AZ-CSJ` contain reviews and ratings given by users for products purchased on `amazon.com`, specifically, books, electronics, and clothing, shoes and jewelry respectively. These datasets are some of the larger ones in the latest release of the Stanford SNAP dataset collection[2] [102, 103]. The next two, which are also from the SNAP collection [101, 104], `BeerAdv` and `RateBeer`, are beer rating datasets constructed from online beer reviewing websites `beeradvocate.com` and `ratebeer.com` respectively. Following [116], for each user, one rating/review each are randomly chosen for the validation and test sets, and all other ratings/reviews are used for training (leave-one-out evaluation).

For some of these datasets, we also constructed variants as follows:

1. `AllUsers`: An `AllUsers` version of a dataset includes all the users of the original dataset. Therefore, it is essentially the full dataset. Users who have written less than 3 reviews are included only in the training set, i.e. by extension, all users in the validation and test have appeared at least once in train.
2. `100K`: A `100K` version of a dataset is a random sample of 100,000 user-item pairs from the dataset that is then split into train, validation and test datasets according to the leave-one-out

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

Dataset	#Users	#Items	#Ratings & Reviews	train	validation & test
Yelp17	1,029,432	144,072	4,153,150		
Yelp17 AllUsers	1,029,432	144,072	4,153,150	3,512,186	320,482
Yelp17 100K	77,427	43,272	100,000	91, 666	4,167
Yelp18	1,326,101	174,567	5,261,669		
Yelp18 25-core	11,796	9,904	652,746	629,154	11,796
Yelp18 40-core	1,002	978	72,469	70,465	1,002

Table 4.1: Yelp Dataset Statistics

Dataset	Category	#Users	#Items	#Ratings & Reviews
AZ-Books	Books	8,025,457	2,329,957	22,504,600
AZ-Books 25-core		19,803	22,084	1,273,564
AZ-Elec	Electronics	4,200,520	475,910	7,824,482
AZ-Elec 15-core		842	846	22,649
AZ-CSJ	Clothing, Shoes	3,116,944	1,135,948	5,748,920
AZ-CSJ 5-core	and Jewelry	39,357	23,014	278,457

Table 4.2: Amazon Dataset Statistics

procedure described above. However, like the AllUsers version, users who have written less than 3 reviews are included only in the training set.

3. k-core: A k-core version of a dataset is its subset such that each user in the subset has written reviews of at least k items in that subset and each item in the subset has reviews of it written by at least k users in the subset. If the original dataset is long-tailed or sparse, its k-core version is necessarily denser. For some of the datasets like Yelp18 and AZ-*, the largest possible k is in the low 20s to 50s, whereas for the beer datasets, it is easy to get a large k of 150 or above.
4. k-core-n: A k-core-n version of a dataset is a subset of a k-core version of that dataset such that for each user in the training set, exactly n ratings/reviews are randomly chosen from their training reviews ($k \geq n$).

Each of the above versions serve a different purpose — they help us understand the performance of various methods under different conditions. For example, the 100K version is quite small compared to the original and the AllUsers version, and equally or more sparse. Therefore, it helps us study the performance of the algorithms in low and sparse data settings. In comparison, the k-core versions are quite dense compared to the original dataset. That helps us test the performance in dense settings. The k-core-n for small n, in comparison, could be tiny. The statistics of the datasets are given in Tables 4.1, 4.2 and 4.3.

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 specific aspects of the beer, the chance for variance is reduced. For example, in BeerAdv, most reviews discuss the following aspects of the beer

Dataset	#Users	#Items	#Ratings & Reviews	train	validation	test
BeerAdv	33,387	66,051	1,586,259			
BeerAdv AllUsers	33,387	66,051	1,586,259	1,548,498	18,884	18,883
BeerAdv 100K	12,383	18,868	101,769	90, 524	5,618	5,627
BeerAdv 150-core	1,601	1,615	542,078	538,824	1,626	1,628
BeerAdv 150-core-50	1,601	1,615	84,410	81,164	1,623	1,623
BeerAdv 150-core-25	1,601	1,615	43,844	40,609	1,618	1,617
BeerAdv 150-core-5	1,601	1,615	11,347	8,101	1,622	1,624
RateBeer	40,213	110,419	2,924,127			
RateBeer 150-core	2,335	3,111	1,272,371	1,267,621	2,369	2,381

Table 4.3: Beer Dataset Statistics

in order: appearance, smell, taste, mouthfeel and drinkability, although each user has their own way of describing the beer [116]. 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 Yelp18 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 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 regular structure. Therefore, we expect most of the models to be better at generating the beer reviews than those from the Yelp and

Amazon datasets.

4.4 Evaluation Metrics

To measure the performance of the models, we will use the metrics given below. Two of the metrics, Word-Level Perplexity and Mean Squared Error do not require the models to generate any review texts. Also, for these two metrics, lower values usually indicate better models. These metrics are popular in the text generation community for measuring the progress of model training as well as comparing different models [116, 188]. Next three metrics are evaluated on the generated text compared to the original review written by the target user for the target item. These metrics are more popular for measuring the performance of translation and summarization techniques [40, 87, 122]. For these three metrics, higher values are better. We observed in our experiments that the above automatic evaluation metrics could be misleading in some cases. At times, the best generative model according to these metrics can produce the same or extremely similar reviews for all user-item pairs. Unless manually checked, such occurrences could go unnoticed. To quantify this observation, in addition to the above five metrics, we introduce an additional metric, called *Input Sensitivity* (IS), that measures the responsiveness of a model towards its inputs i.e. the user and the item information.

1. Word-Level Perplexity (WP): The negative log likelihood of the ground-truth review text. It measures how likely the text is under the trained model.
2. Mean Squared Error (MSE) as described in Section 3.3.4.
3. BiLingual Evaluation Understudy (BLEU) [122] 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.
4. Metric for Evaluation of Translation with Explicit ORdering (METEOR) [40] 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.
5. Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [87] 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.
6. *Input Sensitivity* (IS) is a new metric for measuring the capacity of a model for generating a variety of reviews as its inputs are varied. It is computed as $\frac{n_{rev}}{n_{in}}\%$, where, n_{rev} is the total number of unique reviews produced by the model during testing and n_{in} is the total number of unique user-item pairs tested.

4.5 Experimental Settings

All algorithms, including the variants and competitive baselines, were implemented in Python using TensorFlow[1], an open source software library for numerical computation, and were trained/tested

on NVIDIA GeForce GTX TITAN X GPUs. After each epoch of training (one pass over the training data), the performance of the models are tested on the validation and the test data. If either the MSE or the Word Perplexity has improved on the validation data compared to the previous epoch, the model parameters are saved. Even if there is no improvement, the training is still continued up to a maximum number of training epochs. The reported values are the test scores where the corresponding validation scores are the best.

The implementation of the standard version of TransNet uses the same CNN encoder for the Oracle as the one used earlier for rating prediction as described in Section 3.3.2. The vocabulary is obtained by selecting the most frequent 20,000 words in the training set. The words are not lowercased or stemmed. Stopwords and punctuations are not filtered out. The words are embedded into a 650D space using an embedding matrix, which is learned directly during the training phase. The encoder produces a 100D representation for the review text.

In the Student network, both users and items are embedded into a 50D latent space using an embedding matrix, which is learned during the training phase. The Transform module is 2 levels deep, and transforms the user and items embeddings into a 100D latent space, to correspond to the output of the Oracle's encoder. The Factorization Machine regression layer uses the same settings as the one used earlier for rating prediction as described in Section 3.3.2.

The base language model (decoder) is TensorFlow's benchmark implementation[3] of LSTMs regularized using dropout [188], which was shown to achieve very good results for language modeling tasks. Unless stated otherwise, we use their recommended `medium` configuration: 2 layers of stacked LSTMs with 650 hidden units. However, since we attach the transformed user and item representations to each step, the number of hidden units are increased by 100 units to a total of 750. We use 100 timesteps instead of their 35, and a mini-batch size of 256. The full network is trained with a learning rate of 1.0 for 6 epochs followed by a decay of 0.8 for a total of 40 epochs. The full network is regularized using dropouts with a 0.5 keep probability.

Some experiments also use a `small` configuration, which corresponds to a simpler decoder. The hidden size is only 150 units and the words are embedded into a 50D latent space. The rest of the settings are the same as the `medium` configuration above.

4.6 TransNet Variants

The standard version of TransNet introduced in Section 4.2 can be enhanced in a number of ways. For example, the CNN encoder could be replaced with an RNN encoder like a Bi-LSTM network. The latter is known to substantially improve the performance of models in various natural language related tasks. Similarly, Adversarial training is known to improve the output of image generation models. In this Section, we will discuss different variations to the standard TransNet architecture that modifies the Oracle-Student architecture wholly or partly, and how such changes affect the performance. The modifications are inspired by their success in other related tasks.

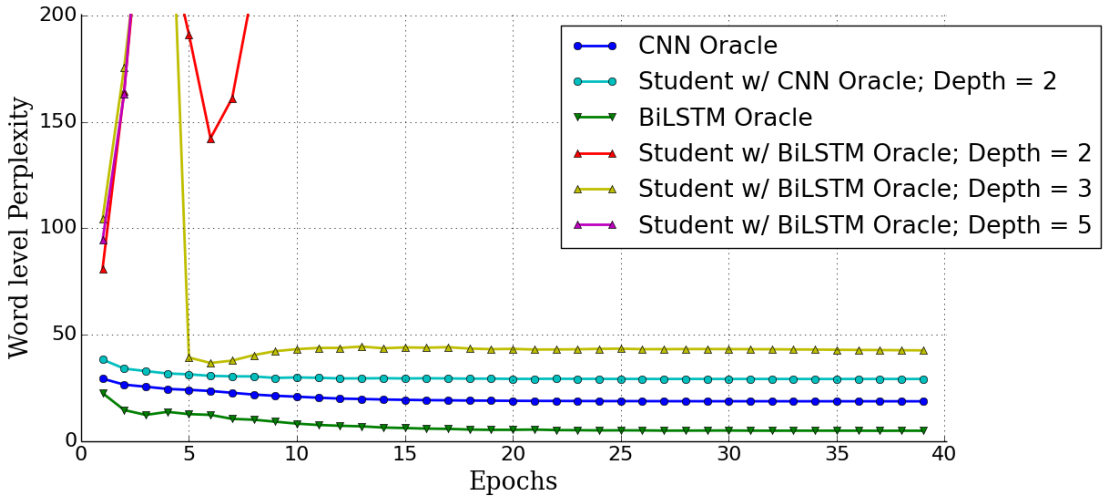

Figure 4.5: Student's performance with different Oracles

4.6.1 Oracle's Encoder: CNN vs. BiLSTM

In this Subsection, we will consider various encoders for the Oracle and study how that affects the performance of the Student. Note that when the Oracle network is modeling the output, its input is the original review text. Therefore, the Oracle's performance corresponds to the best case scenario for predicting the output. Since the Student network is trying to mimic the Oracle, its performance is upper bound on what the Oracle can achieve.

The standard version of TransNet uses a CNN encoder to convert the review text to its latent representation. However, there have been many results in the past that showed that a sequential encoder like LSTM or GRU is usually better at encoding natural language text. Therefore, in this Subsection, we test the performance of a variant of TransNet that uses a Bidirectional LSTM (BiLSTM) [51] encoder for the actual review text used by the Oracle Network.

We plot the Word Level Perplexity of the standard TransNet with a CNN encoder along with that of a variant that uses a BiLSTM encoder in Figure 4.5. The dataset is `BeerAdv 150-core`. In the Figure, the blue line is the test score of the CNN Oracle, which gives a best score of 18.633. That is worse than that of the BiLSTM Oracle, plotted in green, which gives a best score of 4.761. Therefore, it is evident that encoding with a BiLSTM network preserves significantly more information than encoding with a CNN network, a result that is as expected.

However, an Oracle is only as good as the performance of the Student that it can help train. In the Figure, the CNN Oracle's Student, plotted in cyan, is able to achieve a perplexity score of 29.131, closely mimicking that of the Oracle very early on. However, the BiLSTM Oracle's Student, plotted in red, which has the same architecture as the CNN Oracle's Student — specifically, both have 2 layers in Transform — is unable to learn how to produce the encoding constructed by the Oracle network. The perplexity first diverges to a very large value before improving slightly after the 5^{th} epoch, only to diverge once again. It does not converge at all in the 40 epochs that we trained and tested its performance.

[3] `https://github.com/tensorflow/models/tree/master/tutorials/rnn/ptb`

In the Figure, we also plot the test performance of two more Student networks, one with 3 and the other with 5 Transform layers, that learn from a BiLSTM Oracle. The Student with 3 layers, plotted in yellow, diverges initially like the one with 2 layers above, only to converge after the 5^{th} epoch. It shows that a Student network with 3 layers of transformation can learn to construct the encoding that is independently constructed by a BiLSTM encoder to some extent. However, its best test perplexity is 36.653, which is much worse compared to its Oracle. It is also not better than the CNN Oracle's Student. The Student with 5 transform layers is plotted in magenta. As can be seen from the Figure, its training diverges immediately to very large values.

The above experiment shows that a CNN Oracle, although not as competent as a BiLSTM Oracle, is able to help train a Student network that achieves a better performance than that trained by the latter. It could be because of the specific architecture of the Transform module that we use for the Student, which is probably more capable of imitating the selection and pooling operations done by the Oracle's CNN encoder, but not the sequential operations performed by a BiLSTM encoder. It is possible that there are other Transform architectures that could construct a BiLSTM text encoding from user and item representations.

4.6.2 Joint Training of BiLSTM Oracle and Student

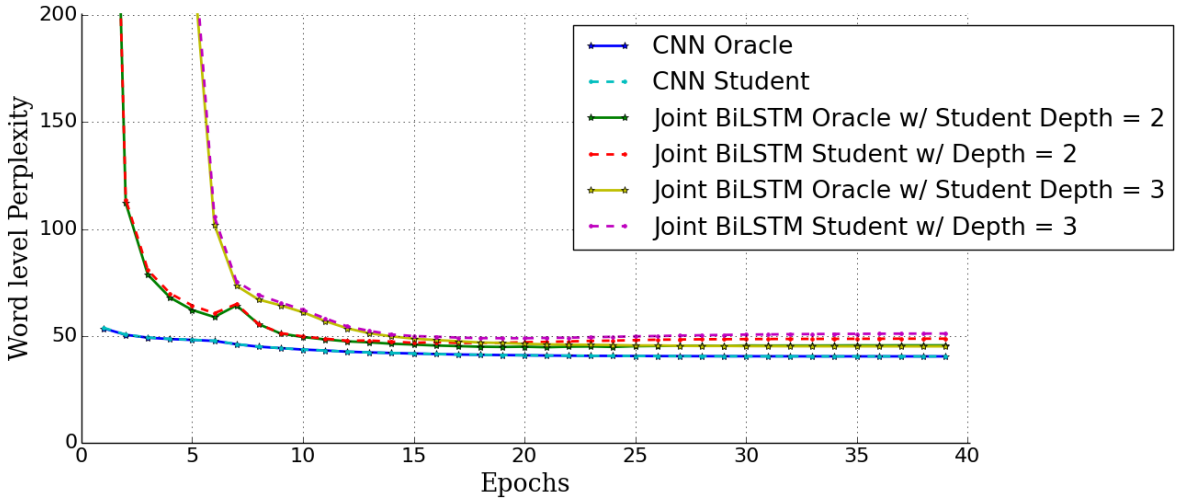

Figure 4.6: Performance with joint training of Oracle and Student

In the previous Subsection, we saw that an Oracle with a BiLSTM encoder is able to regenerate user's reviews with high fidelity that far exceeds the performance of a CNN encoder based Oracle. However, the BiLSTM Oracle's Student is unable to recreate its encoded representation. Therefore, in this Subsection, we will train both the Oracle and the Student jointly in an effort to coerce the Oracle to produce an encoding that the Student would be able to imitate. i.e. instead of training in two sub-steps, we will have only one step — the loss that is minimized is the sum of the losses incurred by the Oracle and the Student. Therefore, the parameter updates to the Oracle will involve an additional delta that is due to the difference in the encodings produced by the Oracle and the

Dataset	CNN Student with Depth = 2	Joint BiLSTM Student with Depth = 2	Joint BiLSTM Student with Depth = 3
BeerAdv 150-core	40.382	46.666	48.888
BeerAdv 100K	53.121	56.992	59.470
Yelp17 AllUsers	26.084	28.405	26.183
Yelp17 100K	61.606	65.113	69.931

Table 4.4: Word Perplexity with joint training of Oracle and Student

Student. By doing so, the Oracle is penalized additionally for producing an encoding that deviates substantially from that constructed by the Student.

In Figure 4.6, we plot the Word Perplexity of joint training BiLSTM Oracles with Students of 2 and 3 Transform layers on BeerAdv 150-core with a small configuration. As can be seen from the figure, the joint training has forced the Oracles and the Students to work together. The Student with 2 layers plotted in red closely mimics its Oracle plotted in green until about 15 epochs after which, the Oracle slightly improves its performance while the Student plateaus. A similar behavior is seen in the case of the Student with 3 layers plotted in magenta. It is close to its Oracle plotted in yellow till about 15 epochs. However, it is interesting to see that in the case of joint training, a 2 layer Student is better than a 3 layer Student, whereas in disjoint training, the opposite was true.

For comparison, in the Figure, we also plot a standard TransNet with a CNN Oracle in blue and its Student in cyan. They are not trained jointly. As can be seen from the Figure, the CNN Oracle-Student pair performs substantially better than the jointly trained BiLSTM Oracle-Student pairs. Also, it is interesting to note that the BiLSTM Oracle is no longer able to outperform the CNN Oracle because it is constrained by its Student when trained jointly.

In the Table 4.4, we compare the performance of these algorithms on three other datasets in addition to BeerAdv 150-core that was studied in Figure 4.6. The configuration is small as before. The best scores are highlighted in blue. As we can see from the Table, the Student that was not jointly trained with the CNN Oracle consistently outperforms both the Students that were jointly trained with a BiLSTM Oracle, on all the datasets. Another interesting observation is that, more data — i.e. in the case of the 150-core and AllUsers versions — does help close the gap between the joint training and the CNN Student compared to their performances with the corresponding 100K dataset versions. With large data as in the case of Yelp17 AllUsers that has about 3.5M training examples, we also see that the joint BiLSTM Student with a transform depth of 3 is able to surpass the performance of the joint BiLSTM Student with a depth of 2, and match the performance of the CNN Student. From the above results, we can infer that when only moderate amount of training data is available, a CNN Oracle-Student pair performs substantially better than a BiLSTM Oracle-Student pair trained jointly. As more training data becomes available, a BiLSTM Oracle can help train a Student with increasingly larger number of parameters, that could possibly surpass the performance of the standard CNN TransNet model.

