Conversational Recommender Systems using Multi-Relational Data

by

Ashwin Deshpande

PROJECT RESEARCH

Submitted in partial fulfillment of the requirements
for the degree of Master of Science, Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2022

Chicago, Illinois

Defense Committee:
Major Advisor: Dr. Natalie Parde
Committee: Dr. Cornelia Caragea

TABLE OF CONTENTS

SUMMARY

The main goal of a recommender system is to solve an individual's information overload when trying to search for personalized recommendations. This is useful in many settings—for instance, companies may find it essential to match their products with the right customers, and recommender systems can help match segments of their customer base to items they are most likely to buy. Recommendation systems have been around since the 1970s, beginning with a computer librarian named Grundy that was built at Duke University to help students select books by asking their preferences. In a similar manner, conversational recommender systems today (such as that which is the focus of this project) take user utterances consisting of preferences and output a list of recommendable items.

Traditional recommender systems, such as content-based and collaborative filtering techniques, do not take into account two problems. First, these approaches assume that user preferences remain the same over a period of time. Second, they try to recommend items even when a user does not have a recorded history of choices. Similar problems to the latter can also arise when new items are introduced to an inventory (known as the *cold start problem*). Conversational recommender systems look to solve these issues by providing an interface to specify current preferences, assuming little or no history about the user. In addition to these two problems, many current recommender systems suffer from a lack of explainability: given a user input, systems may produce a recommendation but in many cases provide no reason for why it was selected. This is a sharp contrast to the norm when recommendations are made in

real-life conversation (e.g., "I think you will like the new Spider-man movie since you said you loved the prequel").

In this research project, we show how multi-relational data helps to obtain explainable and meaningful recommendations. This project is built as a module for task-oriented dialogue systems, and more specifically those acting as conversational recommender systems. Beyond the possible inclusion of a recommender module, typical dialogue systems consist of several other sub-tasks such as slot-filling, intent classification, dialogue management, and natural language generation. Although these other sub-tasks are out of scope of the present work, we address both natural language understanding and recommendation in the context of this project.

CHAPTER 1

INTRODUCTION

Recommendation systems are highly sought-after NLP applications in the commercial domain, and are responsible for large revenue gains for e-commerce sites such as Amazon, Netflix and YouTube. Recommendation tasks can be defined in several ways: (1) retrieving a single most relevant item, (2) retrieving a set of relevant items, or (3) retrieving a ranked list of the most relevant items. These different task definitions are best approached in different ways. We focus on the third, specifically ranking the top 10, 20, 50 or 100 relevant items for a given user query. Since users may not be aware of the scope of options when specifying preferences, providing a larger list of recommendable items (e.g., 50 or 100) allows the user to explore larger set of options and change preferences. On the other hand, offering many possible selections to users may increase information overload; offering smaller sets (e.g., 10 or 20) based on heightened understanding of user preferences may help reduce that burden.

Recommendation systems also help e-commerce sites promote items and increase revenue. Real life recommendation conversations often start with several clarifying questions. For example:

- What are you in the mood for today?
- Do you feel like watching a Sci-Fi movie?

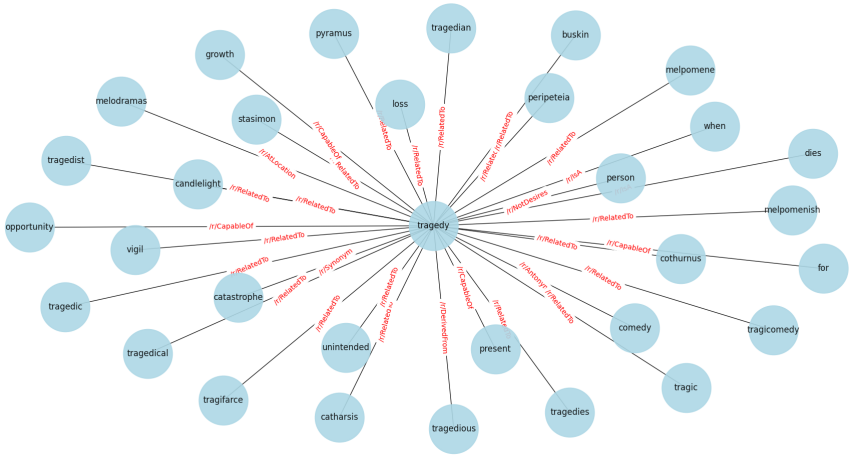


Figure 1: Word Sub-graph for the word "tragedy"

This remains true for a variety of recommender domains, including but not limited to restaurants, music, travel, news, and weather. Traditional recommendation systems such as collaborative-filtering or content-based recommendation do not allow you to explore new items, but modern conversational recommendation systems help to solve this issue by enabling a richer set of interactions that clarify and fine-tune user preferences. This project moves towards exploiting the information present in natural language interactions in text and speech to identify and harness this valuable information. Specifically, we use knowledge graphs for the purpose of extracting path relations between similar items or words (Figure 6). For example, user utterances including the word "tragedy" would be more indicative of a preference for melodramatic movies. Knowledge graphs capture information (*facts*) such as "comedy is the opposite of tragedy" in the form of triples (comedy, antonym, tragedy), allowing for the logical search and manipulation of information based on relational connections in a network.

1.1 Related Work

Some of the earliest work in Conversation Recommender Systems[(1), (2)] focus on matrix factorization methods but suffer from cold-start problems. We obtain our dataset from (3) who use an end-to-end auto-encoder technique to over-come incomplete movie-rating matrix to account for new users. We based our approach of using two distinct knowledge graph on Zhoe et al (4) and we improve the system by adding additional steps (3.1) to learn better node embeddings and design a different method to perform Mutual Information Maximization (More in 3.2).

1.2 Contributions

In this project we focus on several core tasks, including *entity extraction and matching* (§1.2.1), *text extraction and matching* (§1.2.2), *word and entity embedding* (§1.2.3), and *prediction and ranking* (§1.2.4). In combining these components into a pipeline, we develop a recommendation engine that can be used to extract words and entities from raw text input and return a list of the most relevant items. We select movies as our target domain, with associated entities consisting of actors, languages, countries, directors, film production companies, and other stakeholders in the movie industry. Our dataset, described in Chapter 2, consists of 6428 movies and TV shows. Given an input text our system provides a ranked list of recommended movies and TV shows with average 0.70 AUC score. We summarize our individual contributions for each component of our pipeline in the subsections below.

