Improving Dialog Systems using Knowledge Graph Embeddings

by

Brian Carignan, B.C.S.

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Ottawa-Carleton Institute for Computer Science
Department of Computer Science
Carleton University
Ottawa, Ontario
December, 2017

©Copyright
Brian Carignan, 2017

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Improving Dialog Systems using Knowledge Graph Embeddings

submitted by **Brian Carignan, B.C.S.**

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Professor Anthony White, Thesis Supervisor

Professor Mengchi Liu, Chair,
Department of Computer Science

Ottawa-Carleton Institute for Computer Science
Department of Computer Science
Carleton University
December, 2017

Abstract

Dialog systems are systems or applications intended to converse with a human user. Recent dialog systems have employed the sequence-to-sequence framework to treat conversation as a translation problem, translating from question to answer in an open-domain. Knowledge graph embedding started as a way to scale question answering to a large, open-domain dataset without the use of hand-crafted rules. This thesis seeks to connect the two, by converting knowledge graph embeddings to word embeddings and evaluating the resulting dialog models.

To accomplish the above, an adaptable preprocessing pipeline was developed for the Freebase knowledge graph, which was embedded using TransE and TransH. A baseline method was proposed to convert the resulting embedding to word embeddings, to be compared with GloVe and Word2Vec. The four embedding sets were trained with a sequence-to-sequence model from OpenNMT on a dialog dataset from OpenSubtitles. Each model was trained twice, with embeddings variable and fixed. A beam search post-processing method is proposed to generate more diverse answers. A probable response evaluation method is proposed as a way to compare the top answers from a set of dialog systems.

The resulting knowledge graph embeddings were found comparable to the baseline word embedding methods, suggesting they could be used as an alternative for training future dialog systems.

Acknowledgments

To my family, thank you for supporting me and helping me get to where I am today.
To Professor Anthony White, the feedback and discussions were invaluable from the beginning, thank you.

Table of Contents

Abstract	iii
Acknowledgments	iv
Table of Contents	v
List of Tables	viii
List of Figures	x
Nomenclature	xii
1 Introduction	1
1.1 Goals	2
1.2 Problem Statement	2
1.3 Motivation	2
1.4 Contributions	3
1.5 Scope	4
1.6 Document Structure	4
2 Background	5
2.1 Historical Overview	5
2.1.1 Alan Turing	5
2.1.2 Early Chatbots	6
2.1.3 Loebner Prize	6
2.2 Knowledge Graphs	7
2.2.1 Embedding	7
2.3 Word Embeddings	7

2.3.1	Skip-gram	7
2.3.2	GloVe	9
2.4	Neural Networks	9
2.4.1	Perceptron	9
2.4.2	Feed-Forward Neural Networks	10
2.4.3	Back-propagation	10
2.4.4	Recurrent Neural Networks	11
2.4.5	Long Short Term Memory	13
2.5	Neural Machine Translation	14
2.5.1	Encoder-Decoder	14
2.5.2	Beam Search	15
2.6	Evaluation	16
2.6.1	Perplexity	16
3	Related Work	17
3.1	Word Embeddings	17
3.2	Knowledge Graphs	20
3.2.1	Knowledge Graph Embedding	21
3.2.2	Entity Recognition and Disambiguation	24
3.2.3	Slot Filling	24
3.3	Dialog Systems	24
3.3.1	Improving Response Generation	26
3.3.2	Dialog Models	28
3.3.3	Dialog Corpora	32
3.4	Summary	34
4	Implementation	37
4.1	Knowledge Graph Subsystem	39
4.1.1	Raw Dataset	41
4.1.2	Trim Filter	42
4.1.3	Coarse Filter	44
4.1.4	Fine Filter	45
4.1.5	Threshold Filter	45
4.2	Embedding Subsystem	47
4.2.1	Translation Embedding	48

4.2.2	Embedding Conversion	48
4.3	Dialog Subsystem	51
4.3.1	Dialog Corpus	51
4.3.2	Tokenization	52
4.3.3	Preprocessing	53
4.3.4	Embedding	53
4.3.5	Training	53
4.3.6	Response Generation	54
5	Evaluation	56
5.1	Visualizing the Entity Vectors	57
5.1.1	Distribution Analysis	62
5.2	Dialog System Evaluation	67
5.2.1	Results	67
6	Conclusion	69
6.1	Summary of Results	69
6.1.1	Knowledge Graph Pipeline	69
6.1.2	Embedding Conversion	69
6.1.3	Beam Search Postprocessing	70
6.1.4	Plausible Response Evaluation	70
6.2	Future Work	70
6.2.1	Embedding Conversion	71
6.2.2	Beam Search Postprocessing	71
6.2.3	Plausible Response Evaluation	72
	List of References	79
	Appendix A Additional Material	80
A.1	Experiment Environment	80
A.2	Plausible Response Model	80

List of Tables

3.1	Timeline of publication for knowledge graphs, word embeddings, and translation embeddings surveyed in Section 3.2.	22
3.2	Evaluation metrics used for the surveyed NMT and generative dialog systems.	31
3.3	Dialog corpora used for the surveyed NMT and generative dialog systems. Starred (*) quantities are estimated based on the average number of words and sentences in related data.	33
4.1	Results from preprocessing the Freebase knowledge graph.	41
4.2	Sample triples from the raw Freebase dataset. Columns are separated by a tab character.	42
4.3	The same triples as Table 4.2 after running the Trim filter. All triples are significantly shorter and the last column was removed entirely. . .	43
4.4	Sample triples after running the Coarse filter.	45
4.5	The same triples as Table 4.4 after running the Fine filter. 3 triples have been deleted and 1 modified.	46
4.6	Comparing two tokenization methods applied to the dialog corpus: aggressive (A) vs. conservative (C).	52
5.1	Top 10 nearest neighbors of the word “man” for each model.	63
5.2	Pairwise comparisons for each model using Nemenyi multiple comparison test.	68
A.1	Experiment environment hardware.	80
A.2	Plausible responses and their descriptions. 1/2	81
A.3	Plausible responses and their descriptions. 2/2	82
A.4	Number of plausible responses present in the top 10 answers generated by each model. 1/4	83
A.5	Number of plausible responses present in the top 10 answers generated by each model. 2/4	84