Dataset	Student with standard CNN Oracle	Student with 2 layer encoder CNN Oracle
`BeerAdv 150-core`	40.382	43.446
`BeerAdv AllUsers`	45.358	45.248
`Yelp17 AllUsers`	26.084	25.713

Table 4.5: Word Perplexity with an Oracle that has a larger encoder

4.6.3 Enhancing the CNN Oracle

The standard version of TransNet that we used in the previous Subsections had a CNN Oracle with one layer of convolutions. In Figure 4.6, we saw that the Student is able to produce the encoding produced by the Oracle to a very close degree, and therefore, it is able to closely mimic the Oracle. The only limit to Student's performance is that of the Oracle. i.e. if we could improve the Oracle's performance further, it may be possible to further improve the Student as long as it is able to mimic the Oracle.

In this Subsection, we will add an additional layer of convolutions to the Oracle's encoder, effectively increasing the number of parameters in the encoder. Table 4.5 shows the word perplexity obtained by the standard and the enhanced versions on different datasets using `small` configuration. From the table, it can be seen that the standard CNN TransNet performs better on the `BeerAdv 150-core`. However, on the `AllUsers` versions of `BeerAdv` and `Yelp17`, which are about 5 times and 7 times larger than the `150-core` version respectively, the Student that learns from the Oracle with a 2 layer CNN encoder begins to surpass the performance of the Student trained in a standard setting. Therefore, we infer that with considerably larger amounts of training data, to the tune of a million or more, we can gradually increase the complexity of the Oracle for performance gains.

4.6.4 Extended TransNets for Review Generation

In Section 3.2.7, we proposed using the user's and item's identities along with the approximate review representation for predicting their joint rating. In most real world settings, the identities of the users and the items are known to the system. Therefore, it is a reasonable input data to use in predicting their ratings as well as reviews. This version of TransNet was called *Extended TransNet* (TransNet-Ext). It was shown in Section 3.3.4 that TransNet-Ext could outperform standard TransNet in most cases.

In this Subsection, we will study the performance of TransNet-Ext in generating reviews. The Student's architecture is changed to the extended version as in Figure 3.4. We also train a separate 2 layer LSTM Decoder for the Student instead of reusing the Oracle's decoder. The Student's decoder is larger than the Oracle's to accommodate the additional inputs.

In Figure 4.7, we plot the word level perplexity of TransNet-Ext on `BeerAdv 150-core` with `small` configuration. The performance of the standard TransNet is also plotted for comparison. The Oracles in both versions have the same architecture and parameter settings. As can be seen from the figure, although the TransNet-Ext Student initially starts at a worse performance, it does gradually improve its performance and begin to converge after about 20 epochs to come close to the standard

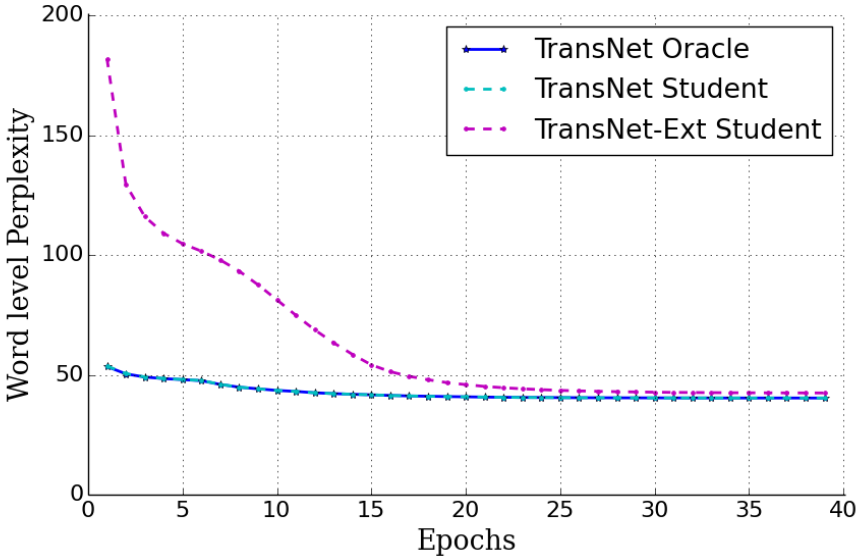

Figure 4.7: Performance of Extended TransNet

TransNet's performance. However, despite having more parameters and more inputs, TransNet-Ext is unable to beat the performance of standard TransNet. We hypothesize that this behavior is because the approximate review representation is constructed from the latent representations of the user and the item, and therefore, there is no new information in the additional inputs passed to the decoder. However, if we had instead constructed the approximate representations from historical reviews as was done earlier in Section 3.2.7, the user and item representations learned via matrix factorization would have carried additional information, and perhaps improved the performance. This experiment also shows that it is possible for the Student to train its own decoder if necessary instead of reusing the Oracle's decoder.

4.6.5 GAN style training

Generative Adversarial Networks (GAN) [50, 131] are comparatively a recent development in deep learning for training generative models, by an adversarial process. A GAN 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 [29, 50, 137]. A recent work, [86], used it for visual paragraph generation. To improve the quality of the generated paragraph, [86] 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.

To enable GAN training, we insert an adversarial network that takes as input, either the output of the Oracle's encoder or the output of the Student's transform layer, and predicts which output is from

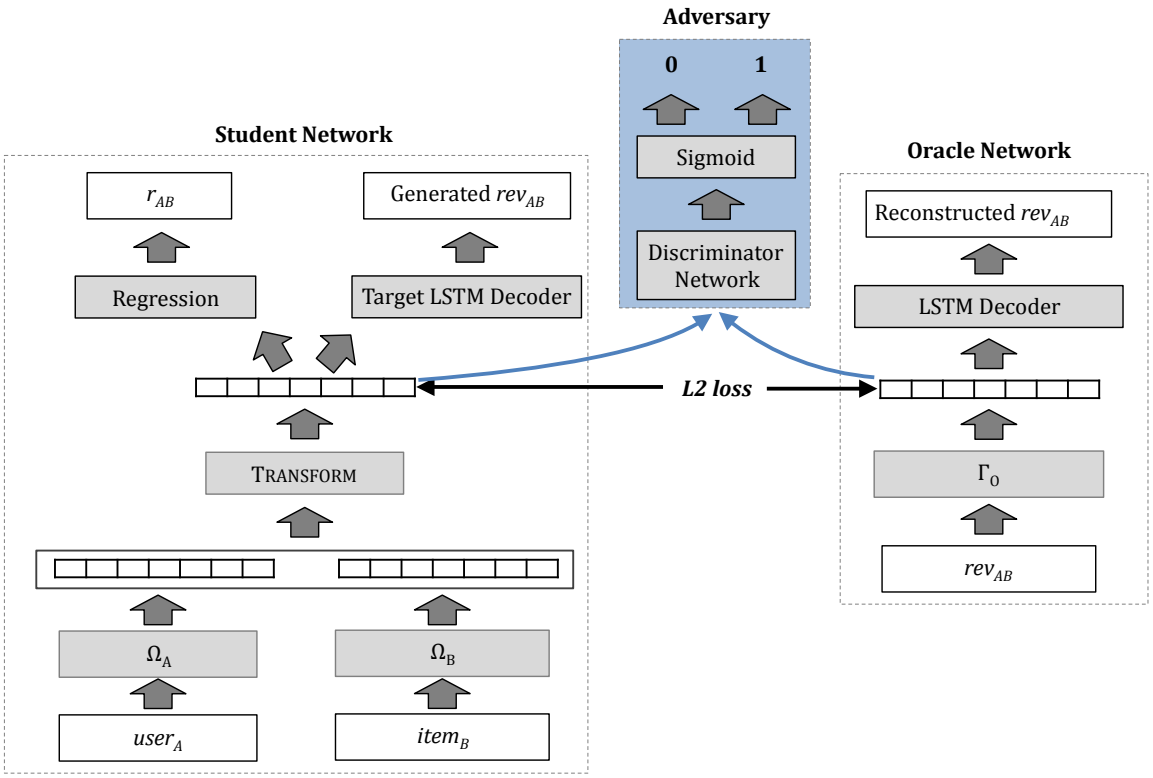

Figure 4.8: TransNet with GAN training: Architecture

which network. The true or real data is the output from the Oracle and the fake data is that from the Student. The adversary has l_A fully connected feed forward layers with $tanh$ non linearity. The last layer is a $sigmoid$ that gives the probability of the input being real or fake. In our experiments, we set l_A to 2. This modified architecture of TransNet is shown in Figure 4.8. The Student's transform network incurs a cost every time the adversary classifies its output as fake. This cost can be specified as $loss^{GAN}_{trans} = -log(A(z_S))$, where z_S is the Student's transform output and $A(.)$ is the adversarial function producing an output in the range $(0, 1)$.

We experiment with two types of GAN training. In the first setting, the Student's transform network is updated solely based on $loss^{GAN}_{trans}$. In the second setting, it is updated based on $loss^{GAN}_{trans}$ as well as the L_2 loss. The performance of the Student on `BeerAdv 150-core` with the two training approaches is plotted in Figure 4.9. For comparison, we also plot the standard TransNet setting that uses only the L_2 loss to update the Transform parameters. The configuration used in this experiment is `medium`. As can be seen from the figure, by using only the GAN loss for training (plotted in red), although we are able to train the Student, its performance converges to an unsatisfactory score. But adding in the additional L_2 loss term, we are able to substantially improve the performance. This setting is plotted in magenta. An adversarial training is known to be beneficial if we want to match the ground truth data's and the generated data's distributions. However, the way our training is set up, for each output produced by the Student, we know exactly what its true output should have been by accessing the Oracle's network. Therefore, instead of matching only the distributions, we can require the network to actually match the exact output. i.e. L_2 is a more assertive loss term that helps

push the Student network further towards Oracle's performance. From the figure, it is also clear that using only the L_2 loss (plotted in cyan) gives the best performance. Hence, we can infer that, in the context of L_2 training, additional GAN-style training is redundant, and could also adversely affect the performance.

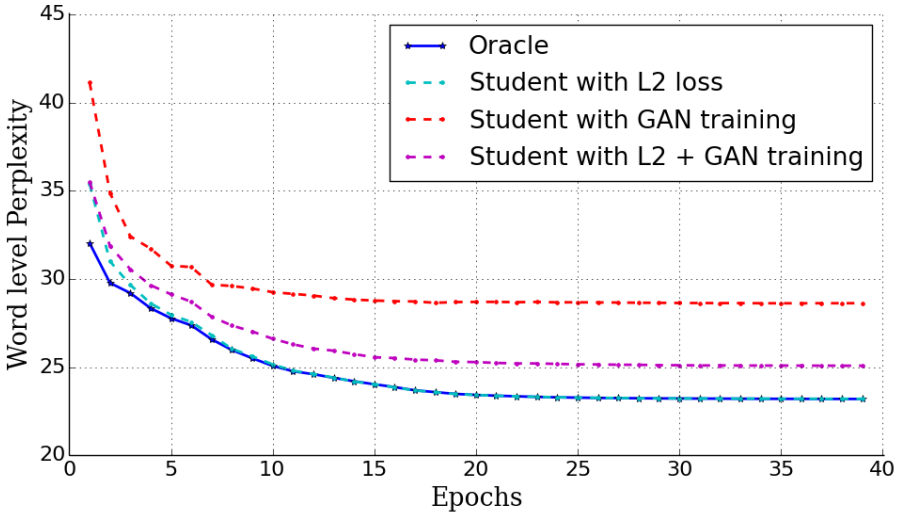

Figure 4.9: Student's performance with different Transform training approaches

4.6.6 Discussion

In the above subsections, we discussed five variations to the standard TransNet architecture. These changes were motivated by their success in other related tasks. However, contrary to expectations, for the task of review generation given user-item pair, the standard TransNet architecture performs the best in most cases or is on par with the alternate architecture in others. With large amounts of training data, certain variants may be able to achieve better performance.

4.7 Comparison to State-of-the-art Methods

In this Section, we will compare the performance of our TransNet architecture to two other recently proposed approaches for review generation. First, we discuss the architecture of the two approaches in the Subsections below, before delving into comparisons in the later Subsections.

4.7.1 Collaborative Filtering with Generative Concatenative Networks

In [116], the authors proposed a straightforward approach called Collaborative Filtering with Generative Concatenative Networks (CF-GCN) to train a review generator for a user-item pair. The users and the items are mapped to their latent representation using embedding matrices, Ω_A and Ω_B, that are learned during the training phase. For a user-item pair, their latent vectors are concatenated and passed as the context vector to a LSTM decoder. The context vector is concatenated to the input at

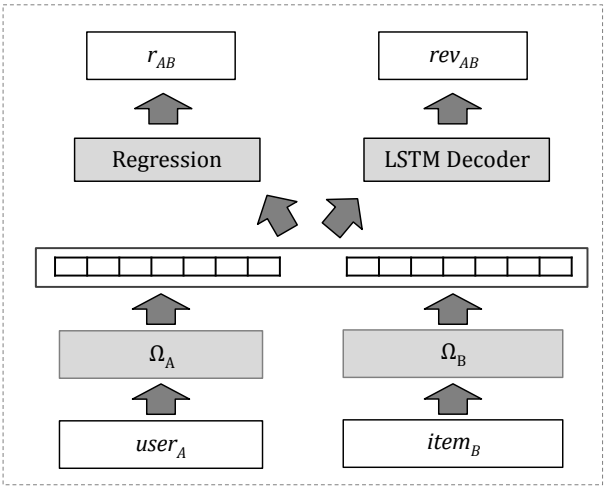

Figure 4.10: CF-GCN Architecture for Review Generation

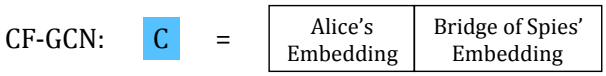

Figure 4.11: CF-GCN Decoder Context

each step of the decoder. That way, the user-item information/signal is preserved through hundreds of steps. The decoder is trained to generate the review, rev_{AB}, directly from $user_A$'s and $item_B$'s representations as the inputs. The context vector is also used by a regression module to predict the rating, r_{AB}. CF-GCN's architecture is depicted in Figure 4.10. Its context vector is constructed as shown in Figure 4.11.

We use an implementation of CF-GCN that is comparable to that of TransNet. Specifically, we make the following changes: CF-GCN as described in [116] is a character-level model. We use a word-level model with the same vocabulary as TransNet. Originally, CF-GCN predicts a boolean output of whether the user rates the item or not. Our implementation uses a Factorization Machine regression layer similar to TransNet for predicting the actual rating.

4.7.2 Opinion Recommendation Using A Neural Model

In [169], the authors proposed a review generator model that uses a Memory Network to refine and customize an item's representation before using it to produce the rating and generate a review. This model, which we call MemNet for short, has three subnetworks – a user model, an item model, and a neighborhood model. The architecture of their proposed approach is shown in Figure 4.12.

The user model converts a user into their latent representation using an embedding operation, Ω_A. This embedding operation could be as simple as a matrix lookup or may use the user's historical reviews. Next is the neighborhood model. Prior to training the MemNet model, the authors compute a matrix factorization on the rating matrix, and use that to find the most similar users, who they call

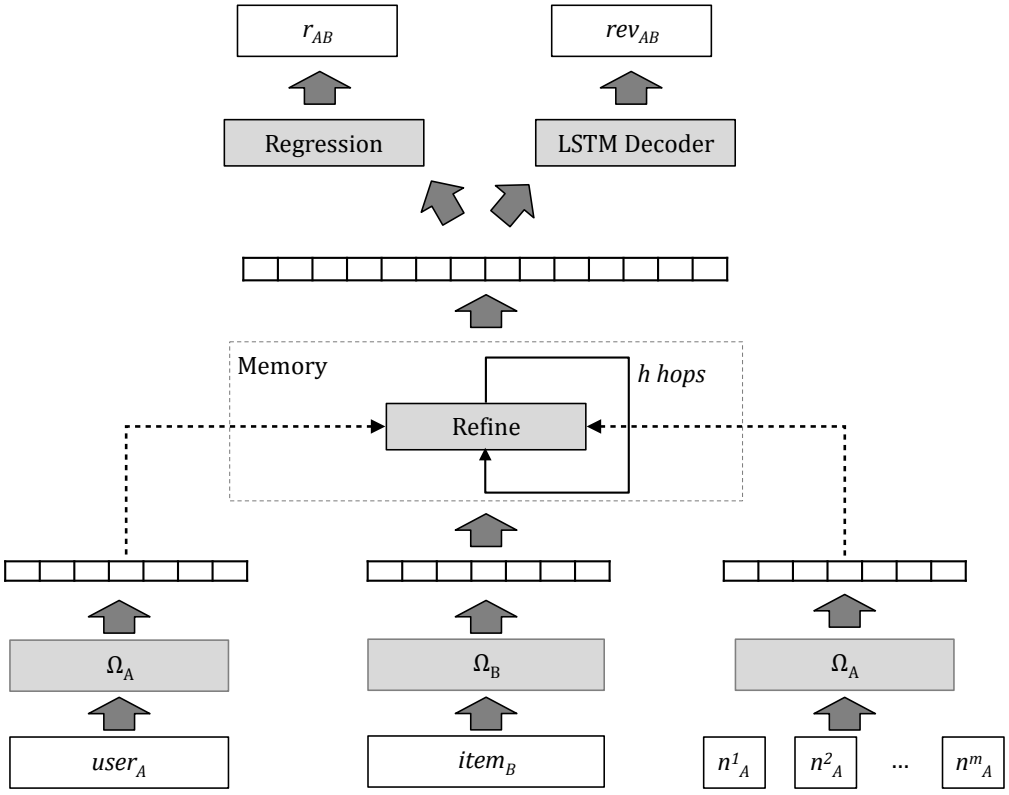

Figure 4.12: MemNet Architecture for Review Generation

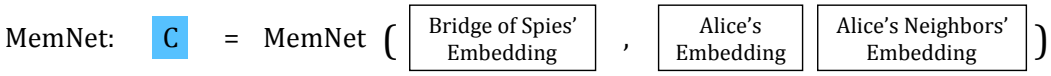

Figure 4.13: MemNet Decoder Context

as neighbors, for each of the users. In the neighborhood model, for $user_A$, retrieve their m precomputed neighbors, $n_A^1, n_A^2, ...n_A^m$, and embed them using the embedding operation Ω_A as earlier. The final output of this subnetwork is the average of the neighbor's embeddings.

The item model first converts the item into its latent representation using an embedding operation, Ω_B that is similar to Ω_A used for users. However, different from TransNet and CF-GCN, the MemNet model uses the user and neighborhood representations to transform the item representation into a customized item representation that suits the tastes of the user. For this purpose, they implemented an adaptation of Dynamic Memory Networks [147, 177], which constructed increasingly abstract representations of the item by injecting the user and neighborhood information. The memory layer consists of h computational layers, also called *hops*, that modifies the item representation like so:

$$v_B^j = tanh(W_B v_B^{j-1} + W_U v_A + W_N v_N + b) \tag{4.1}$$

where, v_B^j is the item's refined representation at hop j, v_A and v_N are the user's and their neighborhood's representations, $W_\bullet$ are the corresponding weights and b, a bias vector. The output of the final hop, v_B^h is the context vector for a LSTM decoder as shown in Figure 4.13, which is concatenated to the input at each step. The same output is also passed through a regression layer to predict the user's rating for the item.