1.2.1 Entity Extraction & Matching

We perform coreference resolution before extracting entities as it reduces the overhead for entity matching. Given a user input text, we frame the extraction of named entities as a sequence tagging task. The resulting entities are spans of text that may or may not be valid nodes in an entity knowledge graph (*entity KG*). Hence, it is necessary that we match our extracted entities to nodes in the entity KG. Our entity KG is extracted from DBpedia (5). DBpedia is a community project that extracts information derived from the hyperlinks in Wikipedia. DBpedia consists of billions of facts and millions of entities, including those holding relevance in our recommendation domain. Since these entities are nodes in a large KG, we can extract one/two-hop neighbors (which may be movies) of the entity node as recommendations. For example, when we perform NER on the sentence "I recently saw a Channing Tatum movie The Vow. It turns out to be a real story!" (Figure 2), we may observe that one-hop neighbors of the extracted entity "Channing Tatum," such as "She's the Man" or "Dear John," are relevant items Figure 3.

1.2.2 Text Extraction & Matching

Similarly, words in text are cleaned in order to find the closest match in our word knowledge graph (*word KG*). Our word KG is a subset of ConceptNet (6) that consists of triples that help extend the meaning of words such as "comedy" to "comic," "jokes," "funny," and "dramedy." Matching words to their valid counterparts in the word KG in turn helps in identifying words that may be related to an entity in the entity KG. We illustrate this in Figure 4, where other

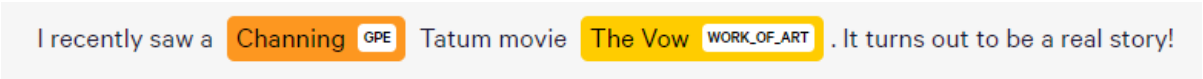


Figure 2: Named Entity Extraction on example sentence

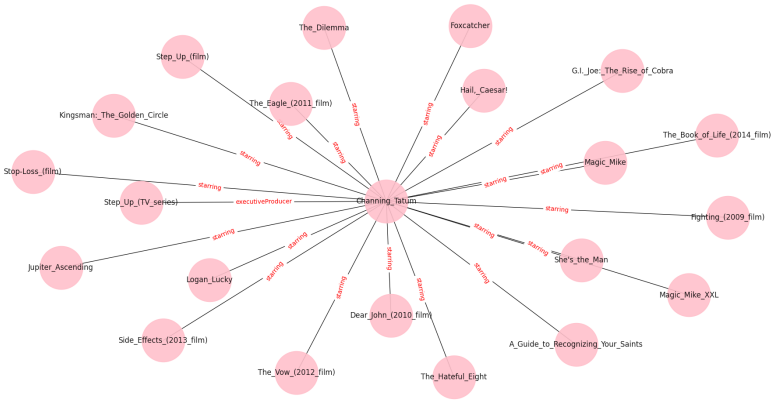


Figure 3: Entity sub-graph consisting of the entities "The Vow" and "Channing Tatum."

words related to "funny" such as "comedy" or "dramedy" are relevant genres closely related to several movie entities in the entity KG.

1.2.3 Word & Entity Embedding

Our model makes recommendations based on a similarity measure that helps us fetch the closest recommendations to extracted entities and words. We take as input a list of word embeddings and a single entity embedding, which helps us map our sentences and entities to a vector space where words such as "animation" or "funny" are closer to entities like "Toy Story" or "Lego Movie."

User Input: I am looking for something funny, something like the movie Nice Guys.

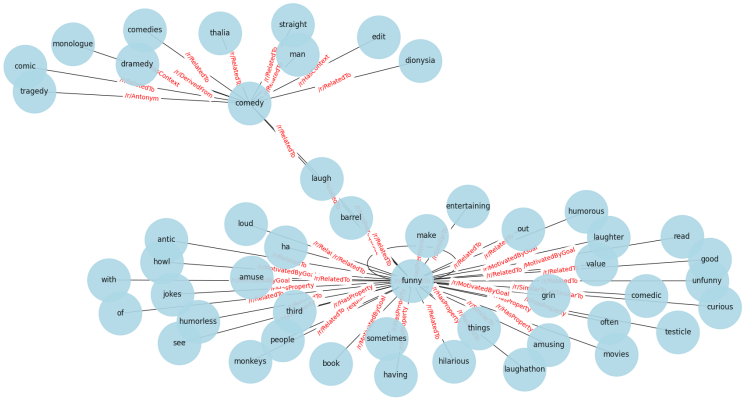


Figure 4: Word sub-graph for the word "funny."

1.2.4 Prediction & Ranking

Finally, user utterances are converted to a list of embeddings representing a sentence. The sentence embedding is then paired with all movie entities' embeddings. Pairs of inputs are mapped onto a vector space such that relevant items have a smaller distance to the sentence, while irrelevant entities have larger distances. These (sentence_embedding, movie_embedding_i) pairs are then ranked according to their distances.

1

[1] https://github.com/AshwinDeshpande96/Conversational_Recommender_System_using_Multi-Relational_Data

CHAPTER 2

DATA COLLECTION

Conversational recommender systems typically consist of two modules: (1) a *conversational module* that involves understanding user utterances, classifying user intent, and producing an appropriate natural language reply; and (2) a *recommendation engine* that is responsible for identifying content for the system's response. The conversational module thus requires a natural language dialogue dataset, and the recommendation engine requires a database of items or entities to infer relations between entities and natural language text. We further make use of information present in inter-connectivity of related nodes in knowledge graphs to infer relations between related entities and synonyms or semantically similar words.

2.1 Dialogue Dataset

For our conversational module, we leverage the ReDial dataset (7). ReDial is a task-oriented dialogue dataset centered around two human participants: A *recommendation seeker* and the *recommender* (Table I). In total, the dataset consists of 10,006 training conversations with an average of 18 messages per conversation, and 1342 test conversations with a similar average length. This data (natural language conversations) was collected using Amazon Mechanical Turk (8). On the interface, participants had the option to mention a movie by typing an "@" symbol, which helped them to obtain a list of movies gathered from DBpedia. Both participants were requested to answer several questions after the conversation (Table II) including:

PARTICIPANT	UTTERANCE
SEEKER:	@163606 It is a nice movie. What say?
SEEKER:	@90950 Was a marvelous 3D experience personally for me
RECOMMENDER:	I really liked that movie but haven't seen it in a while. Are you a Michelle Rodriquez fan?
SEEKER:	No. I am not. I like sci-fi movies. Have you watched @155969
SEEKER:	My another favorite sci-fi movies are @79320 and @84001
RECOMMENDER:	No I have not seen that one but have seen all the @204292 movies. Do you like those
SEEKER:	Yes, I love @204292 series
SEEKER:	In fact I am excited about the upcoming @204292
SEEKER:	What are you favorite movies?
RECOMMENDER:	I am too I think you will like @147598
SEEKER:	I have not seen @147598
SEEKER:	I will watch it soon
RECOMMENDER:	It is a good movie if you like Star wars you should like it.
SEEKER:	Great. I will watch those. Now, let us submit this. Nice talking to you, have a nice day!
RECOMMENDER:	Thanks it was nice talking to you too bye.