A.6	Number of plausible responses present in the top 10 answers generated by each model. 3/4	85
A.7	Number of plausible responses present in the top 10 answers generated by each model. 4/4	86
A.8	Survey results for validating the plausible response descriptions. 1/2 .	87
A.9	Survey results for validating the plausible response descriptions. 2/2 .	88

List of Figures

2.1	Simplified illustration of the Skip-Gram and TransE embedding models.	9
2.2	A perceptron (left) and a simple feed-forward neural network (right).	11
2.3	A recurrent neural network (left) and a visualization of the unfolding step (right).	13
3.1	Overview of the related work chapter in relation to the main topic.	18
3.2	Evolution of dialog models covered in Sections 3.3 and 3.3.1.	25
4.1	High level overview of the implemented components, detailed in Sections 4.1, 4.2, and 4.3 respectively.	38
4.2	Overview of the knowledge graph subsystem pipeline.	39
4.3	Entity frequency relationship in Freebase; entities outside the thresholds (red lines) are removed.	47
4.4	Overview of the embedding subsystem. The entire process is repeated for each translation embedding method evaluated.	48
4.5	Overview of the dialog subsystem. The Training-Response Generation loop is repeated for each embedding method evaluated.	50
5.1	t-SNE projection of embeddings for the top 4k most common English words.	58
5.2	t-SNE projection of embeddings for the top 4k most common English words.	59
5.3	Highlight of the word “man” and its 20 closest neighbors in the original space.	60
5.4	Highlight of the word “man” and its 20 closest neighbors in the original space.	61
5.5	Number of common pairs between the top 20 nearest neighbor lists for each word before and after the embeddings are adjusted by the dialog system, with frequency in log scale.	64

5.6	Number of common pairs between the top k nearest neighbor lists for each word before and after the embeddings are adjusted by the dialog system.	65
5.7	Number of common pairs between the top k nearest neighbor lists for each word before and after the embeddings are adjusted by the dialog system.	66
5.8	Validation perplexity for each model over 13 training epochs.	67
5.9	Distributions for the number of plausible responses per question for each model.	68

Nomenclature

Symbol	Description
AWI	Attention with Intention
BLEU	Bilingual Evaluation Understudy
CKE	Collaborative Knowledge Base Embedding
FFNN	Feed-Forward Neural Network
LSTM	Long Short-Term Memory
METEOR	Metric for Evaluation of Translation with Explicit Ordering
NLP	Natural Language Processing
NMT	Neural Machine Translation
NN	Neural Network
RDF	Resource Description Framework
REM	Relation Encoding Model
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SGNS	Skip-Gram with Negative Sampling
SMT	Statistical Machine Translation
WMT	Workshop on Statistical Machine Translation

Chapter 1

Introduction

Dialog systems are systems or applications intended to converse with a human user. The domain of conversation can be restricted to specific domains, such as tech support, or to any conversation topic (open-domain). Early dialog systems were script-based and usually involved multiple components (or modules) such as: the Language Interpreter, State Tracker, Response Generator, and Natural Language Generator. Later dialog systems built using neural networks are said to be end-to-end when they are used in place of these components [1].

The first example of a generative dialog systems was simple adaptation of seq-to-seq [2], also used in neural machine translation, which captured the relationship between question and response rather than between two languages. This model has been extended many times in order to improve the quality of responses it generates (see Section 3.3.2). One of the areas that have shown consistent improvements to response generation is word embedding, a dimensionality reduction technique commonly used to reduce the size of the input layer for word-level language models. These replaced the previous method of representing words with arbitrary identifiers and the denser representation of words improved performance of a neural model on a wide variety of tasks. Some of the more widely used word embedding methods include Word2Vec [3] and GloVe [4]. These methods use large unstructured corpora, such as news datasets, to create embeddings for words based on their appearance alongside other words.

Knowledge graphs, such as Freebase [5] are a collection of triples in the form “head, relation, tail” and describe the world using these triples as atomic facts. Knowledge graphs leveraged in order to extract common sense knowledge, also known as knowledge graph completion. One way of doing this is by embedding the graph in

a vector space with the use of a translation embedding algorithm such as TransE [6] or TransH [7]. Knowledge graphs have also been used in conjunction with word embeddings in recommender systems, allowing a neural model to benefit from multiple different sources of information.

1.1 Goals

To use human-made facts and real-world knowledge to improve the quality of responses generated by a neural dialog model and to evaluate these models based on their best answers to the same questions.

1.2 Problem Statement

Can a dialog system trained with knowledge graph embeddings generate significantly better or worse answers than a dialog system trained with traditional word embeddings?

1.3 Motivation

In order to achieve the goals outlined in Section 1.1 and the Problem Statement provided in Section 1.2, the following questions were addressed, with the section or chapter in which they are addressed detailed in the paragraphs below.

- How can knowledge graphs be used to improve dialog systems?
- How can the knowledge graph be preprocessed in order to conserve information and be within hardware constraints?
- How can knowledge graph (entity) embeddings be converted to word embeddings?
- How can the dialog model be tuned to generate sufficiently different responses required for evaluation?
- How can each model be evaluated in a way which minimizes bias?

Although there are ways to utilize a knowledge graph with dialog systems explicitly, such as constructing user personalized knowledge graphs [8], the decision to use knowledge graph embeddings was made due to two primary reasons: their success in extracting common sense reasoning (see Section 3.2.1), and the widespread use of word embeddings in dialog systems (see Section 3.1).

Freebase is a very large dataset which is 425GB uncompressed and consists of roughly 3.1B RDF triples and 86M entities. In order to fit within hardware constraints when embedding the knowledge graph, the number of entities needed to be reduced to below 10M. To minimize loss of information, a preprocessing pipeline was created which filters out entities which are less likely to contribute useful information to the embeddings (see Section 4.1).