We use an implementation of MemNet that is comparable to that of TransNet and CF-GCN. Specifically, we make the following changes: MemNet as described in [169] uses reviews to construct user and item representations. We use an embedding matrix to embed users and items, which is learned during the training phase. MemNet's regression layer computes a weighted average of all ratings for that item and a single-layer feedforward regression on the refined item representation. Our implementation uses a Factorization Machine regression on the refined item representation similar to TransNet for predicting the actual rating.

4.7.3 Automatic Evaluation Using Different Metrics

In the above two subsections, we described our competitive baselines. In this subsection, we will study their performance compared to TransNet on multiple datasets using the metrics described in Section 4.4. We have ensured that the models are comparable — the parameter settings and dimensions are the same whenever they serve equivalent purposes in the models.

4.7.3.1 Mean Squared Error

First, we compare the performance of the models according to their rating prediction error computed using the Mean Squared Error (MSE) metric defined in Section 4.4. Recall that TransNet's regression layer is not jointly trained with the rest of the model, whereas CF-GCN's and MemNet's regression layers are jointly trained. The MSE values achieved by the three models on different datasets are charted in Table 4.6. The best scores are highlighted in blue. From the table, it is clear that CF-GCN, the simpler of the three models that trains its regression layer directly, is better at predicting the ratings. Although, it appears to contradict the result that was presented earlier in Chapter 3, note that the two TransNet models have different Oracles. In the rating prediction case discussed in Chapter 3, the Oracle was taking the target review as input and predicting the rating.

Dataset	TransNet	CF-GCN	MemNet
Yelp18 40-core	1.18	1.06	1.16
Yelp18 25-core	1.29	1.19	1.29
AZ-Elec 15-core	0.87	0.82	0.86
AZ-CSJ 5-core	1.25	1.25	1.25
AZ-Books 25-core	1.08	0.94	0.97
BeerAdv 150-core	0.41	0.34	0.41
RateBeer 150-core	10.08	4.44	10.30

Table 4.6: Mean Squared Error comparison

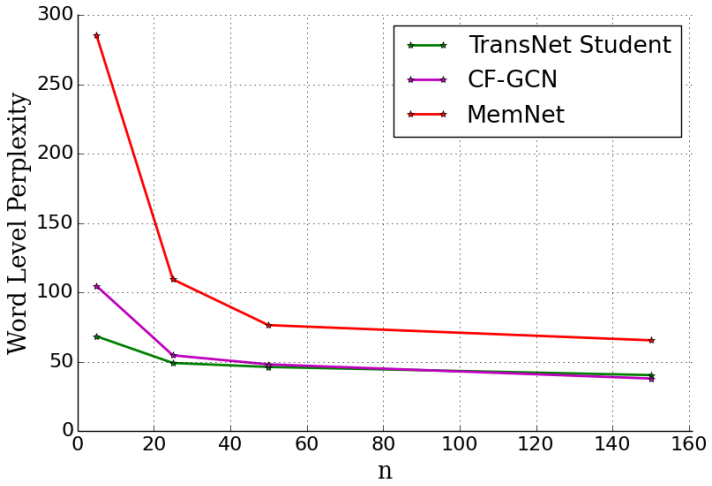

Figure 4.14: Performance of different models under low data settings

Therefore, its internal representation and by induction, that of the Student, were geared towards rating prediction. However, in the setting discussed in this chapter, the Oracle's aim is to regenerate the review. Therefore, neither the Oracle nor the Student is making an effort to adapt their internal representations towards predicting the rating.

4.7.3.2 Word Level Perplexity

In this section, we first compare the models' Word Level Perplexity under low data settings. This experiment is designed to show how the behavior of the models change when the training data is scarce to when it is sufficient. The datasets used for this experiment are variants of `BeerAdv 150-core` constructed using the `k-core-n` procedure described in Section 4.3. The values of `n` used are 5, 25, 50 and 150. In these variants, for example, when $n = 5$, each user in the `BeerAdv 150-core-5` dataset will have exactly 5 reviews in the training set. The number of users are the same as the original `BeerAdv 150-core` dataset. The parameter configuration used for this experiment is `small`.

Figure 4.14 plots the Word Level Perplexity of TransNet and the two competitive baselines under low data settings. As can be observed from the figure, at the lowest data setting (`n = 5`), TransNet

Dataset	TransNet	CF-GCN	MemNet
`Yelp18 40-core`	57.678	49.654	175.006
`Yelp18 25-core`	35.817	25.285	33.456
`AZ-Elec 15-core`	66.974	69.219	110.359
`AZ-CSJ 5-core`	10.393	6.924	7.57
`AZ-Books 25-core`	29.71	22.274	24.005
`BeerAdv 150-core`	29.167	22.33	27.704
`RateBeer 150-core`	8.833	5.899	6.85

Table 4.7: Word Level Perplexity Comparison

achieves a substantially lower perplexity compared to the baselines. As the amount of data is increased, all models improve their performances. At `n = 50`, CF-GCN is on par with TransNet, and with more data, it is able to surpass TransNet marginally. MemNet's performance, however, contrasts that of CF-GCN. At very low data settings, its perplexity is extremely large. With more data, it does improve its performance, but is not comparable to that of TransNet or CF-GCN. MemNet is a much larger model with considerably more number of parameters. With sparse training data, it is conceivable that it might be overfitting. This experiment shows that TransNet is a much better model in low data settings, which is an important property in real world scenarios where the training data available is meagre.

In the second set of experiments, we compare the performance of the models on larger datasets. The parameter configuration used for this experiment is `medium`. The scores achieved by the models are tabulated in Table 4.7. The best scores are highlighted in blue. It is obvious from the table that CF-GCN achieves the lowest perplexity in all cases except for one, compared to TransNet and MemNet. `AZ-Elec 15-core`, where TransNet betters CF-GCN's score is the smallest of the datasets, showing that TransNet does better at low data settings. This is consistent with the observations made in the first set of experiments, plotted in Figure 4.14.

However, there are certain cases in the above experiments where the perplexity numbers are misleading. `AZ-CSJ 5-core` is one such case. Inspecting the review generated by CF-GCN, we see that, it only ever outputs the text, "*i love this shirt . it is very comfortable and fits well . i am 5 ' 4 " and it fits me perfectly . i am 5 ' 4 " and it fits me perfectly . i am 5 ' 4 " and it hits me below my knee . i am 5 ' 7 " and it hits me below my knee . i am 5 ' 7 " and it hits me below my knee . i am 5 ' 7 " and it hits me below my knee . i 'm 5 ' 7 ""*, regardless of the user - item pair. Similarly, MemNet generates the following review for all user - item pairs: "*i love this shirt . it is very comfortable and fits well . i am 5 ' 4 " and it fits me perfectly . i am 5 ' 4 " and it 's a little long on me . i 'm 5 ' 4 " and it 's a little long on me . i 'm 5 ' 4 " and it 's a little long on me . i 'm 5 ' 4 " and it 's a little long . i 'm 5 ' 4 " and it 's a little long . i 'm 5 '*". The items in the dataset include jewelry, shoes, dress, bags, and other types of clothing items etc. besides shirts. Although, according to perplexity numbers, TransNet ranks the lowest on this dataset, on inspecting its output, we see that it is able to generate a variety of reviews such as, (a) "*i have a few of these tops in different colors . i am a size 14 and i ordered a large . it fits perfectly , and the material is so soft and comfortable . i have a few of the tops and i am so glad i did . it 's a beautiful dress , and i am a huge fan of the colors . i am a fan of the colors and the colors are vibrant and the fabric is very soft and comfortable*" and (b)

"i am a size 16 and i ordered a large . i have a large chest and the top of the shirt is not tight . i am not sure how long that will last . i will be returning this item ."

Similarly, in the case of `Yelp18 25-core`, although MemNet achieves a lower perplexity than TransNet, it only ever outputs the following review: *"I love this place. I love the atmosphere and the service is always great. The food is always fresh and the service is always great. I love the atmosphere and the service is always great."*, regardless of the user - item pair. The Yelp reviews have businesses like casinos, stage shows, grocery stores etc. in addition to restaurants. Both TransNet and CF-GCN output a variety of reviews depending on the item.

`AZ-Books 25-core` is the other dataset where a similar phenomenon is observed. According to the perplexity numbers, both CF-GCN and MemNet are better models than TransNet. However, CF-GCN alternates between the following two reviews for all user-item pairs: (a) *"i loved this book . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line . i loved the characters"* and (b) *"i really enjoyed this book . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line . i would recommend this book to anyone who enjoys a good romance . "* Similarly, MemNet produces, *"i loved this book . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line . i loved the characters and the story line ."*, for all user-item pairs. In contrast, TransNet produces a variety of reviews. For example, (a) *"i have read all of the books in this trilogy and have read them all . this one is a great read . the author has a way of making the reader feel like you know them . the author has a way of making you feel like you 're right in the middle of the world . i loved the way she was able to make the reader feel like you know them . the author has a way of making you feel like you 're right in the middle of the story ."*, (b) *"i loved this book . i loved the way the author wrote the whole thing . i loved the way the author developed the emotions ."* and (c) *"i have been a fan of the author for a long time . she has a way of writing that draws you in and keeps you turning the pages . she is a master of the craft and the ability to create a world that is not only a world of magic , but also a great deal of the world . the book is a great read , and i will be reading the next one . i have read all of her books and have been waiting for it to come out ."*

Therefore, we must note that while Word Level Perplexity is an important metric for measuring the competence of a model, there are real world scenarios where a low perplexity model is not necessarily useful at generating good natural language text. Such a discrepancy is most likely because perplexity is measured as the probability of the model generating the next word of the ground truth data **given** the previous word of the ground truth data. This process is different from the actual text generation itself because in the latter setting, the model is fed its own output from the previous step. i.e. it does not have access to any guiding mechanism when generating the text. In real world settings, at the time of generating the text, we cannot assume that the model will have access to the ground truth because in most cases, no such true data exists. Therefore, although perplexity is a good metric for measuring the progress of model training, practitioners must be aware of the pitfalls of relying solely on perplexity and would need to manually check and confirm that the model is indeed generating reasonable text.

Dataset	TransNet	CF-GCN	MemNet
Yelp18 40-core	72.36 %	61.08 %	0.20 %
Yelp18 25-core	70.13 %	13.31 %	0.06 %
AZ-Elec 15-core	72.92 %	30.88 %	11.52 %
AZ-CSJ 5-core	81.42 %	0.02 %	0.005 %
AZ-Books 25-core	13.38 %	0.12 %	0.010 %
BeerAdv 150-core	97.30 %	98.34 %	83.97 %
RateBeer 150-core	63.00 %	97.10 %	70.77 %

Table 4.8: Input Sensitivity comparison

4.7.3.3 Input Sensitivity (IS)

To quantify the observations about repetitive reviews produced by generative models that was discussed in the previous subsection, we propose a new metric, which we call *Input Sensitivity* (IS). It measures a distinct aspect of language generating models that is not covered by the other popular metrics. Computed as the proportion of unique reviews generated, it measures the sensitivity of the model to its inputs. Unlike other metrics that refer to the ground truth for their computation, IS only considers the output of the model itself. Therefore, it is not a be-all and end-all of metrics for comparing language generating models; rather, it gives an important insight into the models' performance that is not otherwise measured by other metrics and hence, needs to be used in conjunction with other metrics.

The Input Sensitivity scores of TransNet and the competing baselines for various datasets are shown in Table 4.8. Best scores are highlighted in blue. It is obvious from the table that TransNet is, in general, producing a diverse set of reviews when compared to the baselines. Specifically, the observations about repetitive texts from earlier sections are quantified by these scores. In the case of `Yelp18 25-core`, `AZ-Elec 15-core` and `AZ-CSJ 5-core`, we can see that TransNet is substantially better than the baselines. It also performs better in the case of `Yelp18 40-core` and `AZ-Books 25-core`, although the difference is comparatively smaller.

It is also clear from the table that all algorithms perform their best with the beer datasets, and CF-GCN is the better of the three models in generating beer reviews. However, CF-GCN tends to generate the same set of texts in a large proportion of the examples in all the other datasets, except `Yelp18 40-core`.

An interesting observation is the performance of MemNet — although it achieves good scores with other metrics, as we can see from the table, its Input Sensitivity is abysmal in all datasets except beer. It is comparable to TransNet and CF-GCN in the case of `BeerAdv 150-core`, and surpasses the performance of TransNet in the case of `RateBeer 150-core`.

4.7.3.4 BiLingual Evaluation Understudy (BLEU)

Given the issues with perplexity scores noted earlier, in this section, we will compare the performance of the three models using the BLEU [122] metric. BLEU is typically used to compare translations with reference sentences. BLEU 1, 2, 3 and 4 are the individual scores for 1-gram, 2-gram,

3-gram and 4-gram phrases. BLEU Overall is the cumulative scores obtained from the n-gram scores by calculating their weighted geometric mean. We use the implementation of BLEU in the Python Natural Language ToolKit (NLTK) library. The BLEU scores of the models are tabulated in Tables 4.9, 4.10 and 4.11. The best scores are highlighted in blue. In the tables, we also include the Input Sensitivity scores for easy reference. IS values of 25% or below are highlighted in red and those between 25–50% are highlighted in yellow — these are the unsatisfactory cases.

From Yelp dataset scores in Table 4.9, it can be seen that although MemNet performs better than TransNet according to the BLEU Overall scores as well as BLEU 2, BLEU 3 and BLEU 4 scores, its IS scores are terribly low. Similarly, in the case of `Yelp18 25-core` dataset, although CF-GCN is the best scoring system according to the BLEU metrics, its IS score is only 13% compared to TransNet's 70%.

In the case of Amazon datasets, from Table 4.10, it can be seen that CF-GCN achieves the best BLEU Overall, BLEU 2 and BLEU 3 scores while TransNet gets the best BLEU 1 scores in all the datasets. However, from the IS scores, we see that CF-GCN's reviews are mostly the same for all user-item pairs, especially in the case of `AZ-CSJ 5-core` dataset, examples of which were discussed earlier in the experiments with Perplexity scores.

All methods do well on Beer datasets as can be seen in the Table 4.11. TransNet achieves the best Overall BLEU score in the case of `BeerAdv 150-core` dataset, and MemNet in `RateBeer 150-core` dataset. For the latter, note that MemNet's IS is only 71% compared to CF-GCN's 97%.

Although, BLEU is a widely used metric to measure the similarity of pairs of natural language texts, it is known to have limitations. Since it uses a modified form of precision to compute the score, it only checks if the words in the candidate text have a match in reference text, but not the other way round. i.e. it does not penalize the candidate if it misses to produce parts of the reference text. Another of the key components of the metric is a brevity penalty that penalizes very short text. However, it overlooks cases where the text is repetitive. i.e. the candidate text is not short in length thereby sidestepping the brevity penalty, but is low in content. Therefore, similar to the contradictory observations in the Word Level Perplexity measurements earlier, there are cases in the BLEU measurements as well where the relative scores do not follow the actual observations.

4.7.3.5 Metric for Evaluation of Translation with Explicit ORdering (METEOR)

METEOR [40], proposed to overcome the limitations of BLEU, uses a harmonic mean of precision and recall along with a fragmentation penalty to compute the score. It has been shown to obtain higher correlation with human judgements than BLEU for evaluating translations. We use the reference implementation provided by the authors of the metric. The METEOR scores of the three models on various datasets are tabulated in Tables 4.12, 4.13 and 4.14.

In the case of Yelp datasets, it can be seen from Table 4.12 that TransNet is the best model according to METEOR, which is also in accordance with the corresponding IS scores. From Table 4.14, it is clear that CF-GCN performs the best on Beer datasets, correlating well with the corresponding IS scores. However, in the case of Amazon datasets, although METEOR ranks TransNet the highest in two of the datasets, on `AZ-CSJ 5-core` dataset, it ranks MemNet the highest. However, as we discussed earlier, this is one of the datasets where MemNet produces the same text for every user-item pair, which is also evident from its extremely low IS score. Therefore, although METEOR is able to rectify the relative scoring provided by BLEU and Word Perplexity in most of the anomalous cases,

Dataset		TransNet	CF-GCN	MemNet
Yelp18 40-core	BLEU overall	16.09	19.69	19.656
	BLEU 1	12.83	10.62	10.396
	BLEU 2	12.49	22.79	21.370
	BLEU 3	43.72	49.31	53.895
	BLEU 4	52.88	56.12	58.997
	IS	72.36 %	61.08 %	0.20 %
Yelp18 25-core	BLEU overall	15.41	19.57	17.73
	BLEU 1	8.58	9.12	6.63
	BLEU 2	16.97	24.66	24.79
	BLEU 3	33.59	43.10	35.60
	BLEU 4	37.81	48.19	37.78
	IS	70.13 %	13.31 %	0.06 %

Table 4.9: BLEU comparison on the Yelp datasets

Dataset		TransNet	CF-GCN	MemNet
AZ-Elec 15-core	BLEU overall	10.77	12.52	12.07
	BLEU 1	12.91	9.71	11.70
	BLEU 2	4.57	9.03	7.16
	BLEU 3	27.63	34.09	31.59
	BLEU 4	40.63	41.95	42.05
	IS	72.92 %	30.88 %	11.52 %
AZ-CSJ 5-core	BLEU overall	24.80	26.24	24.00
	BLEU 1	13.76	12.13	11.92
	BLEU 2	25.36	29.22	25.65
	BLEU 3	52.26	67.07	64.55
	BLEU 4	57.14	76.89	77.01
	IS	81.42 %	0.02 %	0.005 %
AZ-Books 25-core	BLEU overall	12.64	14.72	11.56
	BLEU 1	11.33	9.71	7.56
	BLEU 2	8.55	12.64	10.42
	BLEU 3	28.31	39.96	25.20
	BLEU 4	35.82	47.77	29.21
	IS	13.38 %	0.12 %	0.010 %

Table 4.10: BLEU comparison on the Amazon datasets

Dataset		TransNet	CF-GCN	MemNet
BeerAdv 150-core	BLEU overall	14.63	12.76	13.72
	BLEU 1	19.87	21.95	16.44
	BLEU 2	7.61	8.10	6.63
	BLEU 3	29.68	20.98	31.00
	BLEU 4	51.13	42.04	47.31
	IS	97.30 %	98.34 %	83.97 %
RateBeer 150-core	BLEU overall	14.68	22.61	24.92
	BLEU 1	9.35	21.08	15.81
	BLEU 2	14.94	21.87	25.99
	BLEU 3	28.14	42.17	53.02
	BLEU 4	32.74	55.62	63.47
	IS	63.00 %	97.10 %	70.77 %

Table 4.11: BLEU comparison on the Beer datasets

Dataset		TransNet	CF-GCN	MemNet
Yelp18 40-core	METEOR	6.71	5.34	5.81
	Precision	32.27	24.72	26.37
	Recall	16.58	13.53	14.81
	IS	72.36 %	61.08 %	0.20 %
Yelp18 25-core	METEOR	5.18	4.81	3.76
	Precision	34.11	25.49	27.79
	Recall	11.96	11.88	9.10
	IS	70.13 %	13.31 %	0.06 %

Table 4.12: METEOR comparison on the Yelp datasets

there exists datasets where METEOR is unable to detect the phenomenon.