TABLE I: Example conversation in ReDial Dataset between Recommendation seeker and Recommender

MovieId	MovieName	suggested	seen	liked
79320	Contact (2009)	0	0	1
84001	Donnie Darko (2001)	0	0	1
155969	The Martian	0	0	1
90950	Avatar	0	0	1
163606	Avatar	0	0	0
147598	Indiana Jones and the Kingdom of the Crystal Skull (2008)	1	1	1
204292	Star Wars	1	1	1

TABLE II: Answers from Respondent/Recommender on whether the Seeker suggested, seen, liked a movie. From conversation in Table I

1. Was the mentioned movie suggested by the recommender or the seeker?
2. Had the seeker seen the movie?
3. Did the seeker like the movie, not like the movie, or not provide an indication either way?

These tags were used to create the supervised input for our model. Along with each utterance from the seeker, if there was a movie mention with the tag "like" we used this pair as a positive example of a sentence and corresponding entities; likewise, if the seeker did not "like" the movie, we used this sentence and entity pair as a negative example.

We performed named entity recognition to find additional entities in an utterance to create positive pairs, and we computed Mutual Information Maximization (9) as a pre-training step to learn embeddings that convey the same information given by it's location in the graph and it's neighbors. The identified entities and movies were added as seeds to allow for further expansion of the knowledge graph. A conversation may consist of several lagging utterances from the same participants, which were collapsed such that there was an alternative dialogue. A history of local utterances were then combined to facilitate larger context for movie suggestion, ensuring that, e.g., no other information already provided in a conversation involving the same participant was recommended. We found that the previous five utterances of the seeker were sufficient to increase the number of positive/negative entity and utterance pairs.

Other datasets such as Facebook Rec (10) consisting QA, Recommendation conversations, DuRecDial (11), OpenDialKG (12) and (13) were considered, however are synthetically generated dialogues and do not contain spontaneous natural text.

2.2 Item Dataset

The ReDial dataset consists of a list of 6925 movies with their MOVIEID and the year of release (e.g., *75903, A Star Is Born (1954)*). These help to uniquely identify a movie in DBpedia. With the help of the list of movies we were able to obtain a list of attributes using the SPARKQL interface provided by DBpedia.[1] To ensure that we found the correct movie, we queried (for an example, see Figure 5) for a entity of type: `https://dbpedia.org/ontology/Film` that had a label with the matching year.

```
SELECT *
WHERE {
     ?movie dbp:name "A Star Is Born"@en;
            dct:subject ?Concept ;
            dbo:budget ?budget ;
            dbo:cinematography ?cinematography ;
            dbo:director ?director ;
            dbo:distributor ?distributor ;
            dbo:writer ?writer ;
            dbp:country ?country ;
            dbp:producer ?producer ;
            dbp:language ?language ;
            dbp:music ?music .
     ?Concept rdfs:label "1954 films"@en
    }
```

Figure 5: SPARKQL query used to fetch data for each movie.

[1] `https://dbpedia.org/sparql/`

We also individually revisited each movie entity to get all possible fields. Each item had a unique set of attributes, which may or may not include all attributes mentioned in the above query. It may also have had attributes that are unique to one or several entities; this prevented us from using traditional databases where a fixed number of relations/attributes should be present for all entities. The attributes deviated further for other kinds of entities like "starring." Thus, the item dataset is modelled as a node in a larger knowledge graph.

2.3 Knowledge Graph

A knowledge graph, also known as a *semantic network*, is a structured graph consisting of nodes and edges representing objects or facts in the real world. Node can be entities in a graph (for example, "Channing Tatum" in Figure 3), while edges connect two nodes. If there are multiple types of nodes or edges they are given labels. For example, "executiveProducer" is a label assigned to the edge between "Step_Up(TV_series)" and "Channing_Tatum." This relationship is often represented as (subject, predicate, object) directed triples. We use two kinds of knowledge graphs for entities and text.

2.3.1 DBpedia

DBpedia[1] (5) is a large knowledge graph with 1.46 billion facts or edges and that describes 10 million additional things or nodes. Our dataset is a subset of DBpedia that considers only a subset of nodes and edges that are relevant to our *items* database. In comparison, our knowledge graph has 118,102 edges and 48,107 nodes.

---

[1] `https://github.com/dbpedia/`

English_language	5483
United_States	4880
United_Kingdom	692
Universal_Pictures	585
Paramount_Pictures	537
France	479
Columbia_Pictures	439
Walt_Disney_Studios_Motion_Pictures	373
Warner_Bros.	362
Soundtrack	342
Compilation_album	318
Warner_Bros._Pictures	305
Metro-Goldwyn-Mayer	293

TABLE III: Most Frequent Entities in Entity Knowledge Graph. (All entities are prefixed by https://dbpedia.org/resource/.)

We take our seed set of movies and entities parsed from the dataset as described in §2.1 and perform a one-hop expansion, starting from 6925 movies and adding movies, actors, directors, and TV shows. Once again, we then go through each of the 48,107 nodes to identify any new nodes that are of type `https://dbpedia.org/ontology/Film` or `https://dbpedia.org/ontology/TelevisionShow`. An important step after collecting the knowledge graph is to make sure that nodes are unique (the unique identifier used for each node is the resource URL for that entity). To ensure uniqueness, we resolve any entities such as "New Order, John Robie" to both relevant entities named (i.e., `http://dbpedia.org/resource/New_Order_(band)` and `http://dbpedia.org/resource/John_Robie`). Edges involving compound entities like (e1, r, "New Order, John Robie") are simplified to (e1, r, `http://dbpedia.org/resource/New_Order_(band)`) and (e1, r, `http://dbpedia.org/resource/John_Robie`).