Entities from a knowledge graph are identified with a unique key. Additionally, entities may have an associated non-unique English (or other language) label consisting of one or more words. A baseline method is proposed in order to convert these to word embeddings consisting of a single word (see Section 4.2.2).

One of the biggest problems with generative dialog systems is their tendency to generate vague responses for the majority of inputs, which makes it difficult to evaluate them. To determine if responses from a dialog model are significantly different than another model's, the responses should be sufficiently different when asking the same question to multiple models. The responses should also be different across questions if you assume that the questions you ask warrant different responses. In order to generate these responses, the beam search algorithm was modified and applied uniformly to each model (see Section 4.3.6).

A plausible response evaluation model is proposed in order to determine if a model is significantly different from another. The evaluation model works by selecting the number of plausible responses from a list using a clear definition of what makes a plausible response. This criteria is applied to all models to create a table of scores, which can be tested for significant differences (see Chapter 5).

1.4 Contributions

This section lists the 4 main contributions of this thesis and the section or chapter where they are detailed.

- Development of an adaptable knowledge graph processing pipeline with independent component filters (Section 4.1)
- Proposal of a baseline knowledge graph embedding to word embedding conversion method (Section 4.2.2)
- Development and proposal of beam search post-processing techniques for dialog response diversity (Section 4.3.6)
- Development and proposal of a new evaluation method for comparing generative dialog systems (Chapter 5)

1.5 Scope

This section details the scope of the thesis, along with the relevant sections for additional information on each point.

- Dialog corpus restricted to a single well known dataset for dialog systems (Section 3.3.3)
- Knowledge graph restricted to a single well known dataset for knowledge graph embedding (Section 3.2)
- The evaluation methodology is experimental and only evaluates the relative performance of the models in this thesis (Chapter 5).

1.6 Document Structure

The rest of the thesis is structured in the following way: Chapter 2 contains a brief history of dialog systems and some of the background knowledge recommended for the following chapters. Readers familiar with dialog systems, neural networks and word embeddings may read the rest of the thesis without recourse to this chapter. Chapter 3 covers the related works including sections on word embeddings, knowledge graphs, and dialog systems. Chapter 4 covers the implementation of the dialog system, including the knowledge graph, embedding, and dialog subsystems. Chapter 5 covers the evaluation, results, and an analysis of the word embeddings used for each evaluated model, and Chapter 6 contains closing remarks and possibilities for future work.

Chapter 2

Background

Section 2.1 presents a brief historical overview of chatbots, a component of dialog systems. Section 2.2 provides information on knowledge graphs and knowledge graph embeddings, Section 2.3 provides information on word embeddings, and Section 2.4 provides information on neural networks from the basic networks up to the variant used in Chapter 4.

2.1 Historical Overview

The question of whether or not a machine can think has been asked many times in philosophy, long before the foundation of the artificial intelligence field in 1956, and continues to be debated to this day.

2.1.1 Alan Turing

In Turing’s 1950 paper on machine intelligence [9], he addresses the question “Can machines think?” and replaces it with new questions based on the Imitation Game. This game has three actors, a man (A), a woman (B), and an interrogator (C). The objective of the interrogator is to determine which is the woman and is assumed to be in another room, only communicating by typing. Turing’s new question is then “What will happen when a machine takes the part of A in this game?” which is generally interpreted as “Can machines communicate in natural language in a manner indistinguishable from that of a human being?” [10]. Turing later suggests an equivalent version of the game where a jury asks a computer questions and its role is to convince a significant portion of the jury that it is human.

2.1.2 Early Chatbots

ELIZA [11] is an early chatbot program which uses a series of scripts to process user inputs and output pre-set responses. To minimize the need for real world knowledge, ELIZA (with the DOCTOR script) simulates a Rogerian psychotherapist and provides mostly generic responses using simple pattern matching. For example, a user inputting “I need some help.” would possibly generate “What would it mean to you if you got some help?” as a response. The input is transformed based on the rule(s) associated with the matching keyword and partially outputted in the response. In this case the matching keyword is “need” and the associated rule is “...I (want/need) {0}”, which outputs “What would it mean to you if you got {0}?”. Although fairly simple, ELIZA managed to convince several of its users that they were speaking to another human.

In 1971, a similar chatbot named PARRY [12] was developed to mimic a patient with paranoid schizophrenia. The model uses scripts alongside variables to track emotions like fear and anger that increase or decrease based on the input text. The program scans for words corresponding to sensitive topics and provides different answers when the levels of anger, fear, or mistrust are too high. An experiment was performed to determine if a group of psychiatrists could distinguish the model from a patient with paranoia from their interview transcripts [13]. The experiment found that 51% of psychiatrists correctly distinguished model from patient, consistent with random guessing.

2.1.3 Loebner Prize

The Loebner Prize is an annual competition, inspired by the Turing Test, to find a chatbot which cannot be distinguished from a human. However, only the bronze medal has ever been awarded, which is given to the chatbot which most closely imitates a human through typing. The silver medal is awarded to the first chatbot which passes a Turing Test, and the gold medal to the chatbot which passes a modified Turing Test that incorporates text, audio, and visual inputs.

The Loebner Prize has been subject to much criticism, for various reasons. One of which is that the rules of the contest encourage using different tricks to fool the judges, such as the ones employed by ELIZA. Other tricks include simulating human typing (speed and mistakes), or generating simple comments by transforming the

input [14], in an attempt to pad out the conversation to hide their lack of intelligence. To mitigate this problem, Floridi et al. [15] suggest using trained judges and having multiple levels of time control, allowing conversations of different lengths.

2.2 Knowledge Graphs

A knowledge graph is a collection of structured information arranged in a graph-like format where nodes are **entities** and edges are **relations**. The data is usually stored as triples in a format such as RDF and in the form $(head, relation, tail)$ where *head* and *tail* can be entities, sets, or properties (see Section 4.1.1). Both the *head* and *tail* are assumed to be entities for the purpose of knowledge graph embedding (the rest of the triples can be safely discarded).