4.7.3.6 Recall-Oriented Understudy for Gisting Evaluation (ROUGE)

[87] proposed a recall oriented metric called ROUGE to overcome many of the limitations observed in the BLEU scoring mechanism. It uses both precision and recall numbers, but gives higher importance to the latter. In addition to measuring n-gram overlaps (ROUGE-1, ROUGE-2, ROUGE-3), it can also measure the length of the longest common subsequence overlap (ROUGE-L). Another useful type of ROUGE score is the Skip-gram match. It measures matches of subsequences of 2, 3 or 4 words that are in order, but not necessarily consecutive (ROUGE-S2, ROUGE-S3, ROUGE-S4). We use the Java implementation provided by [46]. The ROUGE scores of the three models on various datasets are tabulated in Table 4.15, Table 4.16 and Table 4.17.

From the comparison on the Yelp datasets shown in Table 4.15, it can be seen that TransNet is able to obtain the best scores in all ROUGE types except ROUGE-L. However, as we can see, ROUGE-L

Dataset		TransNet	CF-GCN	MemNet
AZ-Elec 15-core	METEOR	6.77	5.19	6.26
	Precision	41.84	32.89	39.51
	Recall	15.20	11.96	14.40
	IS	72.92 %	30.88 %	11.52 %
AZ-CSJ 5-core	METEOR	7.13	6.75	7.43
	Precision	27.90	16.18	15.80
	Recall	16.31	16.43	18.63
	IS	81.42 %	0.02 %	0.005 %
AZ-Books 25-core	METEOR	6.17	5.24	3.91
	Precision	42.52	30.69	44.10
	Recall	13.96	11.85	8.18
	IS	13.38 %	0.12 %	0.010 %

Table 4.13: METEOR comparison on the Amazon datasets

Dataset		TransNet	CF-GCN	MemNet
BeerAdv 150-core	METEOR	10.77	13.22	9.44
	Precision	35.55	40.14	34.46
	Recall	21.96	24.12	18.87
	IS	97.30 %	98.34 %	83.97 %
RateBeer 150-core	METEOR	7.15	14.05	9.84
	Precision	31.42	32.67	26.21
	Recall	12.91	23.85	18.37
	IS	63.00 %	97.10 %	70.77 %

Table 4.14: METEOR comparison on the Beer datasets

scores CF-GCN higher in `Yelp18 25-core` dataset although its IS score is very low. Also, in the case of Yelp datasets, CF-GCN consistently scores better than MemNet.

Similarly, in the case of Amazon datasets given in Table 4.16, TransNet is the best in almost all ROUGE computations. However, in contrast to Yelp, MemNet achieves better ROUGE scores in most cases than CF-GCN, although from their IS scores, we know that both models are generating the same or very similar reviews for all user-item pairs. For `AZ-CSJ 5-core` dataset, some of the ROUGE scores, for example, ROUGE 2 and ROUGE 3 score MemNet higher although its IS score is appallingly low.

In the case of beer reviews, it can be seen from Table 4.17 that CF-GCN is the best model in generating user reviews. TransNet is better than MemNet on the `BeerAdv 150-core` dataset, but not on the `RateBeer 150-core` dataset, which is similar to the METEOR results.

4.7.3.7 Discussion

From our experiments discussed above, we observed that all methods perform substantially better on the beer datasets, with CF-GCN being the best model. However, as we know, the beer datasets are comparatively cleaner (reviews almost always focus entirely on the beer being reviewed and not on unrelated details), denser (each user and each item has at least 150 reviews and therefore, the rating matrix has a larger number of filled cells), larger (more number of training examples) and better structured (most reviews discuss a specific set of five aspects of beers and mostly in a fixed order) than the other datasets. We observed that when the amount of training data is less or sparse, TransNet performs better than other models.

Among the automatic scoring metrics, Word Perplexity and BLEU scores can be misleading in certain cases, where the best method according to the metric may not be able to generate any useful reviews. METEOR and ROUGE are able to rectify the relative scoring provided by Word Perplexity and BLEU in most of the anomalous cases, but there exists datasets where not all METEOR and ROUGE scores correspond to the actual observations. Therefore, evaluation by human judges is desirable in such cases.

4.7.4 Human Evaluation on Amazon Mechanical Turk

Our goal in this section is to assess the goodness of the generated reviews by human judges. We used Amazon Mechanical Turk[4] to recruit participants (referred to as *turkers*). Amazon Mechanical Turk (MTurk) is an online marketplace for work that requires human intelligence. The turkers choose from the large number of tasks posted on the website called Human Intelligence Tasks (HIT) to work on, in an on-demand basis and get paid for their services. We used MTurk's Requester website[5] to set up our tasks.

To help make their judgements in the review comparison task as objective as possible, we devised an aspect and reason tagging scheme, detailed below. Also, in order to reduce confusion between liked and disliked aspects, we separated the two aspects and the turkers were asked to tag them in different sessions. This study was approved by CMU's Institutional Review Board (IRB) as STUDY 2018 00000130.

[4]https://www.mturk.com

[5]https://requester.mturk.com

Dataset		TransNet	CF-GCN	MemNet
Yelp18 40-core	ROUGE-1	23.78	19.16	18.92
	ROUGE-2	3.07	2.05	1.70
	ROUGE-3	0.44	0.33	0.16
	ROUGE-L	4.91	5.32	5.16
	ROUGE-S2	3.12	1.96	1.73
	ROUGE-S3	3.58	2.23	2.23
	ROUGE-S4	3.93	2.49	2.35
	IS	72.36 %	61.08 %	0.20 %
Yelp18 25-core	ROUGE-1	19.59	18.95	16.84
	ROUGE-2	2.53	2.03	1.57
	ROUGE-3	0.37	0.35	0.24
	ROUGE-L	6.20	6.81	6.62
	ROUGE-S2	2.49	2.00	1.68
	ROUGE-S3	2.71	2.26	1.93
	ROUGE-S4	3.00	2.43	2.11
	IS	70.13 %	13.31 %	0.06 %

Table 4.15: ROUGE comparison on the Yelp datasets

4.7.4.1 Instructions for the Study

Below are the instructions given to the turkers for judging aspects that were liked by a user:

Task: In this study, you will be shown 3 reviews for an item: The first review is the original review (true data) that a user wrote for that item. The next two, referred to as Test Reviews, are the outputs of two algorithms being tested. Your task is to determine which one of the two test reviews is most similar to the original review with regard to aspects liked by the user.
Criteria: Your answers will be manually and/or programmatically inspected to check if you adhered to the instructions given below.
To successfully judge the Test Reviews, please follow the instructions below:

1. The reviews shown to you can be about movies, restaurants, electronic products, clothing items, beers etc. We will refer to them as items.
    - The first review is the original review (true data) that the user wrote for that item.
    - The next two, referred to as Test Reviews, are the outputs of two algorithms being tested.
2. You need to read the reviews and tag (copy-paste or type) aspects of the item mentioned in each of the reviews that the user LIKED, in the corresponding text boxes - 'Aspect LIKED'.
    - An aspect of an item is anything associated with the item - like features and characteristics. For a movie, it could be specific actors, directors, choreography, etc. For a restaurant, it could be the price, ambience, parking, specific foods, service etc.
    - Use words or phrases from the review shown to you when possible.

Dataset		TransNet	CF-GCN	MemNet
AZ-Elec 15-core	ROUGE-1	21.81	16.85	20.41
	ROUGE-2	3.00	1.52	2.25
	ROUGE-3	0.42	0.22	0.28
	ROUGE-L	4.93	4.81	4.82
	ROUGE-S2	2.86	1.46	2.24
	ROUGE-S3	3.18	1.55	2.59
	ROUGE-S4	3.49	1.84	2.99
	IS	72.92 %	30.88 %	11.52 %
AZ-CSJ 5-core	ROUGE-1	18.15	14.04	14.65
	ROUGE-2	1.85	1.87	2.06
	ROUGE-3	0.24	0.29	0.32
	ROUGE-L	8.89	8.62	8.23
	ROUGE-S2	1.78	1.62	1.74
	ROUGE-S3	1.97	1.72	1.87
	ROUGE-S4	2.19	1.82	1.94
	IS	81.42 %	0.02 %	0.005 %
AZ-Books 25-core	ROUGE-1	19.99	15.48	14.89
	ROUGE-2	2.88	1.81	2.03
	ROUGE-3	0.39	0.27	0.30
	ROUGE-L	6.25	6.63	6.51
	ROUGE-S2	2.80	1.62	1.83
	ROUGE-S3	3.00	1.85	2.11
	ROUGE-S4	3.23	2.11	2.40
	IS	13.38 %	0.12 %	0.010 %

Table 4.16: ROUGE comparison on the Amazon datasets

Dataset		TransNet	CF-GCN	MemNet
BeerAdv 150-core	ROUGE-1	31.91	35.44	29.83
	ROUGE-2	7.05	9.89	6.55
	ROUGE-3	1.66	3.25	1.48
	ROUGE-L	6.85	7.11	5.88
	ROUGE-S2	6.17	8.81	5.67
	ROUGE-S3	6.32	8.86	5.78
	ROUGE-S4	6.75	9.21	6.15
	IS	97.30 %	98.34 %	83.97 %
RateBeer 150-core	ROUGE-1	19.79	32.65	25.64
	ROUGE-2	3.82	10.16	5.95
	ROUGE-3	0.73	3.96	1.53
	ROUGE-L	11.43	15.06	11.46
	ROUGE-S2	3.19	9.16	5.13
	ROUGE-S3	3.34	9.29	5.24
	ROUGE-S4	3.64	9.66	5.70
	IS	63.00 %	97.10 %	70.77 %

Table 4.17: ROUGE comparison on the Beer datasets

- You need to include all LIKED aspects, up to a maximum of 6 per review, and not omit anything.
- Include only aspects applicable to the item being reviewed. If the user talks about other items and their aspects, don't include them.
- Ignore aspects that the user disliked - those will be tagged in a separate task later.

3. For each of the LIKED aspects, you need to also copy-paste the phrase(s) and/or sentence(s) from the review that show why the user liked that aspect, in the corresponding text boxes for the reviews under 'Reason'. If there are multiple sentences or phrases describing why they liked an aspect, include all of them separated by semicolon.
4. If the user mentions two or more experiences in the same review about an aspect, use the latest to check if they liked that aspect or not.
5. If there are aspects that the user mentioned but did not particularly like or dislike it (neutral sentiment), you need to also tag (copy-paste or type) them in the corresponding text boxes for the reviews under 'Aspect Neutral'.
    - You need to include all neutral aspects, up to a maximum of 6 per review, and not omit anything.
    - Include only aspects applicable to the item being reviewed. If the user talks about other items and their aspects, don't include them.
6. For each of the neutral aspects, you need to also copy-paste the phrase or sentence that the user used to describe them in the corresponding text boxes for the reviews under 'Description'.

- If there are multiple sentences or phrases describing the neutral aspect, include all of them separated by semicolon.
- If a sentence describing a neutral aspect also contains other aspects, you do not need to edit it out.

7. If the user did not like anything about the item, enter 'None' in the corresponding 'Aspect LIKED' text box.
8. If the user did not describe any aspect with neutral sentiment, enter 'None' in the corresponding 'Aspect Neutral' text box.
9. You need to tag all three reviews. Even if the Original Review or any of the Test Reviews do not have any aspects liked by the user, you still need to tag the other two reviews.
10. Based on your tagging of aspects and reason/description, you need to then choose the Test Review that is most similar to the Original Review in terms of the number of aspects and reasons/descriptions matched.
    - When you are trying to match the aspects between a test review and the original review, you must take synonyms into consideration. E.g. 'smell' and 'aroma', 'price' and 'affordability' etc.
    - In the case of beer, where the reviews contain a number of aspects, you will be also asked to compute a score for each of the test reviews. To compute a score for a test review, add 2 points for each aspect and reason/description matched with the Original Review, and 1 point if only the aspect is matched, but not the reason/description.
11. If a Test Review is discussing an item that is obviously different from that reviewed in the Original, then there is no match.
    - For example, if the Original Review is about a bag and the Test Review is about an earring, don't count aspect matches.
    - However, if both reviews are about bags, but Test Review's color doesn't match that of the Original, you need to count the aspect matches because in this case, both items are bags, only aspects are different.
12. Note that the algorithms generating the Test Reviews are shuffled in random order. So, your answers won't be all 'A's or all 'B's always.

An example of the screen displayed to the turkers for comparing liked aspects is shown in Figure 4.15. It also depicts the scoring question. A similar task for comparing reviews based on disliked aspects was also setup. An example of that screen is shown in Figure 4.16. The turkers were asked to tag neutral aspects as well, to cover the cases where the reviews did not specifically like or dislike that aspect, but thought it important to mention. We also allowed counting matches between neutral and liked (disliked) aspects to include those cases where one review discusses it as a fact associated with the item and the other review specifically likes (dislikes) it. However, in that case, only the aspect term is matched, not the reason / description. For example, if a user cares about parking space in general, then it is important to mention about the parking situation at the particular business establishment being reviewed. In addition to the above instructions, they were also supplied with sample filled out

examples from each of the datasets. Moreover, their queries and comments were incorporated into a Frequently Asked Questions section and a Notes section as the study progressed.

4.7.4.2 Specifics of the Study

HITs for pair-wise comparisons of each dataset and each combination of competitive algorithms were released as one batch. Each HIT was worked on by only one turker. Once all the HITs in a batch are complete, the MTurk website provides a detailed comma-separated file containing all attributes of the HITs and their responses, that can be downloaded. We implemented a Python script that processes the comma separated file and flags suspicious cases. The Python script specifically looks for cases where there are no tags, tagged words or phrases that cannot be found in the review text, reason and description sentences that are missing, the number of tagged aspects less than a particular number chosen according to the dataset, or missing scores in those cases where they were asked to compute them. The flagged cases were manually checked to verify the turker's response. Responses that were not flagged were automatically approved.

Throughout the study, we used a tally system of `warn` and `reject` counts. If a turker's answer to a HIT was almost correct, but not fully correct, it would count towards their `warn` tally. These include cases where they tagged some aspects, but not all the applicable ones, they forgot to include the reason or description sentences for the tags, or they forgot to compute scores. If a turker chose the wrong test review, that would also count towards their `warn` tally. These cases are harder to flag with the Python script. However, in certain datasets where we know that one of the algorithm's outputs are always the same, we manually checked those HITs where the turker chose the repetitive review as the best matching one. If a turker did not tag anything for a HIT, it would count towards their `reject` tally. The turker was paid for those HITs assessed as `warn` at the same rate as the correct ones, but was not paid for those assessed as `reject`. After each batch is evaluated, their tallies were updated. To continue in the study, a turker would need to maintain their `reject` count under 5 and their `warn` count under 10, which also served as an incentive for the turkers to answer correctly.

Before the turkers could work on the main task, they were required to pass a qualification test. The qualifier was structured identical to the main task, except that it was unpaid and their responses were approved on the MTurk website regardless of whether they got it correct or not. Therefore, by participating in the qualification test, they would not accrue rejects and consequently, their overall approval rating on MTurk would not be affected. The turkers were required to work on 5 HITs from the qualifier task and had to get at least 4 of them correct to pass the test. We limited the participation to only those turkers who had previously worked on another of our studies (discussed in Chapter 5) and had consistently performed well in that study. If throughout that study, their number of rejected answers was at most 5 and their number of answers assessed as 'warn' was at most 10, they were assigned a 'qualification' called `'Passed Level 3 of CMU_STUDY2018_00000337'`. That qualification was used as a prerequisite to be eligible to participate in the qualifier test for the review comparison study. 75 turkers were eligible to take the qualifier test, which was live for 5 days. Only 37 turkers participated in the qualification test. Their responses to the qualifier HITs were checked manually and all of the participants got all their responses correct. The manual checking of the qualifier HITs was helpful in refining the Python script that was later used in the main study.