entity1	relation	entity2
Headhunter_(2009_film)	country	Denmark
Headhunter_(2009_film)	language	Danish_language
Headhunter_(2009_film)	director	Rumle_Hammerich
Headhunter_(2009_film)	starring	Lars_Mikkelsen
Headhunter_(2009_film)	starring	Charlotte_Munck
Headhunter_(2009_film)	writer	Rumle_Hammerich
Headhunter_(2009_film)	cinematography	Dan_Laustsen
Headhunter_(2009_film)	distributor	Nordisk_Film
Angels_in_the_Outfield_(1994_film)	country	United_States
Angels_in_the_Outfield_(1994_film)	language	English_language
Angels_in_the_Outfield_(1994_film)	director	William_Dear
Angels_in_the_Outfield_(1994_film)	starring	Jay_O._Sanders
Angels_in_the_Outfield_(1994_film)	starring	Tony_Danza

TABLE IV: Subset of Entity Knowledge Graph (All entities are prefixed by https://dbpedia.org/resource/)

2.3.2 ConceptNet

ConceptNet (6) is a knowledge graph that describes relationships between words and phrases. It has over 21 million edges and 8 million nodes in 83 languages; however, for our application we only considered 54,672 English words and phrases, and 130,222 edges. The edge relations are independent of language and we use a subset of 44 relation types (the top ten most frequent relations are shown in Table V). ConceptNet has both symmetric (undirected) relations (for example, Antonym, RelatedTo, and Synonym) and asymmetric (directed) relations (for example, DerivedFrom and AtLocation). We do not make this distinction in our graph, instead considering all edges to be undirected.

/r/RelatedTo	78326
/r/DerivedFrom	11079
/r/Synonym	4761
/r/IsA	4555
/r/CapableOf	4325
/r/AtLocation	4237
/r/FormOf	3361
/r/UsedFor	3088
/r/HasContext	2688
/r/HasProperty	1940

TABLE V: Top 10 most frequent relations in Word Knowledge Graph

source	relation	target
expect	/r/RelatedTo	obligatory
perhaps	/r/Antonym	definitely
great	/r/Synonym	gigantic
arrow	/r/RelatedTo	founder
crape	/r/RelatedTo	crape
yeah	/r/RelatedTo	celebration
atsake	/r/DerivedFrom	sake
marvy	/r/RelatedTo	brilliant
few	/r/RelatedTo	sane
pronounceth	/r/DerivedFrom	pronounce

TABLE VI: Sub-graph of Word Knowledge Graph

CHAPTER 3

MODEL ARCHITECTURE

In developing this project, a key goal was to fit a model that predicts semantic relatedness between items and words. For this purpose, knowledge graphs were not directly usable, since machine learning models require inputs (nodes) to have meaning in continuous space. In order for the model to accept relational information, we embedded our nodes and relations in vector space. We refer to these as as distributional vector representations (§3.1). Since our knowledge graphs for words and entities are in independent vector spaces, a distance measure between word embeddings and entity embeddings does not carry any meaning. Our final model helps us obtain new vector embeddings for words and entities in the same continuous space, where we can use distance measures such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and cosine distance. We observe that MAE performs best for our task (§3.1).

3.1 Embeddings

We obtain our node representations in a two step process. First, we represent the knowledge graph as a sparse term-term matrix. This is a co-occurrence matrix that represents the connectivity of each node in the graph. There are several methods to obtain our initial node representations; one of the simplest is known as *TransE* (§3.1.1). Second, we optimize our previously obtained embeddings such that a given node is close to its more-than-one-hop neighbors. For word embeddings, the previously obtained word embeddings are similar to those produced

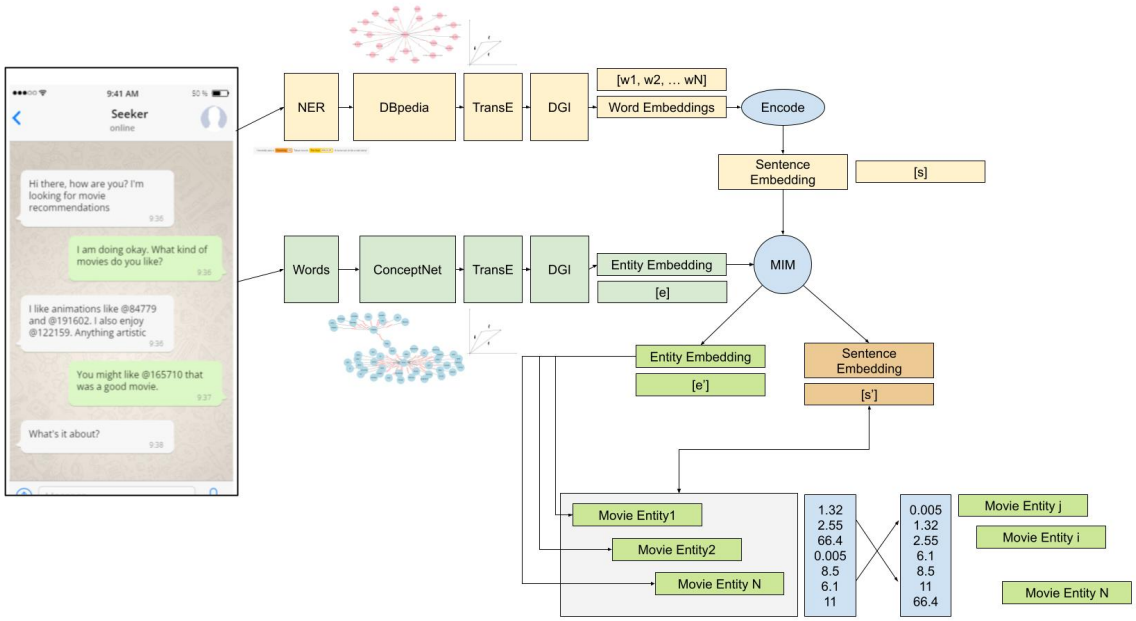


Figure 6: Model Architecture

by word2vec (14) or GloVe (15). While it is possible to directly use word2vec or GloVe embeddings, we show that a combination of semantic meaning in word2vec and relational meaning in embeddings obtained from the knowledge graph lead to best results.