2.2.1 Embedding

Translation-based embedding methods are one way to embed knowledge graphs, with the simplest being TransE [6]. The goal of the model is to represent entities as $h + l \approx t$ (Figure 2.1) and accomplishes this by using gradient descent to minimize $d(h + l, t)$ where d is some dissimilarity measure such as the L_1 or L_2 -norm (Algorithm 1). The set of corrupted triples S' is constructed by replacing either the head or tail (but not both) of an existing triple with a random entity.

2.3 Word Embeddings

Word embeddings (also word vectors) are an N dimensional vector of real numbers used to significantly reduce the dimensionality from one per word (vocabulary size) to N (usually 50 - 1000). Different methods of embedding words into this reduced space may change the type of information the vectors represent.

2.3.1 Skip-gram

Skip-gram is one of the model architectures used by Mikolov et al. [3] in their Word2Vec system. The skip-gram model works by using a log-linear classifier to maximize the classification of surrounding words in a sentence given an input word.

Algorithm 1: The TransE embedding algorithm.

```

1 function TransE ( $S, E, L, \gamma, k$ ) ;
   Input : A training set  $S = (h, l, t)$ , entity and relation sets  $E$  and  $L$ , margin
            $\gamma$ , embedding dimension  $k$ .
   Output: A set of entity embeddings.
   // Initialize relations and entities
2  $l \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$  for each  $l \in L$  ;
3  $e \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$  for each  $e \in E$  ;
   // Normalize relations
4  $l \leftarrow l / \|l\|$  for each  $l \in L$  ;
5 while  $S$  is not empty do
   | // Normalize entities
   |  $e \leftarrow e / \|e\|$  for each  $e \in E$  ;
   | // Sample a minibatch of size  $b$ 
   |  $S_{\text{batch}} \leftarrow \text{Sample}(S, b)$  ;
   | // Initialize the set of pairs of triples
   |  $T_{\text{batch}} \leftarrow \emptyset$  ;
   | for  $(h, l, t) \in S_{\text{batch}}$  do
   | | // Sample a corrupted triple
   | |  $(h', l, t') \leftarrow \text{Sample}(S'_{(h,l,t)})$  ;
   | |  $T_{\text{batch}} \leftarrow T_{\text{batch}} \cup ((h, l, t), (h', l, t'))$  ;
   | end
   | Update embeddings w.r.t.
   |  $\sum_{((h,l,t),(h',l,t')) \in T_{\text{batch}}} \nabla[\gamma + d(h + l, t) - d(h' + l, t')]_+ ;$ 
14 end

```

Figure 2.1: Simplified illustration of the Skip-Gram and TransE embedding models.

The model attempts to predict the words before and after the current word in a sliding window (Figure 2.1).

2.3.2 GloVe

The GloVe [4] model learns word embeddings from a global word co-occurrence matrix, which is a matrix formed by counting the number of times each word has appeared in the context of another word. The counts are normalized and log-smoothed and the matrix reduced to a lower dimension word embedding matrix by predicting co-occurrence and minimizing the error.

2.4 Neural Networks

Neural networks are a collection of interconnected units, called neurons, organized in layers. Although the structure and training methods can differ across variants, the goal remains to propagate a signal from the input layer to the output layer to learn a function.

2.4.1 Perceptron

Perceptrons are an early ancestor to neural networks and are similar to the simplest type of neuron called the *sigmoid neuron* [16]. Each input has an associated weight which is a real number from 0 to 1 indicating the importance of that input to the

output. The output a of a perceptron is defined as:

$$\begin{aligned} z &= w \cdot x + b, \\ a &= \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases} \end{aligned} \quad (2.1)$$

where w and x are vectors whose components are the weights and inputs respectively, $w \cdot x \equiv \sum_j w_j x_j$, and b is some threshold value or *bias*. To convert a perceptron to a sigmoid neuron, the output is changed to $\sigma_g(z)$:

$$\begin{aligned} a &= \sigma_g(z), \\ \sigma_g(z) &\equiv \frac{1}{1 + e^{-z}}, \end{aligned} \quad (2.2)$$

which outputs a continuous value rather than a binary one. Figure 2.2 shows an example of a perceptron and feed-forward neural network.

2.4.2 Feed-Forward Neural Networks

A simple feed-forward neural network is a collection of neurons organized in layers, where each neuron in a layer is connected to every neuron in the next layer, starting from the inputs. The activation of the neurons in the l^{th} layer is defined as:

$$\begin{aligned} z^l &= w^l \cdot a^{l-1} + b^l, \\ a^l &= \sigma_g(z^l), \end{aligned} \quad (2.3)$$

where $z^l \equiv \sum_k w_{jk}^l \cdot a_k^{l-1} + b_j^l$ and w_{jk}^l is the weight from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer.

2.4.3 Back-propagation

A common way of training neural networks is through back-propagation in conjunction with an optimization method such as gradient descent (Algorithm 2). Error δ is calculated at the outputs by calculating the activations of neurons starting at the inputs in a forward pass. The error is then propagated backwards and then the

Figure 2.2: A perceptron (left) and a simple feed-forward neural network (right).

weights and biases are adjusted with gradient descent. Although there are many alternatives to back-propagation, such as Direct Feedback Alignment [17], Target Propagation [18], Equilibrium Propagation [19], and Particle Swarm Optimization [20], back-propagation is usually seen as the most common way to train neural networks.

Gradient Descent

Gradient descent is a first-order iterative optimization algorithm used for finding the minimum of an objective function, in this case the error. Stochastic gradient descent (SGD), a stochastic approximation of gradient descent, is often used with back-propagation to train neural networks. One of the main problems with gradient descent is that it can end up in a local minimum rather than a global one. There are many variants and extensions to SGD, such as applying momentum, averaging, AdaGrad, RMSProp, Adam, and kSGD.