Due to restrictions enforced because of GDPR[6] being adopted in certain countries, CMU's IRB

[6]https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

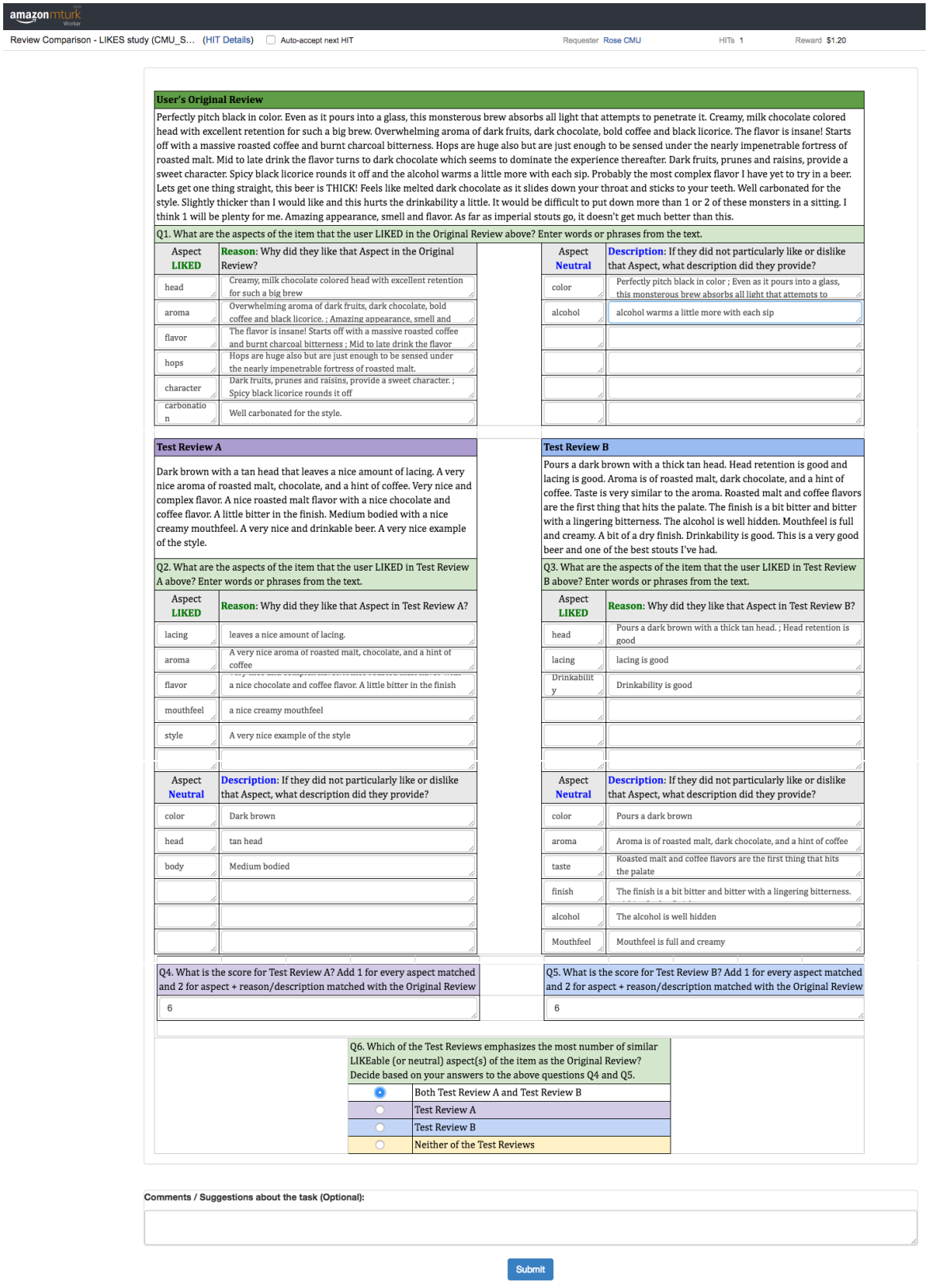

Figure 4.15: Sample Mechanical Turk Screen for Review Comparison with a scoring mechanism for Liked Aspects

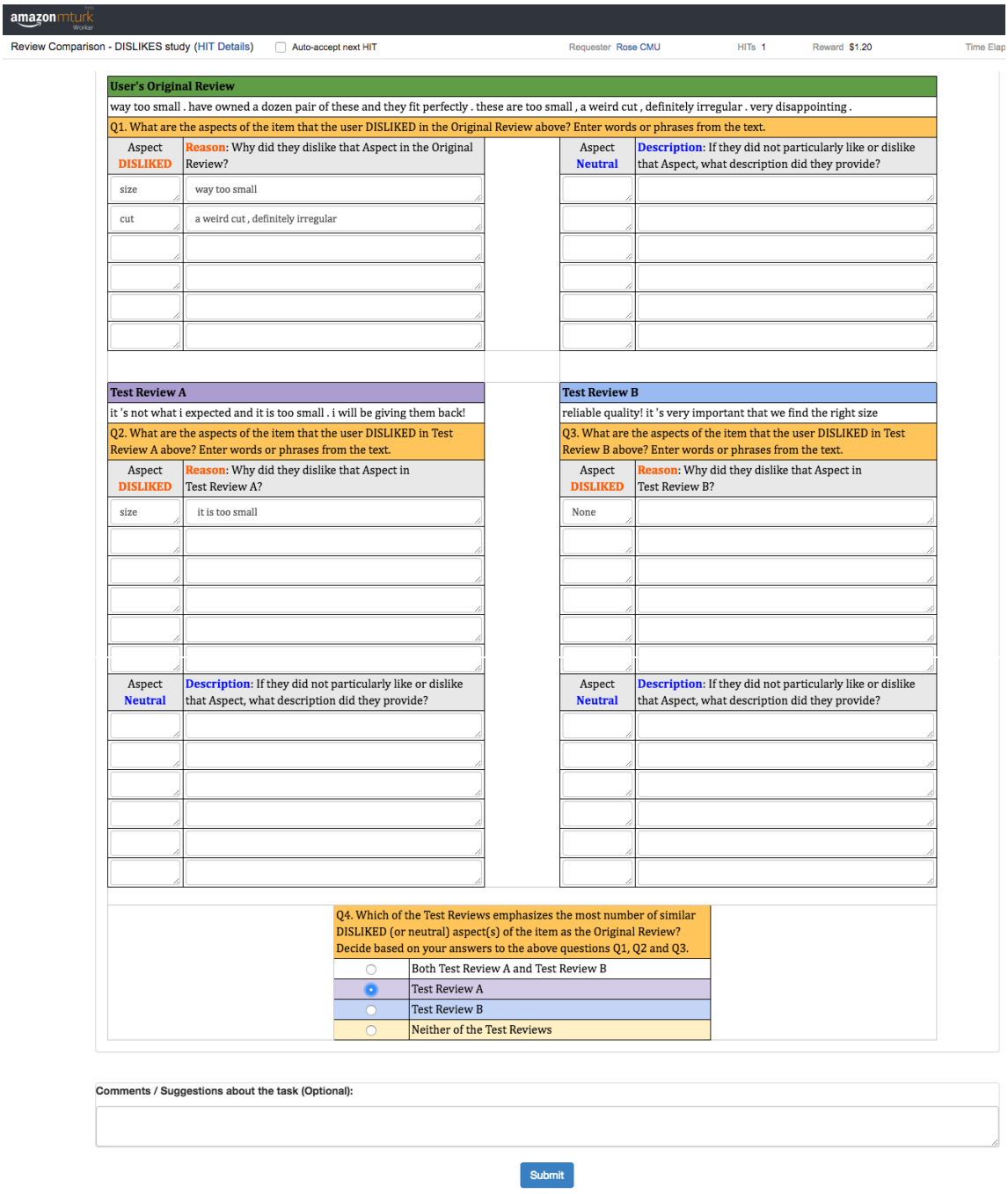

Figure 4.16: Sample Mechanical Turk Screen for Review Comparison for Disliked Aspects

required us to select only participants located in the USA. This was enforced using the `Location` attribute of the turkers as provided by MTurk. To determine the pay rate, we timed ourselves while manually tagging a few HITs from each of the datasets and aimed at a pay of approximately US $10.00 per hour, which is higher than the prevailing US Federal Minimum Wage of $7.25.[7] The pay per HIT was set to a lower value for certain datasets like `AZ-CSJ` that has shorter reviews and lesser number of aspects to be tagged per review and therefore took shorter time to tag, while it was set to a higher value for the `BeerAdv 150-core` dataset and the review selection (not review generation) comparison on the `Yelp17` dataset, which have detailed reviews with 5 or more tags per review, and consequently took longer to tag. The pay per HIT ranged from US $0.80 to $1.30. However, the pay per hour is comparable across datasets and worked out to US $11.50 on average. There was also an additional 20% fee to be paid to Amazon. During the study, it was also brought to our attention that many turkers hoarded HITs using automatic tools. Such tools would accept a specific number of HITs simultaneously on that turker's behalf, thereby ensuring that the turker would have that many HITs to work on before they were all claimed by other turkers. However, we noticed that when turkers were holding on to multiple HITs simultaneously, unfortunately, it increased the average time per HIT reported by MTurk because it is computed as the time of accept to time of submission, which is larger if that HIT was accepted, but not worked on right away. The increase in the average time was observed in real time on the MTurk website. We also noticed that the number of `warns` issued also increased for the HITs submitted later because the turkers were under time pressure to submit them before the allotted time ran out. To discourage this behavior, we reduced the allotted time per HIT to 15 minutes from the earlier 30 minutes and also requested the workers to accept the HITs only when they were ready to work on them, not before. The average pay per hour reported above is for the setting after making this change.

To reduce the cognitive load on the workers, we filtered out reviews that had more than 250 words. To ensure that the variance in workload for the HITs of the same batch was not large, we filtered out reviews that had less than 100 words, but if there were not enough HITs for a batch, then that number was lowered to 75 words and if required, to 50 words. Each batch typically has 100 HITs and each turker got to work through only about 3 to 5 HITs before all HITs of that batch were exhausted. To reduce fatigue, we also spaced out the batches by at least a couple of hours when there was more than one batch released on the same day.

4.7.4.3 Results of the Study

We compare the TransNet model pairwise to CF-GCN as well as a random review generator (`Random`). The random generator in this case is not supplied with the user or the item information. Otherwise, it is architecturally similar to CF-GCN. Its parameter settings and training procedure are same as that of CF-GCN and TransNet. The turkers were asked to evaluate 100 samples each for both pairs of comparisons from three datasets. For evaluating the performance on liked aspects, we chose original reviews with ratings of 4 and 5, and the outputs of the test algorithms for the corresponding user-item pairs. For disliked aspects, we chose those with ratings of 1 and 2. The absolute counts for the number of times turkers chose TransNet, the competitive baseline, 'Both Test Review A and Test Review B', and 'Neither of the Test Reviews' from the options are tabulated in Table 4.18.

[7] https://www.dol.gov/general/topic/wages/minimumwage

Dataset	Setting		TransNet	Baseline	Both	Neither
AZ-CSJ 5-core	TransNet vs. Random	likes	26	6	1	67
		dislikes	10	0	0	85
	TransNet vs. CF-GCN	likes	23	6	3	68
		dislikes	8	2	0	85
BeerAdv 150-core	TransNet vs. Random	likes	37	39	20	4
		dislikes	10	0	0	85
	TransNet vs. CF-GCN	likes	20	47	28	5
		dislikes	6	39	3	52
Yelp18 25-core	TransNet vs. Random	likes	12	27	14	47
		dislikes	9	5	6	80
	TransNet vs. CF-GCN	likes	17	36	5	35
		dislikes	7	5	5	83

Table 4.18: Human Evaluation of Reviews - Absolute Numbers

Dataset		TransNet > Random	TransNet > CF-GCN
AZ-CSJ 5-core	likes	81.25%‡	79.31%‡
	dislikes	100%‡	80%†
BeerAdv 150-core	likes	48.68%	29.85%‡
	dislikes	100%‡	13.33%‡
Yelp18 25-core	likes	30.77%†	32.08%‡
	dislikes	64.29%	58.33%

Table 4.19: Human Evaluation of Reviews - Relative Performance (‡ and † denote results that are statistically significant with *p-value* < 0.01 and *p-value* < 0.05 respectively)

The identities of the competing algorithms were not revealed to the turkers and their order was randomly shuffled. Therefore, the turkers would need to choose solely based on the content of the test reviews.

To measure the performance of TransNet in this paired test, we compute the percentage of times it was strictly better than the competing baseline in those cases where one of the methods was chosen to be better. Therefore, it excludes cases where the answer chosen was 'Both Test Review A and Test Review B' or 'Neither of the Test Reviews'. The relative performances in different settings according to the human evaluation are shown in Table 4.19. Results where TransNet is better than the corresponding baseline is colored in blue and those where it is significantly worse is colored in red. The statistical significance is calculated using McNemar's Chi-Square test [105], which is specifically suited for paired nominal data. It is applied to 2×2 contingency tables for a dichotomous property and with matched pairs of subjects. If b and c are the number of data points where one of the methods was chosen over the other and vice versa, then McNemar's test statistic is computed as $\chi^2 = \frac{(b-c)^2}{b+c}$.

As can be seen from the table, in the case of AZ-CSJ 5-core dataset, TransNet is significantly better than CF-GCN and the random review generator. In contrast to the automatic scoring mechanisms that do not consider whether the original and the test reviews are assessing the same or similar items, human judges were able to discern the difference and choose accordingly. Therefore, references to color, fit, length, quality etc., which are quite common in this dataset and are otherwise accurate word matches to the text of the ground truth data, are no longer important if they do not describe the same item.

In the case of BeerAdv 150-core dataset, it is clear from the table that CF-GCN is a much better model than TransNet with regard to both liked and disliked aspects. The random review in this case is a detailed positive review for a generic beer. Therefore, TransNet is better than the random generator in the case of disliked aspects.

In the case of Yelp18 25-core dataset, TransNet appears to be better at getting the negative aspects right, although that result is not statistically significant. The random review in this case turns out to be a very generic review that mentions atmosphere, food and service, and therefore, matches a number of positive reviews for most of the businesses reviewed in this dataset.

From the absolute numbers reported in Table 4.18, we note that, in general, the number of times neither test reviews matched the original review is higher when evaluating disliked aspects as opposed to liked aspects. This disparity is clearly visible in the case of the BeerAdv 150-core dataset, where for likes, it was a mere 5% but for dislikes, it is about 50% or more. Therefore, it appears that generating disliked aspects and reasons is harder for all the models when compared to generating liked aspects. It might also be reflective of the characteristics of reviews in general. For example, consumers may be more willing to dwell on the details of a number of aspects when they have a positive experience. However, for a negative experience, even though they might elaborate on the details of what caused that specific experience, it is possible that they may not express their opinion on other aspects out of frustration.

From the absolute numbers reported for AZ-CSJ 5-core dataset in Table 4.18, neither test reviews matched the original review in about 65–85% of test cases across liked and disliked aspects. Therefore, in this case, the competing methods are pushing the performance boundary from below and there is a lot of room for improvement. However, in the case of the BeerAdv 150-core dataset, for liked aspects, the Neither option was chosen less than 5% of the times. i.e. the methods are very competitive in generating positive beer reviews and the bar is already very high. However, there is more work to be done in terms of generating negative reviews. Performance on the Yelp18 25-core dataset lies somewhat in between these two extreme cases.

4.7.4.4 Discussion

The task of review comparison has a relatively higher cognitive load when compared to another of our MTurk studies that we discuss in Chapter 5. While it is manageable in the case of AZ-CSJ 5-core and Yelp18 25-core datasets, it is substantially higher in the case of BeerAdv 150-core dataset. For the latter, perhaps tagging neutral aspects in a separate task from the liked and disliked aspects could possibly reduce the load. Another design for the same task could have been to tag aspects in one task, and subsequently tag the reason in another task where the aspect is pre-tagged. A third task would show the tags and reasons, and ask the turker to choose the test review that best matches the original review. Although it would have made the task slightly simpler, it would lead to

duplication of efforts because each time the turker has to still read the full text of the reviews and/or the tags and reasons.

Although there appears to be a lot of work to be done in this task, it can in fact be completed in quite a straightforward manner. The turkers read each sentence of the review and determine whether it is relevant to the item being reviewed or not. If it is, then they determine the aspect(s) discussed in that sentence and the user's sentiment associated with it. If it is of the required sentiment (liked, disliked or neutral), they fill the aspect in the corresponding field and the sentence in its adjacent field. If that aspect has been tagged before and the sentiments match, then they simply append that sentence to the existing reason / description sentence. If the sentiments do not match, then they need to determine which of the sentiments is the most recent or final one, and retain only that. They repeat this for all sentences in each of the three reviews. This process was also independently suggested to us by turkers after they worked on a few HITs. We included this in the FAQ after we got feedback from turkers requesting us to check if this process was acceptable. Turkers also improved their speed once they got accustomed to the task.

It is possible that some of the results of the pair-wise comparison for the `BeerAdv 150-core` dataset may have a higher variance in reality due to its substantially higher cognitive load.

4.8 Contributions

In this chapter, we showed how our Oracle-Student architecture called TransNet can be used to generate users' reviews for items. We discussed how the components of the architecture can be modified to create variants of the basic model and empirically evaluated their effect on the performance. Finally, we compared the performance of TransNet against two state-of-the-art baselines using automatic as well as human evaluations. The results showed that theTransNet architecture that trains a review generator indirectly is substantially better than the baselines that train their generator directly, in a number of datasets and is able to produce diverse reviews.

Chapter 5

From Reviews to Explainable Recommendations: A New Benchmark Dataset

A grand vision for the future of Artificial Intelligence (AI) is to build systems that are autonomous [7, 78], cognitive [35, 38], collaborative [134] and instructable [145], as well as capable of learning by observation [85], exhibiting a personality [11, 115], and building rapport [98, 125], besides being intelligent in a worldly sense. Recent advances in AI algorithms and computing power by means of general-purpose Graphics Processing Units (GPU) and special-purpose Tensor Processing Units (TPU) have provided the impetus for research towards enabling that grand vision. One of the AI fields witnessing rapid development is that of virtual assistants. A number of commercial systems launched in the recent past like Apple Siri, Amazon Alexa, Google Home and Microsoft Cortana have become mainstream. Ideas in the works such as Google Duplex[1] serve to advance the capabilities of such agents in an effort to better assist people in their everyday lives.

Our vision for the next generation of recommender systems is that of a good friend — someone who understands a user's likes, dislikes, preferences and concerns — who is naturally also knowledgeable about the items to be recommended. Such a system's recommendations are not simple words or phrases, but rather a detailed recommendation that resembles what would have been made by a good friend who personally knows the user. We call them, *Explainable Recommendations*. Such recommendations should:

1. be able to provide detailed suggestions or directions to the user for a good experience with the item, taking into consideration, their personal preferences.
2. keep the best interests of the user in mind by pointing out causes for concern if they were to proceed with the recommendation.
3. include factual information about the item that are of interest to the user.
4. be able to elucidate not only *what* aspects of the item a user may like or dislike, but also *why*.
5. be structured in natural language.
6. be personalized in its style of prose as much as it is in the contents.

[1] https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html

The main difficulty in training systems that can produce such recommendations is the lack of availability of ground truth data. In this chapter, we present a benchmark dataset, which we call, *CMU Dataset of Explainable Recommendations*, the first of its kind to be created. It was authored by human participants (turkers) on Amazon's Mechanical Turk[2] platform, and therefore, all recommendations sound natural. The turkers were shown the actual reviews written by users for items and asked to rewrite them in the form of detailed recommendations. Most importantly, since the recommendation data is created from the original review, it maintains parity with the users' experience of the item, and hence, can be considered as the ground truth recommendation for that user-item pair.

Though related, our goal is different from a chatbot-style setup for dialog [191, 192, 193]. The primary objective of chatbots is to sustain a conversation with a user whereas our objective is to help a user make decisions about items. It would be good to have a system that could do both simultaneously, but at this time, our goal does not include conversations with the user. This dataset is better suited for building backend modules that can provide detailed recommendations satisfying user's queries and preferences, which could be relayed back to the user via a text, voice, visual or 3D interface.

We believe this dataset would constitute the first step towards designing and developing systems that can generate explainable recommendations as well as add to the driving force towards realizing the grand vision for AI.

This chapter is organized as follows. We discuss past research relevant to explainable recommendations in Section 5.1. Section 5.2 describes our setup for data collection on Amazon Mechanical Turk. The dataset itself is detailed in Section 5.3. Finally, we report the results of preliminary experiments in Section 5.4.

5.1 Related Work: Explainable Recommendations

In the past , there have been numerous attempts at generating explainable recommendations. Many of the explanations tend to show *how* the system arrived at that particular recommendation and does not attempt to explain to the user *why* they would like or dislike that item. i.e. their goal is the interpretability of the algorithm's decisions. For example, [176] proposes a joint model of matrix factorization and latent topic analysis, and describes how the rationale behind the predicted ratings could be made interpretable by utilizing content features. [58] offers textually and visually interpretable recommendations by identifying overlapping co-clusters of users and items. [130] models each user and item using concept vectors where, each dimension of the concept vector is a Wikipedia article or a Wikipedia category name, making the vector representation readily interpretable. [168] proposes a model named Tree-enhanced Embedding Method that combines the strengths of embedding based neural models and decision tree-based models. Their model has an easy-to-interpret attention network, making the recommendation process fully transparent and interpretable. Although such approaches help the system earn users' trust, it does not particularly assist users in making decisions.