3.1.1 TransE

TransE (16) is an energy-based model that translate nodes in the embedding space. Given a triple (h, l, t), TransE optimizes vectors such that $\mathbf{h}$ plus the relation vector $\mathbf{l}$ is closer to $\mathbf{t}$, i.e., $\mathbf{h} + \mathbf{l} \approx \mathbf{t}$ (Figure 7) when $\mathbf{h}$ and $\mathbf{t}$ are neighbors. It should also hold that $\mathbf{h} + \mathbf{l}$ is far away from $\mathbf{t}$ when $\mathbf{h}$ and $\mathbf{t}$ are not neighbors. For this purpose, TransE optimizes the following objective function:

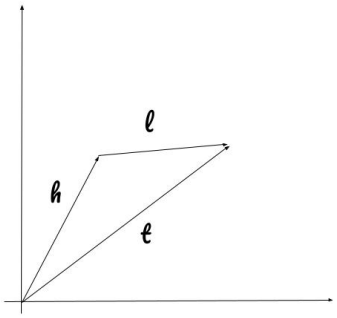


Figure 7: Vector Addition in TransE

$$\mathcal{L} = \sum_{(h,l,t)\in S} \sum_{(h',l,t')\in S'_{(h,l,t)}} [\gamma + d(\mathbf{h}+\mathbf{l},\mathbf{t}) - d(\mathbf{h'}+\mathbf{l},\mathbf{t'})]_+ \tag{3.1}$$

$$S'_{(h,l,t)} = (h',l,t)|h' \in E \cup (h,l,t')|t' \in E \tag{3.2}$$

where S is the training set of triples, γ is the cutoff margin, and $d(\mathbf{x},\mathbf{y})$ is a distance measure between two vectors.

The set of corrupted triples is generated (Equation 3.2) such that one of the head or tail is randomly sampled from the set of nodes, but not both. TransE embeddings can be used to predict (1) the relation between two nodes, (2) the tail node given the head and relation (for example, see Table VII), and (3) the head node given the relation and the tail.

We perform entity target prediction to verify if the head and relation vector addition are close to the tail vectors (for example, see Table VIII). We note that there could be a number of actors, editors, or writers on a movie. This is demonstrated in some of the predictions

source	relation	target	predicted_target
wendigo	/r/RelatedTo	member	wendigo
decoupage	/r/RelatedTo	glued	glued
menthol	/r/DerivedFrom	ol	ol
reluctant	/r/DerivedFrom	reluct	reluct
person	/r/NotDesires	helpless	healthy
day	/r/HasSubevent	smile	day
al	/r/FormOf	all	alabama
appreciate	/r/RelatedTo	grateful	appreciate
pancreatoduodenectomy	/r/RelatedTo	stomach	stomach
euchologion	/r/RelatedTo	ritual	ritual
sf	/r/EtymologicallyRelatedTo	sixty	sf
iriscope	/r/RelatedTo	philosophical	philosophical
asprin	/r/AtLocation	cabinet	asprin
tail	/r/RelatedTo	rod	humans
boozing	/r/RelatedTo	heavily	boozing
not	/r/HasProperty	alike	difficult
space	/r/UsedFor	seperation	space
creed	/r/RelatedTo	credible	creed
cranky	/r/DerivedFrom	crank	crank
problem	/r/CapableOf	work	suffering

TABLE VII: TransE target prediction on word knowledge graph

in Table VIII. We see that given a movie *Beauty and the Beast (1991 film)*, with the relation *starring* and expected actor *Rex Everhart*, TransE predicts *Rex Everhart*'s co-star *David Ogden Stiers*. *David Ogden Stiers* might be the closest vector to *Beauty and the Beast (1991 film)* + *starring*, but *Rex Everhart* is in the top five closest vectors.

3.1.2 Deep Graph Infomax

Since in TransE we only consider neighboring information, it is necessary to encode node embeddings using *Deep Graph Infomax* to capture higher level features involving more than

source	relation	target	predicted_target
Predator_(film)	editing	Mark_Helfrich	Harvey Rosenstock
Patema_Inverted	genre	Fantasy	Fantasy
Kevin_Hart:_Let_Me_Explain	language	English_language	English_language
Beauty_and_the_Beast_(1991_film)	starring	Rex_Everhart	David_Ogden_Stiers
National_Treasure_(Chinese_TV_series)	starring	Cai_Guoqing	Wang_Gang_(actor)
Tokyo_Gore_Police	writer	Sayako Nakoshi	Sayako Nakoshi
X-Men_(film)	language	English_language	English_language
Mission:_Impossible_–_Ghost_Protocol	starring	Simon_Pegg	Maggie_Q
V/H/S	writer	Tyler_Gillett	Chad_Villella

TABLE VIII: TransE target prediction on entity knowledge graph.

one-hop neighbors. To encode nodes in the knowledge graph usign Deep Graph Infomax (17), we need to input an initial embedding for nodes. These initial embeddings are supplied by TransE 3.1.1. Deep Graph Infomax is an unsupervised learning method based on graph convolutions. Since graph convolutions (Equation 3.3) apply repeated aggregations, they help to capture information over a *patch* of nodes around the node_i rather than just its immediate neighbors:

$$\boldsymbol{V}^{(l)} = \mathsf{ReLU}(\mathbf{D}^{-\frac{1}{2}}\boldsymbol{A}\mathbf{D}^{-\frac{1}{2}}\boldsymbol{V}^{(l-1)}\boldsymbol{W}^{(l)}) \tag{3.3}$$

where $\boldsymbol{V}$ is the node vector at layer l, $\boldsymbol{A}$ is the adjacency matrix of the graph ($\boldsymbol{A}_{i,j} = 1$ if there is an edge connecting node_i and node_j, or 0 otherwise), $\mathbf{D}$ is the diagonal degree matrix $\mathbf{D}[i,i] = \sum_j \boldsymbol{A}[i,j]$, and $\boldsymbol{W}$ contains the learned weights at layer l.

Other works such as RDF2Vec (18) make use of random walks given a node to produce sentences (for example, “X-Men (film) is an English language film”) which are then passed

through word2vec to produce node embeddings. This method is sensitive to initial conditions and does not scale to larger graphs.

Deep Graph Infomax takes as input $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ where N is the number of nodes and each node vector $\mathbf{x} \in \mathbb{R}^F$ is a feature vector obtained from TransE. The objective is to learn an encoder $\xi : \mathbb{R}^{N\times F} \times \mathbb{R}^{N\times N} \longrightarrow \mathbb{R}^{N\times F'}$ using the objective function (Equation 3.4) based on (19):

$$\mathcal{L} = \frac{1}{N+M}\left(\sum_{i=1}^{N} \mathbb{E}_{(X,A)}\left[\log \mathcal{D}(\mathbf{h}_i, \mathbf{s})\right] + \sum_{i=1}^{M} \mathbb{E}_{(\tilde{X},\tilde{A})}\left[\log \mathcal{D}(\tilde{\mathbf{h}}_i, \mathbf{s})\right]\right) \tag{3.4}$$

where N is the number of positive examples of nodes, $\tilde{\mathbf{X}} \in \mathbb{R}^{M\times F}$ are negative examples generated by a corrupted generator similar to that in (Equation 3.2), $\mathbb{D}(\mathbf{x}, \mathbf{y})$ is a distance function, and $\mathbf{s}$ is a summary vector that captures global information $\mathcal{R} : \mathbb{R}^{N\times F} \longrightarrow \mathbb{R}^F$. This ultimately results in node embeddings for entities $\mathbf{X}^{(e)} \in \mathbb{R}^{N^{(e)}\times F}$ and words $\mathbf{X}^{(w)} \in \mathbb{R}^{N^{(w)}\times F}$.