2.4.4 Recurrent Neural Networks

Recurrent neural networks are a type of neural network where connections between neurons form directed cycles. RNNs are similar to FFNNs but require an additional unfold step over $t = 1, \dots, T$ time steps before the forward pass (Figure 2.3):

$$h_t^l = \sigma_g(w^l \cdot x_t^l + u^l \cdot h_{t-1}^l + b^l) \quad (2.4)$$

h is the hidden state, and u is the weight of a neuron to itself. The directed

Algorithm 2: The back-propagation training algorithm for feed-forward neural networks.

```

1 function BackPropagation ( $X, L$ ) ;
   Input: A training set  $X$ , a number of layers  $L$ .
2 foreach training example  $x \in X$  do
   // Initialize the input layer.
3    $a^{x,1} \leftarrow x$  ;
   // Forward pass.
4   foreach  $l = 2, 3, \dots, L$  do
5      $z^{x,l} \leftarrow w^l \cdot a^{x,l-1} + b^l$  ;
6      $a^{x,l} \leftarrow \sigma_g(z^{x,l})$  ;
7   end
   // Calculate the error.
8    $\delta^{x,L} \leftarrow \nabla_a C_x \circ \sigma'(z^{x,L})$  ;
   // Backpropagate the error.
9   foreach  $l = L, L-1, \dots, 2$  do
10     $\delta^{x,l} \leftarrow ((w^{l+1})^T \delta^{x,l+1}) \circ \sigma'(z^{x,l})$  ;
11  end
   // Update weights and biases with gradient descent.
12  foreach  $l = L, L-1, \dots, 2$  do
13     $w^l \leftarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$  ;
14     $b^l \leftarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$  ;
15  end
16 end

```

Figure 2.3: A recurrent neural network (left) and a visualization of the unfolding step (right).

cycles allows the network to see new inputs along with the context of previous inputs, making it well suited for information sequences. Two main disadvantages of RNNs are the exploding and vanishing gradient problems, which negatively affects its ability to model long sequences. These problems occur due to the large number of layers (unfolded) in an RNN and the activation function, which can push the error to very small or very large values, slowing down training or generating noise. Common ways of addressing these problems are gradient clipping and using LSTM cells (Section 2.4.5) for exploding and vanishing gradients respectively.

2.4.5 Long Short Term Memory

RNNs are commonly implemented with long short-term memory cells replacing regular neurons to address the vanishing gradient problem, allowing these networks to model longer sequences [21]. The original LSTM cell consists of three gates: an input gate i and output gate o which control information entering and exiting the cell, and a forget gate f which erases the memory of the cell. The cell state c and hidden state are modified in the following way:

$$\begin{aligned}
i_t^l &= \sigma_g(w_i^l \cdot x_t^l + u_i^l \cdot h_{t-1}^l + b_i^l) \\
f_t^l &= \sigma_g(w_f^l \cdot x_t^l + u_f^l \cdot h_{t-1}^l + b_f^l) \\
o_t^l &= \sigma_g(w_o^l \cdot x_t^l + u_o^l \cdot h_{t-1}^l + b_o^l) \\
c_t^l &= f_t^l \circ c_{t-1}^l + i_t^l \circ \sigma_h(w_c^l \cdot x_t^l + u_c^l \cdot h_{t-1}^l + b_c^l) \\
h_t^l &= o_t^l \circ \sigma_h(c_t^l)
\end{aligned} \tag{2.5}$$

where $\circ$ denotes the Hadamard (or *elementwise*/Shur) product and σ_h is a hyperbolic tangent.

2.5 Neural Machine Translation

Neural Machine Translation is an approach which uses large neural networks, typically some RNN variant, to translate a *source* language to a *target* language. A basic dialog generation system can be implemented as a neural machine translation model by changing the **source** to a list of questions and the **target** to their respective answers.

2.5.1 Encoder-Decoder

Encoder-Decoder is a neural network architecture consisting of two RNNs (commonly LSTMs); one *encodes* a variable-length sequence into a fixed-length context vector, and the other *decodes* a fixed-length context vector into a variable-length sequence [22]. The goal is to estimate the conditional probability $p(a_1, \dots, a_{T'} | x_1, \dots, x_T)$ of an output sequence of length T' given an input sequence of length T :

$$p(a_1, \dots, a_{T'} | x_1, \dots, x_T) = \prod_{t=1}^{T'} p(a_t | v, a_1, \dots, a_{t-1}) \tag{2.6}$$

where each $p(a_t | v, a_1, \dots, a_{t-1})$ is a softmax over all the words in the vocabulary v [23]:

$$p(y = j | x) = \frac{e^{x^\top w_j}}{\sum_{k=1}^K e^{x^\top w_k}}. \tag{2.7}$$

Once the encoder converts the input sequence into a context vector (which is the encoder state at the end of the sequence), the decoder can be sampled repeatedly to

generate new sequences.

2.5.2 Beam Search

Beam search is a heuristic search algorithm that is commonly used when sampling decoders in order to construct a complete sentence with a low error value (Algorithm 3). Breadth-first search is used to create a search tree, adding and pruning children until the number reaches a specified beam size parameter and terminating once it reaches an end-of-sequence tag. This method uses far less memory than a full breadth-first search, although the accuracy of the result depends highly on the pruning methods used and is not guaranteed to arrive at an optimal solution.