Many other techniques proposed in the past sought to construct a summary or generate user reviews. Our approach proposed in Chapter 4 is one such method. A detailed discussion of past research on personalized review generation was discussed earlier in Section 4.1. In the absence of other mechanisms, showing a user what they may write about an item if they were to use it, is beneficial.

[2]https://www.mturk.com

Past embodiments of explainable recommendations have mainly been structured as discrete words and phrases. For example, [190] extracts explicit product features (aspects) and users' sentiments towards these aspects using phrase-level sentiment analysis. [132] proposes a social collaborative viewpoint regression (sCVR) method, for predicting item ratings based on user opinions and social relations. They use viewpoints, represented as tuples of a concept, topic and a sentiment label extracted from user reviews and trusted social relations, as explanations. [56] is a related method that ranks aspects for each user on a tripartite graph consisting of the users, items and aspects. [110] jointly models reviews and ratings using a Hidden Markov Model to provide explanations using words from latent word clusters that would explain important aspects of the item. [165] develops a multi-task approach, where two tasks that perform user preference modeling for recommendation and opinionated content modeling for explanation are integrated via a joint tensor factorization. Their method predicts aspects of an item that may be of interest to a user and displays them using simple manually created sentence templates. [27] proposes Attention-driven Factor Model (AFM) that uses an attention mechanism to estimate users' attention distributions on different item features. Such distributions serve as explanations for the recommendations produced by their model. [143] uses a probabilistic relational learning framework to generate the rules that can provide explanations of the form, "users who liked X also liked Y". They specifically studied cross-domain preferences of users.

All methods proposed in the past use data that consists of user and item content / meta-information, rating and reviews, and therefore, are limited in their abilities in generating truly explainable recommendations. There exists no dataset that provides explicit detailed natural language recommendations.

A related task that has been researched in the past is that of converting structured data into natural language text. For example, in [171], the authors introduce a dataset consisting of basketball game statistics and the corresponding game descriptions. They show that neural models can be trained to convert the statistics into fluent natural language text. Another example is the case of biography sentence generation from Wikipedia fact boxes proposed in [80] and [54]. In [72], the authors propose models for generating natural language text from database records. These datasets, although in natural language, are not suited for generating recommendations.

5.2 Amazon Mechanical Turk Setup for Data Collection

Amazon Mechanical Turk (MTurk), like we discussed in Chapter 4, is a convenient framework for collecting data requiring human intelligence, at scale. The explanation collection was structured as a review rewriting task on the MTurk's Requester website. This study was approved by CMU's Institutional Review Board (IRB) as STUDY 2018 00000337.

5.2.1 Instructions for the Study

Below are the instructions given to the turkers for rewriting reviews as detailed recommendations with explanations:

Task: You will be shown a review written by a user for an item. Your task is to read the review and

rewrite it in a form that would resemble a suggestion or recommendation that someone would have given to the user before he or she experienced that item. For example, suppose that Alice wrote, "I loved the chicken wings at American Wings ! They were creamy and tangy." If we knew that Alice would be writing this review if she were to visit the American Wings restaurant, an ideal recommendation/suggestion to Alice before she decided to visit the American Wings restaurant would probably be of the form, "You will love the chicken wings at American Wings. They are creamy and tangy."

Criteria: Your answers will be manually and/or programmatically inspected to check if you adhered to the instructions given below.

To successfully rewrite, please follow the instructions given below:

1. You must adhere to the information given in the review shown to you.
2. You must not add new information, which you may know already about that item.
3. You must include all information relevant to that item as described in the review and not omit anything relevant to the item.
4. You can and should exclude information from the review that is about that user's experience with anything not relevant to the item. E.g. "I was talking to Katie from work" can be excluded.
5. You will need to change the first person past tense of sentences in the input review to second person future tense in your rewritten form. E.g.1. "I liked it" would become "You will like it". E.g.2. "I ordered the pesto pasta. It was tasty." could be rewritten as, "You could order the pesto pasta. It is tasty."
6. When you read the review, try to understand what things or aspects this user cares about according to the review. E.g. one user cares about price whereas another user cares about quality.
7. From the review, also try to understand how those things or aspects are provided by the item, and whether it is to the user's liking or not. E.g. one user is happy that the restaurant is cheap and affordable, and therefore likes it, while another user is unhappy that it is cheap, worrying that it's of lower quality, and therefore does not like it.
8. From your understanding of the user's preferences and how it is reflected in the item, construct your recommendation, *assuming that the user is yet to use the item or go to the restaurant*.
9. If the user likes some aspects of the item but not other aspects according to the review, you must preserve that information in your rewritten form. E.g.1. a user liked the ambience but thought the menu was overpriced, your recommendation would be, "You will like the ambience but the menu is overpriced". E.g.2. a user wrote, "I won't be visiting FooBar anymore. It is a total waste of money", your recommendation could be, "Don't visit FooBar. It is a total waste of money", or "FooBar is a total waste of money".
10. Use grammatically correct sentences and punctuations.
11. Stick to the user's word choices. E.g. one user said, "The place is beautiful", while another user said, "The place is fantastic", use the words from the review when you rewrite ('beautiful' for the first user and 'fantastic' for the second).
12. Stick to the user's sentence ordering where applicable. E.g. one user wrote about price first and then ambience, while another user wrote about ambience first and price later. Use the

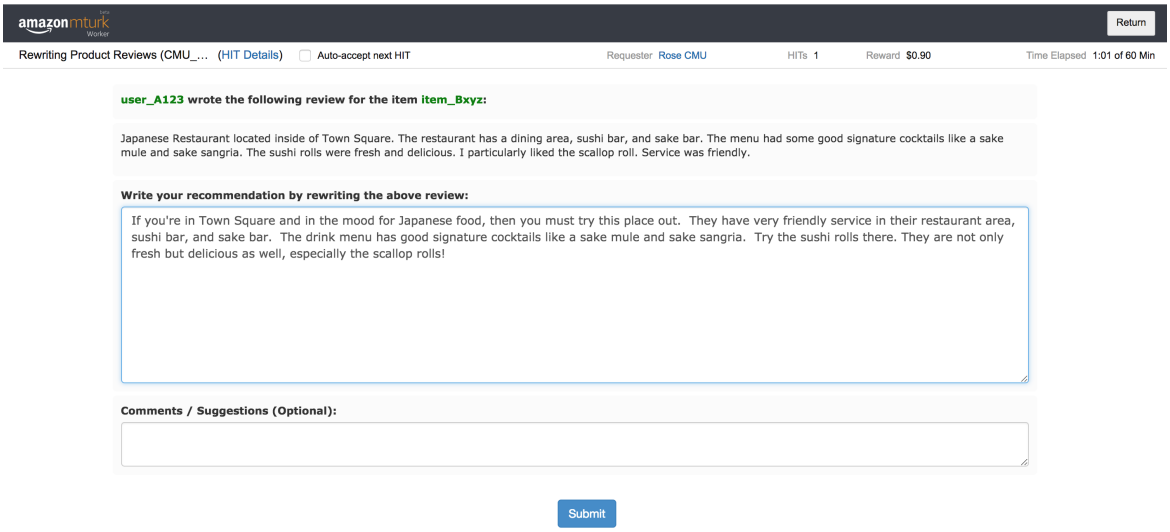

Figure 5.1: Sample Mechanical Turk Screen for the Review Rewriting task

sentence order from the review when you rewrite.

13. Include all factual information about the item/restaurant that is available in the review. E.g. "Beers on draft are around $4", "There is ample parking space", "This place gets crowded and noisy in the evenings", "The beef steak is juicy and tender", etc. must be included if the user wrote it in their review.
14. If the name of the item/restaurant is missing, you could describe it in a third person point of view. E.g. "They have music on patio". Don't include the anonymized id of the item.
15. Never use first person (e.g. I, we, us, my, our) point of view in your rewritten version. E.g.1. Don't write, "I think you will like...". Instead write, "You will like...". E.g.2. Don't write, "We ordered the brussels sprouts. It was mushy and terrible." Instead write, "Don't order the brussels sprouts. It is mushy and terrible."
16. Your rewritten text must look like a recommendation and not like a prophecy. For example, don't write, "You will order ramen", "You will go to Wild Wings this Sunday with a Yelp coupon". Instead, make it sound like a suggestion. E.g. "You could try the ramen". Or, you could write in passive voice. E.g. "Yelp coupons are accepted at Wild Wings".
17. Put yourself in the shoes of Amazon Alexa, Apple Siri, Google Home or any other of your favorite virtual assistants, and check if your answer sounds okay if the virtual assistant were to read it out to the user. E.g. Don't own the item/place by writing, "Visit our restaurant, you will like the ambience". Instead write, "You will like the ambience at this restaurant".

An example of the screen displayed to the turkers for rewriting reviews is shown in Figure 5.1. In addition to the above instructions, they were also supplied with sample rewritten reviews. Their queries and comments were incorporated into a Frequently Asked Questions section and a Notes section as the study progressed.

5.2.2 Specifics of the Study

HITs for the review rewriting task were released as batches of approximately 500 HITs each. Each HIT was worked on by only one turker. Once all the HITs in a batch are complete, the MTurk website provides a detailed comma-separated file containing all attributes of the HITs and their responses, which can be downloaded. We implemented a Python script that processes the comma separated file and flags suspicious cases. The Python script specifically looks for cases where first person pronouns are used in the rewritten recommendation. It also looks for occurrences of specific verbs in their past tense that are typically associated with a first person point of view in this domain. Examples include 'ate', 'ordered', 'waited', 'sat', 'went', 'lacked' etc. This list was updated throughout the study to improve the accuracy of the script. The flagged cases were manually checked to verify the turker's response. Although technically, one could look for any verb in its past tense to improve recall, it comes at the expense of precision. The script also flags cases where the rewritten text has less than half of the word count of the original text. Responses that were not flagged were automatically approved, although in each batch, we randomly sampled a few of the approved responses to ensure the integrity of our checking mechanism.

We also implemented a neural model that takes as input, the original and the rewritten reviews, and predicted whether it was correct or not. The neural model used `word2vec` to learn an embedding for the words and subsequently, produced two latent representations for each of the texts — one, using a BiLSTM and the other, using a CNN. A Factorization Machine regressor took these latent representations as input and predicted the probability of the rewritten recommendation being an acceptable answer for the original review. However, since it was tested during the early phases of this study, there were not enough training examples, and therefore, it failed to make any useful predictions.

Throughout the study, we used a tally system of `warn` and `reject` counts. If a turker's answer to a HIT was almost correct, but not fully correct, it would count towards their `warn` tally. These include cases where there are two to four first person pronouns, or words and phrases in their first person past tense form. It also includes cases where the turker forgot to include in their rewritten text, two to four key aspects and/or their reasons and descriptions mentioned in the original review. Anything more than the above would count towards their `reject` tally. Copy-pasting the input review as their answer is also a `reject`, unless the original review itself was in the form of a recommendation. The turker was paid for those HITs assessed as `warn` at the same rate as the correct ones, but was not paid for those assessed as `reject`. After each batch is evaluated, their tallies were updated. To continue in the study, a turker would need to maintain their `reject` count under 5 and their `warn` count under 10, which also served as an incentive for the turkers to answer correctly.

Before the turkers could work on the main task, they were required to pass a qualification test. The qualifier was structured identical to the main task, but was paid at a slightly lower rate. The turkers were required to work on 5 HITs from the qualifier task and had to get at least 3 of them correct to pass the test. If they attempted more than 5 HITs, they were required to obtain a success rate of at least 60%. The criteria for rejection was relaxed to include only verbatim copy-paste of the original review as their answer, to minimize the impact on their overall approval rating on MTurk. However, on hindsight, the qualifier should have been unpaid and the turkers should have been restricted to attempting at most 5 HITs to reduce spamming. There were large number of participants who attempted substantially more than the required 5 HITs in the qualifiers and got paid for most of

them even though their answers did not make the cut. In one instance, that number was as high as 94. This behavior was partly because we were rejecting only extreme cases and counting the others towards their `warn`, which were subsequently paid out.

Due to restrictions enforced because of GDPR[3] being adopted in certain countries, CMU's IRB required us to select only participants located in the USA. This was enforced using the `Location` attribute of the turkers as provided by MTurk. Otherwise, the participation was open to turkers who had an overall MTurk approval rating of at least 70%. 534 turkers took the qualifier test, which was live for 4 days. Their responses to the qualifier HITs were checked manually and only 232 participants passed the test. The manual checking of the qualifier HITs was helpful in refining the Python script that was later used in the main study.

To determine the pay rate, we timed ourselves while manually rewriting 10 reviews and aimed at a pay of approximately US $10.00 per hour, which is higher than the prevailing US Federal Minimum Wage of $7.25.[4] The pay per HIT was set to US $0.70 for the qualifiers and US $0.80 for the main study. The pay per hour for the main study worked out to US $11.87 on average. There was also an additional 20% fee to be paid to Amazon. As an additional reward to turkers who had consistently performed well and maintained their `warn` and `reject` tallies below the required counts, the last few batches totaling to about 2800 HITs were paid at a rate of US $0.90 per HIT. Therefore, the hourly pay for those HITs are higher than the average figure reported above. During the study, it was also brought to our attention that many turkers hoarded HITs using automatic tools. Such tools would accept a specific number of HITs simultaneously on that turker's behalf, thereby ensuring that the turker would have those many HITs to work on before they were all claimed by other turkers. However, we noticed that when turkers were holding on to multiple HITs simultaneously, unfortunately, it increased the average time per HIT reported by the MTurk website because it is computed as the time of accept to time of submission, which is larger if that HIT was accepted, but not worked on right away. The increase in the average time was observed in real time on the MTurk website. We also noticed that the number of `warns` issued also increased for the HITs submitted later because the turkers were under time pressure to submit them before the allotted time ran out. To discourage this behavior, we reduced the allotted time per HIT to 10 minutes from the earlier 30 minutes and also requested the workers to accept the HITs only when they were ready to work on them, not before. The average pay per hour reported above is for the setting after making this change.

Some of the reviews are substantially long. A couple of batches into the study, we received feedback about the length of the reviews, and subsequently posted only reviews whose word count was between 100 to 250. But if there were not enough reviews for a particular user in the dataset, that number was lowered to 75 words and if required, to 50 words.

As the study progressed, the number of eligible participants according to the tally system dropped from 232 to 151 and further down to 75 towards the end of the study. Each batch typically has 500 HITs and each turker got to work through only about 2 (when there were a large number of participants) to 10 (towards the later part of the study) HITs before all HITs of that batch were exhausted. To reduce fatigue, we also spaced out the batches by at least a couple of hours when there was more than one batch released on the same day. The schedule for each day was also announced beforehand to the turkers using a mailing list (signing up for the list was optional). The full study was completed

[3]https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

[4]https://www.dol.gov/general/topic/wages/minimumwage

in a span of 12 days.

Although this task appears at first to be of high cognitive load, it is in fact quite straightforward. The turkers read each sentence of the review and determine whether it is relevant to the item being reviewed or not. If it is, then they change it from its first person point of view to its second person point of view taking care that it still looks like a suggestion, a recommendation or a factual information. They repeat this for all sentences in the review in the order in which they appear. This process was also independently suggested to us by turkers after they worked on a few HITs. The turkers may also optionally move sentences around so that the whole text looks more natural, although we did not require them to do that nor did we specifically check for that. Since this task does not involve them to summarize the information, it is actually easier that it appears to be. Turkers also improved their speed once they got accustomed to the task.

5.3 CMU Dataset of Explainable Recommendations

Our source data is the Yelp challenge[5] dataset released in 2018, containing 5M reviews and ratings of businesses by more than 1M users. We first constructed a 40-core subset of that dataset, referred to as `Yelp18 40-core`, that was also used for experiments in Chapter 4. Of the 1,002 users and 978 businesses in that subset, we randomly sampled 1,000 users and 900 businesses. That yielded 66,886 reviews written by the shortlisted users for the shortlisted items. We constructed a subset of 10,000 reviews for the MTurk review rewriting task by randomly sampling 10 reviews from this subset for each of the shortlisted users. We inspected the rewritten reviews manually and programmatically, and discarded the ones that do not adhere to our requirements. That gave us 9,773 detailed recommendations.

Sample recommendations from the collected data are given in Table 5.2 and Table 5.3. As can be observed from the tables, the recommendations are realistic and maintain parity with the actual experience of the user with that item. They also exhibit the characteristics of explainable recommendations that we delineated in the introduction of this chapter. By adhering to the users' choice of words and phrases, these recommendations are also personalized to the user in its style of delivery as much as it is in content. In addition to providing training data for learning to generate detailed recommendations automatically, this dataset also offers a parallel corpus for review to explainable recommendation. A neural translation model with a copy mechanism [66, 94] can be easily trained using this parallel corpus and subsequently applied on reviews from other datasets to obtain training data for generating explainable recommendations in those domains.

5.4 Preliminary Experiments

For our preliminary experiments with the explainable recommendation dataset, we ran the TransNet model that was discussed in Chapter 4 as well as CF-GCN [116] using the `medium` configuration described in Section 4.5, and evaluated their performance using metrics that were detailed in Section 4.4. Instead of feeding reviews as input, we fed the recommendations as input. The results of these experiments are shown in Table 5.1.

[5] `https://www.yelp.com/dataset_challenge`

Metric	TransNet	CF-GCN
Word Level Perplexity	39.38	36.49
BLEU overall	12.78	6.67
BLEU 1	15.82	8.49
BLEU 2	8.99	4.57
BLEU 3	27.28	11.85
BLEU 4	43.41	18.22
METEOR	9.01	7.00
Precision	36.42	50.66
Recall	19.69	13.60
ROUGE-1	22.42	21.59
ROUGE-2	3.54	4.12
ROUGE-3	0.76	0.97
ROUGE-L	6.96	7.26
ROUGE-S2	3.16	4.02
ROUGE-S3	3.33	4.31
ROUGE-S4	3.65	4.49
Input Sensitivity	54.92 %	48.85 %

Table 5.1: Preliminary results on generating explainable recommendations

As can be observed from the table, although CF-GCN achieves a lower perplexity, TransNet's output is rated the best according to BLEU, METEOR and IS scores. TransNet's performance is higher for ROUGE-1, but CF-GCN performs better on all the other ROUGE types. There is a switch in the relative performance of the two models on ROUGE scores, when compared to their performance on the `Yelp18 40-core` dataset of reviews. In the former, CF-GCN has better ROUGE scores while in the latter, TransNets was better. This change could be because the explanations dataset is less noisy and more clean than its review counterpart — when turkers rewrote reviews, they were asked to ignore details irrelevant to the item being discussed. Also, the overall language used in the former is slightly more standardized than that in the latter because turkers rewrote them in a certain way and corrected grammatical errors, uncommon word usage and typographical mistakes.

5.5 Contributions

In this chapter, we presented a new dataset called "CMU Dataset of Explainable Recommendations" that provides close to 10K real world ground truth recommendations for 1,000 users and 900 businesses. These were written by human participants on Amazon Mechanical turk and therefore, resemble natural recommendations and explanations that people provide to each other. Most importantly, since these were constructed from the original reviews written by the users, the recommendations are faithful to their actual experience with the item. After describing the procedure for constructing

the dataset, we also discussed our preliminary experiments and results obtained on the dataset. We hope that this dataset will pave the way for future research in generating explainable recommendations.