3.2 Mutual Information Maximization

Our final module in this architecture maps $\mathbf{X}^{(w)} \in \mathbb{R}^{N^{(w)}\times F}$ and $\mathbf{X}^{(e)} \in \mathbb{R}^{N^{(e)}\times F}$ to a single vector space $\mathbf{X} \in \mathbb{R}^{N^{(e+w)}\times F}$. We simultaneously merge two vector spaces for words and entities as well as perform mutual information maximization between the sentence and entity pairs described in 2.1 (for example, see Table IX).

Seeker Utterance	Entity
i am in the mood to watch a romantic comedy What do you suggest	Romantic_comedy
i am in the mood to watch a romantic comedy What do you suggest, @115908 Have you seen that one, Oh i have seen that one I really like Drew Barrymore	Drew_Barrymore
Hello, Hello What movie would you suggest to watch, @184418 Have you seen that movie before, I love that scary movies I love @125431	Annabelle_(film)
Hey What kind of movies do you like to watch, i am really big on indie romance and dramas Ok what has / what is your favorite movie Staying with that genre have you seen or @88487 or @104253 Those are two really good ones When I was a kid I liked horror like @181097	Misery_(film)
HI there Hi what kind of movies doe you like to watch I like mostly anything Especially muscials and comedy i am not fond of movies like @125954, I love comedy movies have you seen @93013 it is very funny, I have not seen that yet Is that with the Rock quot, No it has Will Ferrel and Mark Wahlberg	Mark_Wahlberg

TABLE IX: Training data pairs of sentences with history and corresponding entity

MovieID	Movie Name	suggested	seen	liked
125431	Annabelle (2014)	0	1	1
184418	Get Out (2017)	1	1	1
181097	Misery (1990)	1	0	1
104253	The Perks of Being a Wallflower	1	0	1
88487	Juno (2007)	1	0	1
125954	Star Trek: Of Gods and Men	0	1	0

TABLE X: Movie Mentioned in Table IX along with response to questions.

Given a sentence $\mathbf{s} \in \mathbb{R}^{N_{train} \times K \times F}$ of length K, with an embedding length F we define a sentence encoder that produces a vector $\hat{\mathbf{s}} \in \mathbb{R}^{N_{train} \times F}$ and an entity embedding $\boldsymbol{e} \in \mathbb{R}^{N_{train} \times F}$. Our *MIM Loss* (Equation 3.5) is then defined as:

$$MIM(\hat{\mathbf{s}}, \boldsymbol{e}^{+}, \boldsymbol{e}^{-}) = \frac{1}{N_{train}} \sum_{i=0}^{N_{train}} \left[g(\hat{\mathbf{s}}.\boldsymbol{W}_s, \boldsymbol{e}^{+}.\boldsymbol{W}_e) - g(\hat{\mathbf{s}}.\boldsymbol{W}_s, \boldsymbol{e}^{-}.\boldsymbol{W}_e) \right] \tag{3.5}$$

where $\boldsymbol{W}_s, \boldsymbol{W}_e \in \mathbb{R}^{F \times F'}$ are learnable parameters that produce $\mathbf{s}' \in \mathbb{R}^{F'}$ and $\boldsymbol{e}' \in \mathbb{R}^{F'}$ such that $\mathbf{s}'$ and $\boldsymbol{e}'$ are close together for positive pairs and farther apart for negative pairs.

CHAPTER 4

RESULTS

To evaluate the performance of our architecture, we conduct a series of experiments using 25,340 pairs of sentences and entities. We obtain the ranked list of movies by performing a breadth-first search using the seed entities in Table IX. Note that in Table IX we show a single entity for each utterance; however, utterances in this dataset are typically long, and each utterance usually contains more than one entity. Movies ranked from 101 to 6925 are in random order. For every positive pair of a sentence and an entity, we generate one incorrect (negative) pair by randomly sampling an entity from the rank 101 to 6925. Thus, our input is (sentence embedding, positive entity embedding, negative entity) of shape: ($\mathbb{R}^{N_{batch} \times 70 \times 128}$, $\mathbb{R}^{N_{batch} \times 128}$, $\mathbb{R}^{N_{batch} \times 128}$).

In Figure 8, we show the ROC Curve for five utterance in the test set. Our goal is to obtain the maximum number of relevant movies at a given rank $n \in [10, 15, 20, 50, 100]$.

After we have obtained the embeddings for sentences and movie entities, we obtain the distances between sentences and entities and rank the movies. We assess our model's performance at this task using the following metrics:

- **AUC**: Area under the ROC curve, for which the target variable has the top 50 movies marked as true, while the rest are false. The top 50 closest movie entities are hence marked as positive and the rest are false.

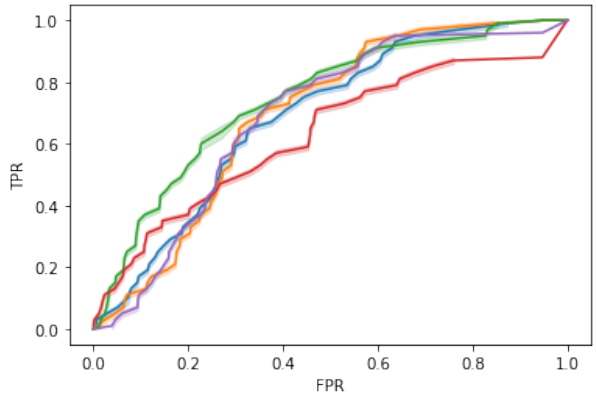


Figure 8: Test-ROC curve for 5 utterances

- **MRR**: Mean Reciprocal Rank, defined as the inverse of the rank of the most relevant movie in a true ranked list (i.e., if movie_i is ranked 1 in a true ranked list, and located at rank_i in the predicted ranked list, the MRR score is defined as:

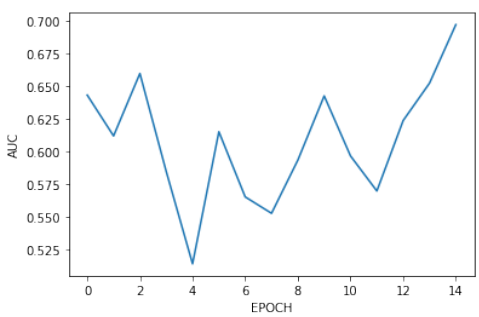
$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{rank_i} \tag{4.1}$$

  where Q is the number of queries, or in our case the number of utterances).