Algorithm 3: The beam search algorithm for obtaining a path based on a heuristic (or rule set) and memory size.

```

1 function BeamSearch ( $G, R, s, g$ ) ;
   Input : A graph  $G$ , a rule set  $R$ , a memory size  $s$ , a goal node  $g$ .
   Output: A beam  $b$  of nodes starting at the root of  $G$  and ending on  $g$ .
2  $o \leftarrow$  new memory of size  $s$  ;
   // Create a temporary node  $t$  and assign it the root of  $G$ .
3  $t \leftarrow G.\text{Root}()$  ;
4  $o \leftarrow t$  ;
5  $b \leftarrow t$  ;
6 while  $t \neq g$  do
7   Delete  $t$  from  $o$  ;
8   Get children from  $t$  ;
9   Delete children if they violate any rule  $\in R$  ;
10   $o \leftarrow$  children ;
11  if  $o$  exceeds memory then
12    Delete worst node in  $o$  according to  $R$  ;
13  end
14   $t \leftarrow$  best node in  $o$  according to  $R$  ;
15   $b \leftarrow t$  ;
16 end
17 return  $b$ 

```

2.6 Evaluation

2.6.1 Perplexity

Perplexity is an evaluation metric that is used during training to give some indication of how well a neural network has learned from the training data. It is defined as how well a model is able to predict example each word in target Y given source X :

$$ppl(X, Y) = \exp\left(\frac{-\sum_{i=1}^{|Y|} \log P(y_i | y_{i-1}, \dots, y_1, X)}{|Y|}\right) \quad (2.8)$$

where y_i is the i^{th} target word. During training, X and Y are taken from the training set to measure training perplexity, and between epochs they are taken from the validation set to measure validation perplexity.

Chapter 3

Related Work

This chapter provides an overview of recent work on dialog systems, word embedding methods, knowledge graph embedding methods, and joint (word and knowledge graph) embedding methods. The overall structure of the chapter is as follows: Section 3.1 reviews word embedding methods, Section 3.2 reviews knowledge graphs, their embedding methods, and some alternate related tasks. Section 3.3 reviews dialog systems and its sub-concepts.

Figure 3.1 provides a overview of the types of works present in this chapter. This figure displays the main topic of interest (Joint Text Embedding) as the root, with related concepts branching out from it. These branching concepts then have sub-concepts as children. Table 3.1 displays a time-line of the research unto knowledge graphs, word embeddings, and (knowledge graph entity) translation embeddings. Figure 3.2 displays the dependency and the evolution of dialog system features over time. Tables 3.2 and 3.3 display the evaluation methods and the datasets respectively for the works presented in Section 3.3.

3.1 Word Embeddings

Research on the vector space representation of words has been ongoing since the 1990s, with models being successfully used in information retrieval tasks such as Latent Dirichlet Allocation and Latent Semantic Analysis. Bengio et al. [24], who first proposed the term “word embeddings”, combined several earlier approaches and proposed a neural model for the continuous vector representation of words. The goal of this model is to learn the probabilities of a word given another, and extract the

Figure 3.1: Overview of the related work chapter in relation to the main topic.

distributed feature vector (or word embeddings) for each word in the vocabulary. Due to the architecture of the neural network model, the complexity for each training example is dominated by the size of the hidden and output layers, which causes the model to scale poorly. Collobert and Weston [25] are among the first to use pre-trained word embeddings for NLP tasks by simplifying the previous model and extracting the embeddings before training their own model. Their language model was fairly small (100 hidden units, 30k word vocabulary) but still took a week to train on their 637M word dataset. In order to gather embeddings for large numbers of vocabulary words and training examples, and fully exploit the power of pre-trained word embeddings, Mikolov et al. [3] proposed Word2Vec. This model significantly reduced the complexity of the language model, which greatly reduced training time. Their proposed skip-gram and continuous bag-of-words models use a single layer architecture based on the inner product between two word vectors. Released alongside their paper is a dataset consisting of 300-dimension pre-trained word embeddings for a 3M word vocabulary, trained on a 100B word dataset, along with the toolkit to train your own. This release helped popularize pre-trained word embeddings as a method to improve NLP models for a wide range of applications. One of the disadvantages of these shallow window methods is that they do not consider the co-occurrence statistics of the entire corpus, only the current context window. This means the model could fail to learn global knowledge such as repetition in the data. Pennington et al. proposed GloVe [4] to capture global corpus statistics. This model transforms the dataset into a word-context co-occurrence matrix, and then factorizes it to yield matrices for word and context vectors. Although GloVe and Word2Vec perform similarly in downstream NLP tasks, GloVe is easier to parallelize, which helps speed up training for large datasets. In 2016 Bojanowski et. al proposed fastText [26], improving the skip-gram model by splitting words into character n-grams and representing each word as a sum of the vector representation of its n-grams.

Levy et al. [27] conducted a study to evaluate many different word embedding methods including Word2Vec and GloVe using several evaluation metrics. They found that the most important difference between embedding methods is the tuning of hyperparameters, often making a bigger difference than changing algorithm or using larger amounts of data. They also found that the skip-gram with negative sampling (SGNS) model outperforms GloVe in all their comparison tasks, but it is still possible that GloVe could outperform SGNS in other tasks.

Serban et al. [1] make heavy use of word embeddings in their dialog system. They found that pre-training their model on a different corpus, and then fixing the embedding layer for the final training, outperforms pre-trained Word2Vec embeddings.

3.2 Knowledge Graphs

The origins of knowledge graphs can be traced back to The Semantic Web in the 1990s, with the term being used in psychology before then [28]. Although there are many different knowledge graphs available today, only a small subset are freely accessible, freely usable, and open domain [29]. These include: DBpedia¹, Freebase², ResearchCyc³, Wikidata⁴, and YAGO⁵. These datasets can all be retrieved in full from their respective sources and are commonly used by researchers in academia and industry for a wide variety of projects.

DBpedia is a crowd-sourced effort to extract structured facts from Wikipedia [30]. Initially released in 2007, DBpedia is updated roughly once a year due to its computationally expensive information extraction processes. It also serves as a hub, linking its entities to all four other mentioned knowledge graphs. Freebase was released in 2008 and integrated data from Wikipedia, Notable Names Database⁶, Fashion Model Directory⁷, MusicBrainz⁸ and provided users with an interface to edit structured data to add their own [5]. Although Freebase shut down in 2015, a snapshot of its data is still available and contains nearly 2 billion RDF triples. Wikidata was launched by the Wikimedia foundation in 2012 as a way to unify data from all Wikipedia projects [31]. The Wikidata project began integrating Freebase data after it shut down, which accelerated its growth significantly. YAGO [32] is an ontology started in 2007, which was built by automatically extracting data from Wikipedia and WordNet [33]. Cyc [34] is a proprietary project started in 1984 with the goal to gather and store common sense knowledge. In 2006 Cycorp released ResearchCyc as a free alternative for researchers, replacing the now obsolete OpenCyc experiment.