Original Review	Recommendation
Japanese Restaurant located inside of Town Square. The restaurant has a dining area, sushi bar, and sake bar. The menu had some good signature cocktails like a sake mule and sake sangria. The sushi rolls were fresh and delicious. I particularly liked the scallop roll. Service was friendly.	If you're in Town Square and in the mood for Japanese food, then you must try this place out. They have very friendly service in their restaurant area, sushi bar, and sake bar. The drink menu has good signature cocktails like a sake mule and sake sangria. Try the sushi rolls there. They are not only fresh but delicious as well, especially the scallop rolls!
Now that was a show! Was I a fan of the Jabbawockeez during their quick rise to fame on the show ABDC? No. I cannot get into that show, so don't think it's a necessity to enjoy this show. Am I a big hip hop fan? Yes, but still...not a necessity. Do I like getting my money's worth? Hell yes! That's what this show is. SO worth it! Entertaining from top to bottom. The music selection was varied, and to my surprise, was much more than just hip hop. The Jabbawockeez are true performers and masters of their craft in that they can dance to just about anything, and dance they did.	Now, this is a show. You will get your money's worth, that is what this show is. So worth it! Entertaining from top to bottom. The music selection is varied, much more than just hip hop. The Jabbawockeez are true performers and masters of their craft, they can dance to just about anything, and they will.
Grand Lux Cafe is conveniently located on the casino floor at The Venetian/Palazzo; their brunch menu is priced similarly to Las Vegas 24-hour cafes; I did a build-your-own omelette, which consisted of bacon, cheddar cheese, onion, and assorted peppers (you have the option to add up to 4 ingredients); the omelette was good (nicely cooked and all of the ingredients worked well together); the omelette came with crispy shredded hash browns and good and hearty wheat toast (a toasted English muffin is available as an alternative to the toast); the brunch was decent and filling; you can get a more upscale and more delicious brunch at Bouchon (also located at this hotel), but that restaurant is harder to get to, not as filling and more expensive Like almost all of the major resort hotels on The Strip, The Venetian and Palazzo offer complimentary self and valet parking.	Grand Lux Cafe is conveniently located on the casino floor at The Venetian/Palazzo. Their brunch menu is priced similarly to Las Vegas 24-hour cafes. You can make a build-your-own omelette with up to four ingredients. The omelettes are good (nicely cooked and the ingredients work well together). The omelette comes with crispy shredded hash browns and good, hearty wheat toast or toasted English muffin. The brunch is decent and filling. You can get a more upscale and more delicious brunch at Bouchon, also located at this hotel, but that restaurant is harder to get to, not as filling and more expensive. Like almost all of the major resort hotels on The Strip, The Venetian and Palazzo offer complimentary self and valet parking.
Gorgeous new casino that's located far off the strip. It's clean, it's modern, it's full of great employees. We didn't stay at this hotel, but came for the buffet. If you're looking for a good value for your money, but don't want to sacrifice on style, this is the place to stay. The air isn't filled with smoke (not yet anyway) and the casino itself isn't jam packed to the brim. If you don't mind a 15 minute drive to the strip, or if you actually want to stay away from the strip, this is a must for you.	This gorgeous new casino is located far off the strip. It is clean, modern and full of great employees. If you are looking for a good value for your money, but don't want to sacrifice on style, this is the place to stay. The air is not filled with smoke and the casino itself is not jam packed to the brim. If you don't mind a 15 minute drive to the strip, or if you actually want to stay away from the strip, this is a must for you.

Table 5.2: Sample reviews rewritten as recommendations

Original Review	Recommendation
I let my non-Yelping brother-in-law select where we would have dinner on his final night in Las Vegas, and his choice, Sushi Roku, did not disappoint. He doesn't eat sushi but his lady and I both love it, and we all left Sushi Roku very happy with our meals. Sushi Roku is located on the third floor of The Forum Shops at Caesars Palace, almost directly above the valet (so really easy to find). The space is sleek and sexy, and we had a great view of the Strip from our booth. Service was excellent all night; patient (we are a chatty group and I think our waiter must have come back to take our order 5 times before we decided on anything) accommodating to special requests and able to answer the plethora of questions we had about the menu. Highlights from our meal would be: Brussel Sprout Chips with truffle oil - addictively delicious Spicy Pork Belly Fried Rice - My brother in law was so in love with this dish he had the waiter bring out another order before he was even halfway finished. He was kind enough to share the pork belly which was absolutely divine. American Wagyu Beef Skewer - The pork belly stole the spotlight, but this skewer was a close second. Melt-in-your-mouth deliciousness. Albacore Sashimi - Super fresh and served with a nice, light ponzu sauce I didn't see the check so I can not tell you what our meal set us back, but based on how happily my brother-in-law paid it, our meal was obviously worth every penny to him and that's all that really matters to me. I will absolutely return to Sushi Roku in the future!	Sushi Roku does not disappoint, you will leave happy with the meal. Located on the third floor of the forum shops at Caesers Palace, almost directly above the valet and really easy to find. Sleek and sexy with a great view of the strip. Service is excellent, patient and accommodating to special requests. They are also able to answer questions about the menu. Highlights are brussels sprout chips with truffle oil, addictively delicious. Spicy pork belly fried rice is absolutely divine, you will be in love with the dish. You can opt for the American Wagyu Beef skewer but the pork belly steals the spotlight with the skewer a close second. Melt in your mouth deliciousness. May be you want the Albacore Sashimi, super fresh and served with a nice, light ponzu sauce. Worth every penny.
I haven't been back to Pho Thanh Huong since I moved to the southwest area 5 years ago. Since I was in the mood for Pho and couldn't find any good Pho in the chinatown/west part of town. I yelped and saw a couple posting about PTH. Anyway, a bunch of my family members came back over the week and tried different things and here what we found: - Pho is similar to other places I found on the west side of town. The broth is below average and tasted just like someone dipped a pho flavor bag into hot water. It has no natural sweetness at all. - Bun thit nuong (grill pork vermicelli (not sure if this is correct spelling but too lazy now to look it up) was good. - Except soda that came in a can, all their specially made drinks weren't good. - The best thing I found at this place is their grill pork sandwich. It was very very good. I would drive all the way here just for this sandwich. By the way, some yelpers mentioned about the weird smell of this place, which I experienced while I was there. It has that stale cigarette smell that you normally experience when visit really old casino.	You may not be impressed with the Pho at Pho Thanh Huong. The broth is below average and tastes just like someone dipped a pho flavor bag into hot water. It has no natural sweetness at all. Try the Bun thit nuong (grilled pork vermicelli) it is good. The soda comes in a can. Don't order the specially made drinks, they are not good. The best thing you can find is the grilled pork sandwich. It is very very good. You'll be willing to drive here just for the sandwich. You might think the place has a weird smell, it has that stale cigarette smell that you normally experience when you visit a really old casino.

Table 5.3: Sample reviews rewritten as recommendations

Chapter 6

Conclusions and Future Work

Our vision for the next generation of Recommender Systems is that of a good friend — one who understands the user's likes, dislikes, preferences and concerns, and at the same time, knowledgeable about the items being recommended. Such a system's recommendations are not simple words or phrases, but rather a detailed recommendation akin to that given by close friends. We call them, *Explainable Recommendations that are Explanatory*, or *Explanatory Recommendations* for short.

Although providing explanations for recommendations have been shown in the past to serve multiple purposes, almost none of the recommender systems show them in practice. In designing and building such systems, the main challenges we encountered were that the type of explanations that can be generated is limited by two aspects — first, the type of the underlying model, and second, the type of available training data.

In this thesis, we experimented with two classes of models: Knowledge Graph (KG) based and Neural models. In both cases, we proposed approaches to produce a recommendation as well as generate an explanation, both of which were personalized to the user. In moving from KG to a neural setting, our input data changed from discrete content in the form of metadata and entities, to free form text in the form of reviews. Accordingly, our explanations also changed from discrete entities to detailed natural language texts.

We proposed an oracle-student architecture, called *Transformational Neural Network* — TransNet for short — that learned how to transform latent representations of users and items into that of their joint review. We showed that such a network architecture could be used to not only predict the ratings but also generate user's reviews for items. We explored different variants of that architecture by replacing one or more of its components with those that are known to be more successful in related tasks. However, contrary to our expectations, the standard setting of TransNet was empirically observed to be better than or at par with all the variants. Some settings did produce slightly better results but only with substantially more training data. The number of possible variants are practically countless and it is highly probable that there exist variants that are more efficient and better at generating reviews than the standard TransNet. However, we leave that for future research.

We compared the performance of our TransNet model to that of the state-of-the-art models extensively using both automatic and human evaluations. We demonstrated that TransNet is the better model when the amount of training data is less or sparse. Part of the story was also about the pitfalls of relying solely on automatic evaluation metrics in measuring the performance of generative models. Metrics such as Word Perplexity and BLEU, although very popular in the text generation

and translation communities, can be terribly misleading in certain settings. We observed that in many cases, the best models according to these metrics were not able to produce any useful text. Two other metrics, METEOR and ROUGE, that are also popular in translation and summarization communities, seemed to be more correlated to our observations about the generated reviews. While they were able to correct many of the relative rankings provided by Word Perplexity and BLEU, there exists datasets where not all of the relative ordering provided by them corresponded to the actual observations. As a simple sanity check, we also proposed a metric called *Input Sensitivity*, that measures an aspect of generative models which is not otherwise measured by these metrics. By using it in conjunction with the other automatic metrics, we will be able to discern these anomalous situations and take actions such as using human judges for evaluating the performance.

To measure the goodness of generated reviews using human judges on Amazon Mechanical Turk, we devised an aspect-reason tagging scheme. We also separated liked and disliked aspects to reduce the cognitive load on the turkers (human judges). In addition to the overall performance of the models, we also gained new insights into their behaviors that were not immediately apparent from our automatic evaluations. For example, these models appeared to be better at generating positive aspects about items than their negative aspects, which may also be a reflection of the underlying training data used. It was surprising that the such a disparity existed even in the case of the beer dataset, which is one of the best datasets available for this task in terms of being clean, detailed, structured and dense. Also, it became clear that there was plenty of room for improvement in developing models for generating reviews especially when the training data is less and sparse.

In our quest for explanatory recommendations, we progressed from selecting the most likely review to generating user's review and to finally generating detailed suggestions personalized to the user. To accomplish the latter, we created a new benchmark dataset, called *CMU Dataset of Explainable Recommendations*, the first of its kind for personalized explanations, written by human authors on the Amazon Mechanical Turk platform. The turkers were shown the original review written by a user for an item and were asked to rewrite it in the form of a detailed recommendation. Therefore, by design, the rewritten recommendation maintains parity with the user's actual experience with the item. We hope that the availability of this dataset will galvanize existing researchers in this area as well as new researchers to work towards generating explainable recommendations that are explanatory.

There are a number of interesting directions of future research that stem from the studies and results reported in this thesis. The most important of all is furthering our understanding of explanatory recommendations. Unlike explainable recommendations that are interpretable and those that serve other purposes, explainable recommendations that are explanatory have not been studied as much. It is imaginable that not too far in the future, we would be interacting and collaborating with AI agents in various disciplines. In such a setting, explanations that are explanatory would be required of these agents, not just desirable.

In this thesis, we moved from KG-based models that operate on discrete entities to neural models that operate on natural language text. However, there is a middle ground that we did not explore due to time constraints — an amalgamation of the two types of models. There have been some recent research that attempts a graph convolutional approach to image recommendation [184], matrix completion [109] and influence prediction on social networks [129]. However, using reviews, which is a joint feature of the user and the item, remains largely unexplored. In our experiments, we observed that neural models typically require large amounts of training data and when there is not enough

data or if the available data is too sparse, their behavior can become unpredictable. We also observed that KGs are most valuable when there is only a limited amount of training data and their utility diminished when there is enough training data. i.e. each architecture is stronger in settings where the other is weaker. Therefore, we believe that a joint architecture that plays to the strengths of both KGs and neural models would be better than each of them individually.

It was evident from our experiments that automatic evaluation metrics have limitations, some more so than others. However, getting human judgements for every evaluation is cumbersome, time consuming and expensive. It is also not suitable for measuring incremental improvements during the intermediate stages of model development. Therefore, it is important to develop metrics for measuring the performance of (conditional) natural language generative models that take into consideration, their specific failure modes such as generating text without regard to their input and generating repetitive text. These are special cases that have not been carefully considered when defining the existing metrics. It is also not clear when an automatic metric can be trusted as a proxy for human evaluations. There is definitely a need for a new metric that is specifically designed for evaluating generative models.

By creating a new dataset for explanatory recommendations, we have facilitated the development of models that can generate detailed suggestions in that specific domain. We believe that its utility is beyond that one domain. This dataset provides parallel data that can be used to develop models for style transfer such as [156], that convert reviews to detailed recommendations, thereby mimicking the process we achieved using turkers.

When one imagines all that is possible in the near future with the rate at which the field of AI in general and Deep Learning in particular have been accelerating, one realizes that this thesis has only scratched the surface of a tiny subfield contained within those broader fields. However, we will be content knowing that our humble efforts today have the potential to inspire bigger changes tomorrow for a better future for all of humanity.

Acknowledgements

This thesis would not have been possible without the guidance, assistance, support and prayers of many. I would like to thank them in my own small way in this thesis as a token of my appreciation for all that they have done for me and for helping me bring this work to fruition. This acknowledgement section is necessarily long. For a TLDR, please refer to the first two sentences of this section.

I would like to express my sincere thanks and gratitude to my advisor, Prof. William Cohen, for being the Best. Advisor. Ever. His excellent guidance helped me streamline my thoughts and ideas into research work that led to coherent papers and chapters of this thesis. His uncanny intuition about why an idea works a certain way or did not work at all has definitely saved me quite a lot of tedious debugging. I am especially thankful to him for giving me the latitude and freedom to experiment and explore my own ideas, even if they diverged from his immediate plans and requirements. His patience is practically unending, and although I have given him plenty of reasons to be frustrated, he has always taken it in his stride. I thank him for being a wonderful advisor, mentor and role model. I am proud to have been his student!

I would also like to thank my committee members, Prof. Maxine Eskenazi, Prof. Ruslan Salakhutdinov and Prof. Jure Leskovec, for their invaluable comments and suggestions during the thesis proposal and the subsequent research efforts. Prof. Eskenazi also helped design one of the MTurk studies and provided a lot of guidance on how to successfully get the study done, from her vast prior experience with the platform. I am also thankful to her for checking on me from time to time to make sure that everything was alright. Her homegrown lavender decorated my office for about four years.

Thanks are also due to my InMind family of professors and students who gave me a sense of teamwork and a lofty vision to work towards. The overall goal of this thesis was, in part, shaped by that vision. I applaud the leadership of Prof. Justine Cassell and Prof. Tom Mitchell in taking on the big challenge as well as encouraging a bottom-up approach to research and development. I appreciate the opportunities that I got for collaborating with Prof. Maxine Eskenazi, Prof. Alan Black and Prof. Emma Brunskill, and their students, Ting-Yao Hu and Yulun Du. I am fortunate to have had an ever-enthusiastic team of Dr. Igor Labutov, Dr. Timo Baumann, Dr. Florian Pecune and Dr. Yoichi Matsuyama who were mighty determined in getting a reluctant SARA (their virtual assistant) to speak to my then half-baked recommender module. Their camaraderie certainly did reduce my anxiety about demos that break at the last minute. I would also like to thank Dr. Oscar Romero and Sushma Akoju, who were always available to help out with any architecture or integration issues. This research was supported in part by a generous funding from Yahoo! (now Oath/Verizon) through the CMU-Yahoo InMind project. I also appreciate their biannual retreats that provided a venue for grasping the full extent of the project's goals and for honing my presentation skills. I am grateful to Dr. Donald McGillen for being especially supportive of our research efforts.

My fellow advisees, Fan Yang, Bhuwan Dhingra, Zhilin Yang and others, have together made the weekly group meetings interesting by discussing their ideas as well as the most recent developments in Deep Learning. Participating in those discussions and answering their questions have served to keep me on my toes and made me strive to keep myself up-to-date. I can never thank Kathryn Mazaitis enough for her assistance and support with the lab machines. She knows more (perhaps everything) about the ProPPR codebase than I ever will. Her can-do attitude is certainly praiseworthy.

I am also grateful to Dorothy Holland-Minkley, Stacey Young and Sandra Winkler for taking care of all the administrative paperwork and travel reimbursements on time. They were also accommodating of my last minute requests, which have at times been insanely last minutey and totally unanticipated. Mary Jo Bensasi has been exceptionally considerate with the office and common spaces in the Gates Hillman Centre.

My officemates and office friends over the years, Dr. Hyeju Jang, Qinlan Shen, Dr. Pallavi Baljekar, Fatima Al-Raisi, Dr. Behnaz Nojavanasghari and Yipei Wang have been delightful to talk to. Lengthy discussions were made about topics ranging from politics to fashion, in addition to more research worthy ideas. Watching them work diligently provided me with the much needed motivation to delve into my own research with renewed passion.

My friends at CMU over the years, Dr. Bhavana Dalvi Mishra, Dr. Meghana Kshirsagar, Anjali Menon, Dr. Ruta Desai, Dr. Sunayana Sitaram, Dr. Pallavi Baljekar, Anuva Kulkarni and Dr. Derry Wijaya, have continuously strived to make me feel at home. Together, we enjoyed organizing, attending and celebrating birthday parties, picnics in the park, Diwali get-togethers, late-night movie screenings, ice-skating, snow-boarding, rock climbing, kayaking, farm visits and bachelorette parties. Dr. Arun Srivatsan, Vivek Ramachandran and Dr. Aditya Mishra also helped out with many of the outdoor activities.

I am immensely grateful to my loving parents-in-law, Mrs. Maria Jacob and Mr. Jacob Maliakal, for being always encouraging of all my endeavors with words of support and constant prayers. I respect their talent for elaborate planning around multiple constraints, and hope to be able to emulate them one day. I also admire how energetic, enthusiastic and astute they are with regard to most things, even the highly monotonous and tedious ones. I am also thankful to my in-law family, Sanju Maliakal, Jim Maliakal and Gita Painadath, and their kids, baby Lisa and David Maliakal, for their prayers and support. Sanju has also been an unsuspecting recipient of my culinary creations that are mostly experimental.

Words cannot express how thankful I am to my uncle, Dr. Paulson Puthur, my aunt, Mrs. Sangeeta Paulson, and my cousins, Jennifer and Jessica Paulson for being the best hosts ever during my numerous visits to the Bay Area. I have been ecstatic every time to find that they have prepared my favorite Kerala food delicacies. It felt like home. I also loved the long road trips with them as well as shopping with my cousins. Their words of encouragement and prayers mean a lot to me.