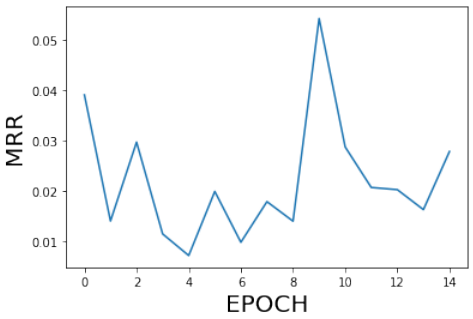
- **Recall@N**: The number of movies that are present in both the true and predicted ranked lists, divided by N:

$$Recall@N = \frac{|y \cap \hat{y}|}{N} \tag{4.2}$$

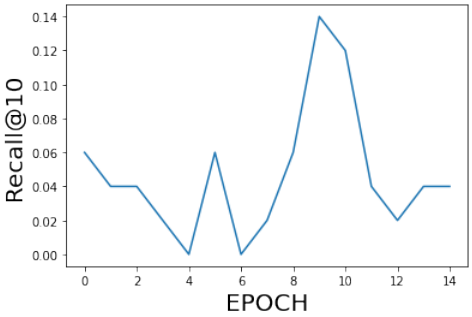
In Table XI we show the AUC, MRR, Recall@10, Recall@15, Recall@20, Recall@50, and Recall@100 for our model.



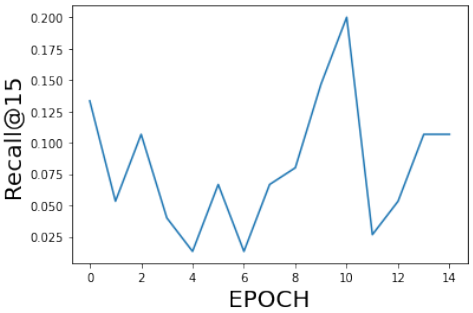
AUC Score



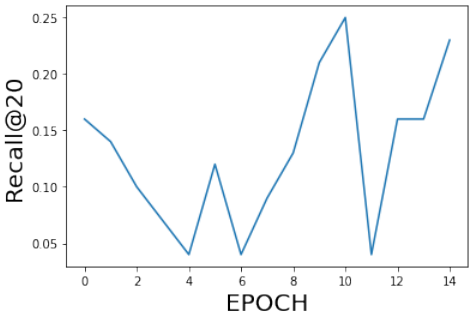
MRR



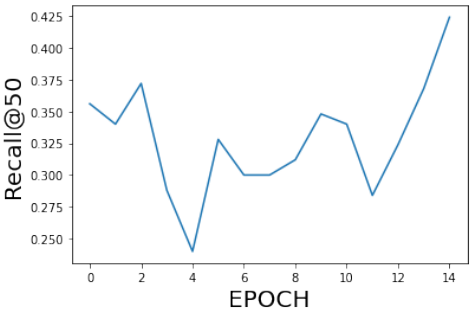
Recall@10



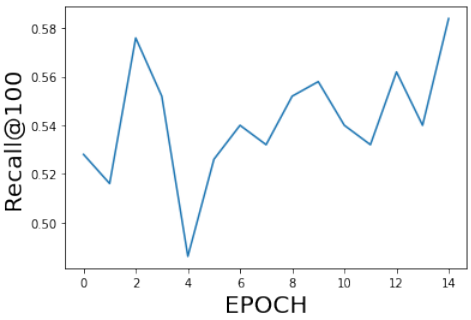
Recall@15



Recall@20



Recall@50



Recall@100

Figure 9: Validation Metrics during training

Score	min	median	max	mean	std
AUC	0.491	0.609	0.697	0.606	0.046
MRR	0.007	0.019	0.118	0.027	0.024
R@10	0.0	0.04	0.14	0.048	0.035
R@15	0.013	0.087	0.213	0.085	0.046
R@20	0.03	0.11	0.25	0.122	0.053
R@50	0.24	0.326	0.424	0.326	0.043
R@100	0.486	0.533	0.584	0.536	0.025

TABLE XI: Scores for validation set over 37 EPOCHS

4.0.1 Testing AUC Score

In Table XII we sample N utterances from the test set and rank them according to their distances.

Score	min	median	max	mean	std
AUC@5	0.577	0.6	0.606	0.594	0.013
AUC@10	0.518	0.608	0.625	0.584	0.047
AUC@15	0.569	0.594	0.606	0.59	0.015
AUC@20	0.585	0.603	0.606	0.598	0.01