¹<http://wiki.dbpedia.org/>

²<https://developers.google.com/freebase/>

³<http://www.cyc.com/platform/researchcyc/>

⁴<https://www.wikidata.org/>

⁵<http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/>

⁶<http://nndb.com/>

⁷<http://www.fashionmodeldirectory.com/>

⁸<https://musicbrainz.org/>

Freebase and WordNet are often used as the datasets of choice for translation-based embedding methods (see Section 3.2.1). Table 3.1 shows a timeline that includes the deployment of knowledge graphs and the publication of translation-based and word embedding methods. These and several other knowledge graphs were evaluated [35] in the work prior to TransE [6]. Since TransE is the “root” publication of the translation-based embedding line, the publications that followed it use Freebase and WordNet in order to use it as a baseline.

3.2.1 Knowledge Graph Embedding

Translation-Based Embedding

Translation-based embeddings model relations between entities as translations in the embedding space and are evaluated on link prediction tasks on various datasets. Although not the first method of embedding knowledge graphs, TransE [6] is one of the first translation-based methods and a very influential algorithm in its field. Due to its influence on the rest of the translation-based methods and its position as a baseline method, additional information can be found in Section 2.2.1. TransE is a simple optimization method that embeds entities in a vector space and relations as translations between entities. One of the flaws of this model is its inability to deal with reflexive/one-to-many/many-to-one/many-to-many relationships. Wang et. al [7] proposed TransH to solve these issues by representing relations with two vectors, the norm and the translation, and translating entities on hyperplanes. Since both TransE and TransH assume relations and entities are in the same vector space, Lin et. al proposed TransR [36] to map entities and relations in distinct vector spaces, and performing the translations in the relation space. TransH and TransR significantly outperform TransE and all previous baselines; however, they still struggle to predict entities on the “many” side of one-to-many/many-to-one relations. The first major improvement in this area came from PTransE [37], which introduced two significant changes to the original TransE algorithm: adding inverse relations to the dataset, and composing relations into paths. This method predicts inference patterns of direct relations and their paths but does not model more complex (indirect) inference well, such as $Queen(e) \rightarrow Female(e)$.

There are many more variations of translation-based knowledge graph embedding

Table 3.1: Timeline of publication for knowledge graphs, word embeddings, and translation embeddings surveyed in Section 3.2.

Year	Knowledge Graph	Word Embedding	Translation Embedding
1998	WordNet [33]		
2003		Bengio [24]	
2006	ResearchCyc [34]		
2007	DBpedia [30], YAGO [32]		
2008	Freebase [5]	Collobert [25]	
2013		Word2Vec [3]	TransE [6]
2014	WikiData [31]	GloVe [4]	TransH [7], TransM [39]
2015			PTransE [37], TransD [38],
			TransR [36]
2016		fastText [26]	TransG [40]

methods (TransD [38], TransM [39], etc.) but one stands out as the current state-of-the-art: TransG [40]. This model addresses the issue of multiple relation semantics by employing a Bayesian non-parametric infinite mixture embedding model to automatically discover semantic clusters of a relation. This allows relations to be represented by multiple vectors for translating different clusters of entities. Other methods achieve similar results with manifold-based [41] and density-based [42] techniques.

Joint Text Embeddings

Another way to make use of knowledge graphs is using them to enhance the quality of word embeddings extracted from other sources, such as Word2Vec, by enriching them with entity relations and encoding them into a similar model. Celikyilmaz et al. [43] proposed two models which apply this technique to semantic tagging with favorable results. The context constrained model (CECM) implicitly encodes the entity relations by including entities in the objective of the context window, to maximize the log likelihood of query tokens. The relation encoding model (REM) explicitly encodes these relations, inspired by the translation-based methods in Section 3.2.1, by adding the log likelihood of the embeddings to their objective function. Multi-word entities

need to be compounded into a single word in order to be used in this model. Socher et al. [44] deal with these multi-word entities by representing each one as an average of their word vectors. They use these entities as input to a neural tensor model, which replace the standard neural network layer with a bilinear tensor layer that directly relates two entities, and outputs a confidence score for that triple. The scores are used for extracting common sense reasoning from the knowledge graph (called knowledge graph completion). The model can be used with randomly initialized word vectors but is further improved with the introduction of pre-trained vectors. Wang et al. [45] propose a model that joins the Word2Vec skip-gram model with TransE by using an alignment model which uses both Wikipedia anchors and entity names to match entities to words. Wikipedia anchors are used in order to disambiguate entities. All three models are trained simultaneously using multi-threaded versions of their respective algorithms. The final model is evaluated on knowledge graph completion, relation extraction, and analogical reasoning tasks. Toutanova et al. [46] improve upon the method of representing words and knowledge graph entities in the same vector space by modeling and linking related relation paths. These paths are treated as a sequence of words, rather than a single atomic unit, in order to account for co-occurrence in the knowledge graph.

Xie et al. [47] approach the combination of entities differently, using a convolutional neural network encoder to build entities from their textual descriptions. This approach allows for both pre-trained word and knowledge graph embeddings to be used, which increases the accuracy of the model for entity classification. Similarly, Xiao et al. [48] combine translation-based embedding with their definitions from Freebase and Wordnet by computing their topic vectors and mapping the entities onto a semantic hyperplane.

Zhang et al. [49] combine a knowledge graph with text and image components to create an item latent vector for their recommender system. Their Collaborative Knowledge Base Embedding model (CKE) uses a Bayesian formulation of TransR for knowledge graph embedding and combine it with the embeddings for words and images in their joint neural model. Interestingly, the model architecture suggests that embedding methods could be modified or removed, and that any other components could be added as long as there exists a way to embed that information.