I am also extremely grateful to our aunt and uncle in New York, Mrs. Kochurani Sunny and Mr. Sunny Matthew, for taking us under their wings and for looking after us in a number of ways, small and big. We loved their impromptu lunch and dinner invitations over weekends because it meant that she has prepared something special. We were also delighted to be part of the large family gatherings during Thanksgiving, Christmas and Easter, that are reminiscent of our homes during those festivities.

I would like to express my sincere and heartfelt thanks to my uncle, Fr. Johnson Puthur, and

my grandaunts, Sr. Lawrencetta A.C., Sr. Theres Matthew A.C. and Sr. Maria Amala A.C., for their unceasing prayers and words of encouragement. I really appreciate how they check on me regularly to make sure that my ventures are on track.

Thanks are also due to my aunt Mrs. Daisy Puthur, my uncle Mr. Joy Therukattil, my uncle Col. Lawrence Kanjirathinkal, my cousins, Cmde. Ignus Kanjirathinkal and Dr. Rose Mary Lawrence Kanjirathinkal, and the rest of my extended family and cousins, for their words of support and encouragement, and for having confidence in me.

I would like to dedicate this thesis to my loving maternal grandparents, Mr. Devassy Puthur and the late Mrs. Thressiakutty Puthur, and to my loving late paternal grandparents, Mr. Inasu Kanjirathinkal and Mrs. Kochurosa Kanjirathinkal, whose memories will be always with me.

This thesis is dedicated to my ever-loving parents, Mrs. Lisy Puthur and Prof. Allesu Kanjirathinkal. Although I officially started my Ph.D. studies only four years ago, its seeds were planted a long time ago when I was a small baby keeping my Dad awake while he was writing his doctoral dissertation at IIT Kanpur. Countless number of times have I asked for his advice and drawn from his experience as a grad student with a family to take care of. Knowingly or unknowingly, he got me interested very early on in research in general and engineering in particular, when he let me tinker with his tools and watch him devise different mechanical gadgets out of everyday objects. Going back to the basics is an important troubleshooting technique that I picked up from him. Watching him redesign and refine his creations when they failed, and later, assisting him in these tasks, served as a sort of apprenticeship for me and has benefited me in my own research later. A side effect of this was that I did not learn to cook until I came to the US — thank God for YouTube video recipes. I am grateful for my dearest Mom who is always loving and caring. I truly appreciate the fact that she mostly never tried to impose her rules on me, even though her students consider her to be a strict and uncompromising teacher. Left to my own devices, I enjoyed the freedom to explore and learn anything and everything that piqued my curiosity. I would like to express my heartfelt thanks to her for her unceasing prayers, and for teaching me to trust in God to lead me when the path seems crooked and the journey, tiring. I am immensely grateful to my parents for the wisdom and life lessons that both of them together have imparted in me. I am indebted to them for all that they have done and continue to do for me.

This thesis is equally dedicated to my beloved husband, Bejoy Maliakal. He has shared in all my little joys and mighty pains that came with wins that appeared to be small and despairs that appeared to be endless in my life as a grad student. He is always enthusiastic to accompany me to conferences and other out-of-town presentations, and drive me around the town exploring new places to shop, interesting cuisines to experiment with, and beautiful sunsets to watch. He has always been there to provide me emotional support and some much needed pep talk whenever I was a nervous wreck, which was usually before giving a talk to a large audience. For most part of my Ph.D. studies, we were in two different cities (New York and Pittsburgh) and had to shuttle between the two. I found these frequent travels, long and tedious. Whenever I complained about the distance, he would smile and joke about getting into his car right away and driving all the way to Pittsburgh, which I believe he would have, had I not talked him out of it. The one time he changed his usual answer was to ask me to take the next flight out of Pittsburgh to New York, which I gladly did! He is also happy to do most of the heavy lifting with regard to household chores when I am glued to my chair typing this thesis or pulling all-nighters to meet deadlines. I have also banked on his English language skills when I volunteered him to proofread parts of this thesis. I am extremely thankful to him for putting up with

me — an adamant and impatient grad student with crazy ideas, unrealistic plans and lofty dreams. I'm not sure if he was fully aware of what he was signing up for when he said 'I do', but I am glad that he did !

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org. 4.5

[2] Behnoush Abdollahi and Olfa Nasraoui. Explainable restricted boltzmann machines for collaborative filtering. *CoRR*, abs/1606.07129, 2016. URL `http://arxiv.org/abs/1606.07129`. 2.2.2

[3] 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. 2.2.2

[4] 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. 3

[5] 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`. 4.1.4.1

[6] 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. 3, 3.1.2

[7] Nadav Kiril Altshuler and David Sarne. Modeling assistant's autonomy constraints as a means for improving autonomous assistant-agent design. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018*, pages 1468–1476, 2018. 5

[8] Andrew Kachites McCallum. MALLET: A Machine Learning for Language Toolkit. 2002. 2.4.2

[9] Authors anonymized. Inmind movie agent - a platform for research. Under Review, 2017. 2.6

[10] 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. 3.1.3.2

[11] Amos Azaria and Jason Hong. Recommender system with personality. In *Proc. RecSys*, 2016. 5

[12] 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. 2.2.1, 2.2.2

[13] 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

[14] 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. 3, 3.1.2

[15] 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. 3, 3.1.1

[16] 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. 4.1.1

[17] 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. 2.2.1

[18] Mustafa Bilgic and Raymond J. Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization Workshop*, January 2005. 2.2.2

[19] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. 3.1.1

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

[21] 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. 3.1.3.1

[22] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998. 4.1.2

[23] Rose Catherine and William Cohen. Personalized Recommendations Using Knowledge Graphs: A Probabilistic Logic Programming Approach. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 325–332, 2016. 2

[24] Rose Catherine and William Cohen. TransNets: Learning to Transform for Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 288–296, 2017. 3

[25] Rose Catherine and William Cohen. TransNets for Review Generation. In *Proceedings of the*

6th International Conference on Learning Representations (ICLR) Workshop, 2018. 4

[26] Rose Catherine, Kathryn Mazaitis, Maxine Eskénazi, and William W. Cohen. Explainable Entity-based Recommendations with Knowledge Graphs. In *Proceedings of the Poster Track of the 11th ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 28, 2017.*, 2017. 2

[27] Jingwu Chen, Fuzhen Zhuang, Xin Hong, Xiang Ao, Xing Xie, and Qing He. Attention-driven factor model for explainable personalized recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018*, pages 909–912, 2018. 5.1

[28] 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. 3.1.2

[29] 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. 4.6.5

[30] 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. 4.1.1

[31] Zhiyong Cheng, Ying Ding, Xiangnan He, Lei Zhu, Xuemeng Song, and Mohan S. Kankanhalli. Aˆ3ncf: An adaptive aspect attention model for rating prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 3748–3754, 2018. 3.1.1

[32] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011. 3.2.2

[33] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. Automatic generation of natural language explanations. 2017. 4.1.3

[34] 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. 3.1.2

[35] Michael T. Cox. Perpetual self-aware cognitive agents. *AI Magazine*, 28(1):32–46, 2007. 5

[36] 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. 4.1.4.3

[37] 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. 2.2.1.1

[38] Darryl N. Davis. Cognitive architectures for affect and motivation. *Cognitive Computation*, 2 (3):199–216, 2010. 5

[39] Luis de Campos, Juan Fernández-Luna, Juan Huete, and Miguel Rueda-Morales. Combin-

ing content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *Int. J. Approx. Reasoning*, 2010. 2.2.1

[40] 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. 4.4, 4, 4.7.3.5

[41] 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. 4.1.2

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

[43] 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

[44] 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. 3, 3.1.2

[45] 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. 4.1.4.3

[46] Kavita Ganesan. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. 2015. 4.7.3.6

[47] 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. 4.1.1

[48] 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

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 3.3.5

[50] 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. 4.6.5

[51] A. Graves and J. Schmidhuber. Framewise phoneme classification with Bidirectional LSTM networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052 vol. 4, 2005. 4.6.1

[52] 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. 4.1.1

[53] 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. 2.2.1

[54] 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. 4.1.4.1, 5.1

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

[56] 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. 4.1.2, 5.1

[57] 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. 3.1.2

[58] Reinhard Heckel and Michail Vlachos. Interpretable recommendations via overlapping coclusters. *CoRR*, abs/1604.02071, 2016. 5.1

[59] J. Herlocker, J. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *CSCW*, pages 241–250, 2000. 2.2.2

[60] 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. 3.1.3.1

[61] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, November 1997. 3.1.2

[62] Julia Hoxha and Achim Rettinger. First-order probabilistic model for hybrid recommendations. In *Proc. ICMLA '13*, pages 133–139, 2013. 2.2.1

[63] Guangneng Hu, Yu Zhang, and Qiang Yang. Mtnet: A neural approach for cross-domain recommendation with unstructured text. 2018. 3.1.2

[64] 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. 3.1.3.1

[65] Mohsen Jamali and Martin Ester. Trustwalker: A random walk model for combining trust-based and item-based recommendation. In *Proc. KDD*, 2009. 2.2.1

[66] Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. 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*, pages 1–10, 2015. 5.3

[67] Xiaotian Jiang, Quan Wang, Baoyuan Qi, Yongqin Qiu, Peng Li, and Bin Wang. Attentive path combination for knowledge graph completion. In *Proceedings of The 9th Asian Conference on Machine Learning, ACML 2017*, pages 590–605, 2017. 2.2.1

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

[69] 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. 3, 3.1.2

[70] 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. 3.1.3.1

[71] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014. 6

[72] Ioannis Konstas and Mirella Lapata. A global model for concept-to-text generation. *J. Artif. Intell. Res.*, 48:305–346, 2013. 5.1

[73] Ioannis Konstas, Vassilios Stathopoulos, and Joemon M. Jose. On social networks and collaborative recommendation. In *Proc. SIGIR '09*, pages 195–202, 2009. 2.2.1

[74] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proc. KDD '08*, pages 426–434, 2008. 2.3.4, 3, 3.2.3

[75] 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. 2.2.1, 2.2.2

[76] 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. 2.2.2

[77] 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`. 3.1.3.2

[78] John E. Laird and Shiwali Mohan. Learning fast and slow: Levels of learning in general autonomous intelligent agents. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 5

[79] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014. 3.2.4

[80] 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. 4.1.4.1, 5.1

[81] 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. 3.2.2

[82] 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. 4.1.4.3

[83] 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. 4.1.3

[84] 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. 3, 3.1.2

[85] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. SUGILITE: creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 6038–6049, 2017. 5

[86] 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. 4.6.5

[87] Chin-Yew Lin and Eduard H. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL 2003*, 2003. 4.4, 5, 4.7.3.6

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

[89] 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. 2.2.1

[90] 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. 3, 3.1.1

[91] 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. 4.1.4.2

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

[93] 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`. 4.1.4.2

[94] Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. 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 11–19, 2015. 5.3

[95] 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. 2.2.1

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

[97] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and

David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. 1

[98] Yoichi Matsuyama, Arjun Bhardwaj, Ran Zhao, Oscar Romeo, Sushma Akoju, and Justine Cassell. Socially-aware animated intelligent personal assistant agent. In *Proceedings of the SIGDIAL 2016 Conference, The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 224–227, 2016. 5

[99] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *RecSys '13*, pages 165–172, 2013. 2.2.2

[100] 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. 3, 3.1.1, 3.2.3, 3.3.2, 3.3.3

[101] 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. 4.1.2, 4.3

[102] 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. 3.3.1, 4.3

[103] 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. 3.3.1, 4.3

[104] 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. 4.3

[105] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, Jun 1947. ISSN 1860-0980. doi: 10.1007/BF02295996. URL `https://doi.org/10.1007/BF02295996`. 3.4, 4.7.4.3

[106] 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. 4.1.4.1

[107] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, 2013. 3.2.2, 1

[108] 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, 2.2.1

[109] Federico Monti, Michael M. Bronstein, and Xavier Bresson. Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks. In *Advances in Neural Information Processing Systems*, pages 3700–3710, 2017.

[110] 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. 4.1.2, 5.1

[111] 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*, 2015. 2.2.1

[112] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814, 2010. 3.2.2

[113] 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. 4.1.1

[114] 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`. 4.1.1

[115] Ary Fagundes Bressane Neto and Flávio Soares Corrêa da Silva. A computer architecture for intelligent agents with personality and emotions. In *Human-Computer Interaction: The Agency Perspective*, pages 263–285. 2012. 5

[116] Jianmo Ni, Zachary C. Lipton, Sharad Vikram, and Julian McAuley. Estimating reactions and recommending products with generative models of reviews. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017*, pages 783–791, 2017. 4, 4.1.3, 4.2.2, 4.3, 4.3, 4.4, 4.7.1, 5.4

[117] 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. 4.1.1

[118] 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. 3.1.2

[119] 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, 2013. 2.2.1

[120] 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. 2.2.1

[121] Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy, Elena Baralis, Michele Osella, and Enrico Ferro. Knowledge graph embeddings with node2vec for item recommendation. In *The Semantic Web: ESWC 2018 Satellite Events*, pages 117–120, 2018. 2.2.1

[122] 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. 4.4, 3, 4.7.3.4

[123] 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

[124] Michael J. Pazzani and Daniel Billsus. The adaptive web. chapter Content-based Recommendation Systems, pages 325–341. 2007. 2.2.1

[125] Florian Pecune, Jingya Chen, Yoichi Matsuyama, and Justine Cassell. Field trial analysis of socially aware robot assistant. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, pages 1241–1249, 2018. 1, 5

[126] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. 3.2.2

[127] 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. 3.1.3.2

[128] Mickaël Poussevin, Vincent Guigue, and Patrick Gallinari. Extended recommendation framework: Generating the text of a user review as a personalized summary. 2015. 4.1.2

[129] JZ Qiu, Jian Tang, Hao Ma, YX Dong, KS Wang, and J Tang. DeepInf: Modeling Influence Locality in Large Social Networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDDâĂŹ18)*, 2018.

[130] M. Atif Qureshi and Derek Greene. Lit@eve: Explainable recommendation based on wikipedia concept vectors. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017*, pages 409–413, 2017. 5.1

[131] 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. 4.6.5

[132] Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. Social collaborative viewpoint regression with explainable recommendations. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017*, pages 485–494, 2017. 5.1

[133] Steffen Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 995–1000, 2010. 3.2.1

[134] Charles Rich and Candace L. Sidner. COLLAGEN: when agents collaborate with people. In *Proceedings of the First International Conference on Autonomous Agents, AGENTS 1997*, pages 284–291, 1997. 5

[135] 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. 4.1.1

[136] 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. 3, 3.2.3, 3.3.3

[137] 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. 4.6.5

[138] 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. 3, 3.1.2

[139] 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`. 4.1.4.1

[140] 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. 2.2.2

[141] 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. 3.1.3.2, 3.2.4

[142] 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. 4.1.1

[143] Sirawit Sopchoke, Ken-ichi Fukui, and Masayuki Numao. Explainable cross-domain recommendations through relational learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 5.1

[144] 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. 3.2.1

[145] Shashank Srivastava, Igor Labutov, and Tom M. Mitchell. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, pages 1527–1536, 2017. 5

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

[147] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-To-End Memory Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448, 2015. 4.7.2

[148] 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. 2.2.1.1

[149] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. Rating-boosted latent topics: Under-

standing 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. 3.1.1

[150] 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. 2.2.1

[151] 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`. 4.1.3

[152] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Translational recommender networks. *CoRR*, abs/1707.05176, 2017. URL `http://arxiv.org/abs/1707.05176`. 3.1.2

[153] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2309–2318, 2018. 3.1.2

[154] Nava Tintarev and Judith Masthoff. *Designing and Evaluating Explanations for Recommender Systems*, pages 479–510. Springer US, Boston, MA, 2011. 1, 1, 4

[155] 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. 2.2.2

[156] Yulia Tsvetkov, Alan W. Black, Ruslan Salakhutdinov, and Shrimai Prabhumoye. Style Transfer Through Back-Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 866–876, 2018.

[157] J. Vig, S. Sen, and J. Riedl. Tagsplanations: Explaining recommendations using tags. In *IUI '09*, pages 47–56, 2009. 2.2.2

[158] 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. 3.1.2

[159] 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. 4.1.4.2

[160] 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. 4.1.4.2

[161] 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. 3.1.1, 3.1.2, 3.2.3, 3.3.3

[162] 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. 3, 3.1.2, 3.2.3, 3.3.3

[163] Lu Wang and Wang Ling. Neural network-based abstract generation for opinions and argu-

ments. 2016. 4.1.1

[164] 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. 2.2.1

[165] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. Explainable recommendation via multi-task learning in opinionated text data. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018*, pages 165–174, 2018. 5.1

[166] 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

[167] 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, 2.2.2, 2.3.2, 2.3.2

[168] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. TEM: tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018*, pages 1543–1552, 2018. 5.1

[169] Zhongqing Wang and Yue Zhang. Opinion recommendation using A neural model. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1626–1637, 2017. 4, 4.1.3, 4.7.2, 4.7.2

[170] 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. 3.1.3.2

[171] Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263, 2017. 5.1

[172] 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. 4.1.2

[173] Libing Wu, Cong Quan, Chenliang Li, Qian Wang, and Bolong Zheng. A context-aware user-item representation learning for item recommendation. *CoRR*, abs/1712.02342, 2017. 3.1.2

[174] 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. 3.1.2

[175] 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. 2.2.1

[176] Xin Xin, Chin-Yew Lin, Xiaochi Wei, and Heyan Huang. When factorization meets heterogeneous latent topics: An interpretable cross-site recommendation framework. *J. Comput. Sci. Technol.*, 30(4):917–932, 2015. 5.1

[177] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33nd International Conference on Machine*

Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, pages 2397–2406, 2016. 4.7.2

[178] 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. 2.2.1

[179] 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. 4.1.4.2

[180] 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. 4.1.4.2

[181] 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`. 4.1.1

[182] 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. 4.1.1

[183] 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. 3.2.1

[184] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, pages 974–983, 2018.

[185] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. Recommendation in heterogeneous information networks with implicit user feedback. In *Proc. RecSys*, 2013. 2.2.1

[186] 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, 2.2.1, 2.2.1.1, 2.4.1, 2.4.2, 2.4.5

[187] 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. 3.1.3.2

[188] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014. 4, 4.2.2, 4.4, 4.5

[189] 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, 2014. 2.2.2

[190] 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. 3.1.1, 4.1.2, 5.1

[191] Tiancheng Zhao and Maxine Eskénazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of the SIGDIAL 2016 Conference, The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 1–10, 2016. 5

[192] Tiancheng Zhao, Allen Lu, Kyusong Lee, and Maxine Eskénazi. Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 27–36, 2017. 5

[193] Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 654–664, 2017. 5

[194] 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. 3, 3.1.2, 3.2.1, 3.2.2, 2, 3.3.3, 1

[195] Ming Zhou, Mirella Lapata, Furu Wei, Li Dong, Shaohan Huang, and Ke Xu. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*, pages 623–632, 2017. 4.1.3