TABLE XII: AUC Scores captured at 5, 10, 15 & 20 utterances

Rank	True	Predicted
1	Hot_Fuzz	The_Duchess_(film)
2	Big_Talk_Productions	Bourne_(franchise)
3	Nanny_McPhee_and_the_Big_Bang	Jersey_Boys_(film)
4	Dead_Man_Walking_(film)	Green_Zone_(film)
5	Bridget_Jones_(film_series)	War_Horse_(film)
6	About_Time_(2013_film)	Chicken_Run
7	Romeo_Is_Bleeding	My_Cousin_Rachel_(2017_film)
8	Definitely,_Maybe	Rush_(2013_film)
9	Johnny_English	Once_Upon_a_Time_in_Mexico
10	Baby_Driver	The_Life_of_David_Gale
11	Robin_Hood_(1991_British_film)	The_Heart_of_Me
12	The_World's_End_(film)	Shaun_the_Sheep_Movie
13	Les_Misérables_(2012_film)	Ladybird,_Ladybird_(film)
14	Bridget_Jones's_Baby	The_Borrowers_(1997_film)
15	The_Snowman_(2017_film)	Night_of_the_Living_Dead_(film_series)
16	Wimbledon_(film)	Hanna_(film)
17	Hail,_Caesar!	Gravity_(2013_film)
18	Bridget_Jones:_The_Edge_of_Reason	Sleepless_in_Seattle
19	Johnny_English_Reborn	Johnny_English
20	Captain_Corelli's_Mandolin_(film)	Becoming_Jane
21	A_Serious_Man	The_Jewel_of_the_Nile
22	Mr._Bean's_Holiday	Nanny_McPhee_and_the_Big_Bang
23	Nanny_McPhee	Terminator_3:_Rise_of_the_Machines
24	French_Kiss_(1995_film)	Unbreakable_(film)
25	The_Man_Who_Wasn't_There_(2001_film)	Harry_Potter_and_the_Philosopher's_Stone_(film)
26	The_Soloist	The_Adventures_of_Huck_Finn_(1993_film)
27	High_Fidelity_(film)	The_Man_Who_Knew_Too_Little
28	Anna_Karenina_(2012_film)	Notting_Hill_(film)
29	Ali_G_Indahouse	Kingsman:_The_Secret_Service
30	United_93_(film)	Kingdom_of_Heaven_(film)
31	Everest_(2015_film)	Syriana
32	Notting_Hill_(film)	Allied_(film)
33	Fargo_(1996_film)	Wimbledon_(film)
34	Tinker_Tailor_Soldier_Spy_(film)	The_Woman_in_Black_(2012_film)
35	Atonement_(2007_film)	The_Thing_(2011_film)
36	Barton_Fink	Lonesome_Dove_(miniseries)
37	The_Theory_of_Everything_(2014_film)	Kingsman:_The_Golden_Circle
38	Thirteen_(2003_film)	Solo:_A_Star_Wars_Story
39	Four_Weddings_and_a_Funeral	Schindler's_List
40	Green_Zone_(film)	Minority_Report_(film)
41	The_Big_Lebowski	Saving_Private_Ryan
42	Paul_(film)	Jane_Eyre_(2011_film)
43	The_Danish_Girl_(film)	The_Madness_of_King_George
44	Senna_(film)	So_I_Married_an_Axe_Murderer
45	Wild_Child_(film)	Sense_and_Sensibility_(film)
46	Frost/Nixon_(film)	Seven_Pounds
47	Legend_(2015_film)	Home_Alone
48	Drop_Dead_Fred	Annihilation_(film)

TABLE XIII: True and Predicted ranked list of movies(All entities can be visited by prefixing http://dbpedia.org/resource/)

CHAPTER 5

CONCLUSION

We can see that it is beneficial to explore multi-relational data as it helps to produce meaningful recommendations. One of the possible methods of citing the reasoning for recommendation is finding intermediate nodes on the path between nodes identified in user utterance and recommended movie nodes. Since we use the distance function and neighboring entities have closely related embeddings the lower ranked movies also carry meaning. One of the strengths of using knowledge graph as a database is that we can easily scale item dataset by simply adding new nodes in the knowledge graph.

This project builds a basis for exploring more natural human interaction with the user. Speech features such as pitch, volume, pauses & hesitation, emphatic stress on syllables that inform stronger inclination, hedging - phrases that soften the effect of what one is saying. Such spontaneous speech features are important in inferring implicit feedback from user. In future work we shall explore incorporating additional features to obtain better results, and design experiments to assess the performance of fully automated recommender system which do not require annotated or transcribed data.

ACKNOWLEDGMENT

I would like to express my gratitude to my Major Advisor Prof. Natalie Parde for your guidance and direction. This project would not have been possible without your continued support.

I wish to acknowledge the help provided by Prof. Cornelia Caragea for introducing fundamental concepts that motivated me to pursue this project.

PES

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

FIGURE **PAGE**

CITED LITERATURE

1. Koren, Y.: Factorization meets the neighborhood: A multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, page 426–434, New York, NY, USA, 2008. Association for Computing Machinery.

2. Christakopoulou, K., Radlinski, F., and Hofmann, K.: Towards conversational recommender systems. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, page 815–824, New York, NY, USA, 2016. Association for Computing Machinery.

3. Li, R., Kahou, S., Schulz, H., Michalski, V., Charlin, L., and Pal, C.: Towards deep conversational recommendations, 2018.

4. Zhou, K., Zhao, W. X., Bian, S., Zhou, Y., Wen, J.-R., and Yu, J.: Improving conversational recommender systems via knowledge graph based semantic fusion, 2020.

5. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C.: Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web, 6:167–195, 2015.

6. Speer, R., Chin, J., and Havasi, C.: Conceptnet 5.5: An open multilingual graph of general knowledge. 2016.

7. Li, R., Kahou, S. E., Schulz, H., Michalski, V., Charlin, L., and Pal, C.: Towards deep conversational recommendations. In Advances in Neural Information Processing Systems 31 (NIPS 2018), 2018.

8. Crowston, K.: Amazon mechanical turk: A research tool for organizations and information systems scholars. In Shaping the Future of ICT Research. Methods and Approaches, eds. A. Bhattacherjee and B. Fitzgerald, pages 210–221, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

9. Tschannen, M., Djolonga, J., Rubenstein, P. K., Gelly, S., and Lucic, M.: On mutual information maximization for representation learning, 2019.

10. Dodge, J., Gane, A., Zhang, X., Bordes, A., Chopra, S., Miller, A., Szlam, A., and Weston, J.: Evaluating prerequisite qualities for learning end-to-end dialog systems, 2015.

11. Liu, Z., Wang, H., Niu, Z.-Y., Wu, H., Che, W., and Liu, T.: Towards conversational recommendation over multi-type dialogs, 2020.

12. Moon, S., Shah, P., Kumar, A., and Subba, R.: OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 845–854, Florence, Italy, July 2019. Association for Computational Linguistics.

13. Kang, D., Balakrishnan, A., Shah, P., Crook, P., Boureau, Y.-L., and Weston, J.: Recommendation as a communication game: Self-supervised bot-play for goal-oriented dialogue, 2019.

14. Mikolov, T., Chen, K., Corrado, G., and Dean, J.: Efficient estimation of word representations in vector space, 2013.

15. Pennington, J., Socher, R., and Manning, C.: GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

16. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In Advances in Neural Information Processing Systems, eds. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, volume 26. Curran Associates, Inc., 2013.

17. Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D.: Deep graph infomax, 2018.

18. Ristoski, P. and Paulheim, H.: Rdf2vec: Rdf graph embeddings for data mining. In SEMWEB, 2016.

19. Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y.: Learning deep representations by mutual information estimation and maximization, 2018.

VITA

NAME Ashwin Deshpande

EDUCATION B.V. Bhoomaraddi College of Engineering Technology, 2018

PUBLICATIONS Shahla Farzana, Ashwin Deshpande, and Natalie Parde. "How You Say It Matters: Measuring the Impact of Verbal Disfluency Tags on Automated Dementia Detection" In Proceedings of the BioNLP (ACL, 2022).