3.2.2 Entity Recognition and Disambiguation

Knowledge graphs are not of much use to a dialog model if the model cannot properly recognize and disambiguate entities. It must be able to recognize when a known entity is being mentioned in conversation, since its full name might not be used (e.g., referring to a professional sports team by its city's name). If the mention refers to multiple entities then it will need to be disambiguated to ensure the model receives the right information.

Huang et al. [50] propose a deep semantic relatedness model for entity disambiguation which works by outputting the similarity between two entities. This model assumes that information within the same context will belong to the same topic. Entity names are split into letter tri-grams, reducing the size of the input vector considerably, since knowledge graphs can contain millions of entities. This also means that the model can handle out-of-vocabulary words and newly created entities. Feeding into the input vector is a feature vector which contains linked entities, relations, entity types, and a bag-of-words representation of its description. Entity embeddings can also be extracted from this model once it is trained [51].

3.2.3 Slot Filling

Knowledge graphs can also be built dynamically, and queried directly during conversation. Li et al. [8] present a statistical language understanding approach for user-centric knowledge graphs in conversational dialogs. These graphs are built from knowledge gained during the conversation and can be queried later to give the machine memory about the user it is interacting with. The proposed approach combines three natural language understanding tasks: personal assertion classification, relation detection, and slot filling. The latter is also a common task in knowledge graph completion.

3.3 Dialog Systems

Although dialog systems have been around for a long time (see Section 2.1), generative dialog systems are fairly new. Past dialog systems included several modules such as the Language Interpreter, State Tracker, Response Generator, and Natural Language Generator. End-to-end generative systems are meant as a replacement for

Figure 3.2: Evolution of dialog models covered in Sections 3.3 and 3.3.1.

standard dialog systems, leveraging the abilities of neural networks to handle the tasks themselves. Generative dialog systems can be categorized into two different categories based on their objective: closed-domain or open-domain. Closed-domain (or goal-driven) systems tend to focus on specific domains such as technical support, or computer game characters [63], while open-domain systems can access any topic of discussion.

Early attempts at developing these systems, such as the one proposed by Ritter et al. [60], adapted a phrased-based SMT model to the response generation task, showing that it could be approached as a source-target translation problem. With the introduction of sequence-to-sequence learning (Seq2Seq), Sutskever et al. [23] showed that a relatively simple model of a multi-layered LSTM with a limited vocabulary could outperform previous phrase-based systems on machine translation tasks. Their approach uses the RNN encoder-decoder model proposed by Cho et al. [22] which encodes a variable-length sequence into a fixed length vector, and then decodes a given fixed-length vector back into a variable-length sequence. This model can be used to score a pair of input and output sentences, but more importantly, can be used to generate new sequences. The decoder generates the output sequence by predicting the next symbol given its current hidden state, and repeating for a set number of symbols or until the end-of-sentence symbol appears. This approach can get “stuck” predicting a suboptimal sequence since it will always greedily pick the next best symbol. A common way to address this problem is by using beam search [23] which maintains a small list of the most probable output sequences at each time-step until they reach the end-of-sentence symbol, and then chooses the highest scoring output from the list of complete sequences. This motivated the work on beam search in Section 4.3.6.

3.3.1 Improving Response Generation

Although beam search helps produce a better scoring output sentence, the result is often generic, repetitive, short-sighted, and lacking in substantial information; which makes conversations with these models generally uninteresting [62]. Several approaches exist for tuning the output of a neural model in order to produce higher quality responses.

Clustering

Google’s SmartReply [57] uses semi-supervised learning to categorize responses into semantic intent clusters, which are learned clusters of similar responses (e.g., “ha, ha”, “lol”, and “that’s funny”). They then use normalization to penalize generic responses, omit redundant responses based on their intent, and ensure both positive and negative answers in the response set. The result is a set of three unique, high-quality responses presented to the end user. This system’s normalization method can be generalized to open-domain generative dialog responses but discovering open-domain semantic intent clusters could be more difficult. Ritter et al. [64] propose an approach to model Twitter conversations by clustering utterances into dialog acts by extending a multi-document text summarization model. One of the results of this work is a transition diagram that models the flow of conversations on Twitter using probabilities as transitions and topic clusters as states. Building this type of tree on a dialog corpus might help a dialog system output a more relevant response to the given context. It could also help such a system know when to reply to user and when the conversation is over, in cases where the system is not designed to reply to every message it sees (dialog triggering).

Context

Sordoni et al. [58] address the problem of response generation by introducing context, defined as the sentence before the one the model is replying to, which requires a dataset composed of triples (context, message, response), also known as three-turn dialog. They propose three context-based generation models as well as three feature extraction methods in order to help capture and promote contextual information in the replies. The best performing model concatenates the linear mappings of the bag-of-word representations of the context and message and feeds it into the encoder network. The model also makes use of additional features extracted from the original triples by counting [1-4]-gram matches between context and reply, and between message and reply. Serban et al. [1] extend the hierarchical encoder-decoder model for tracking context in their dialog model. This model works by using two encoders: one to track sequences of words (as before) and one to track sequences of sequences. This gives the model a better understanding of the flow of the conversation. They also experimented with using a bidirectional RNN as the first encoder in order to help with longer sequences, which outperforms regular a RNN in their model.

Personality

A prevailing issue with responses from neural dialog models is speaker consistency and lack of coherent personality. Since training these models requires large datasets that usually contain many different speakers, responses lack the consistency of a single persona needed to model human-like behavior. This makes it difficult for these models to pass the Turing test [9] [2]. Li et al. [56] propose two persona-based models: the Speaker model and the Speaker-Addressee model. The Speaker model attempts to capture speaker information through a user embedding layer which is learned jointly with word embeddings in the target. This layer groups similar users and infers answers based on proximity in the vector space. A benefit of this is that user information does not have to be explicitly present, which is useful since the dataset does not contain information for each attribute and every user. The Speaker-Addressee model attempts to capture how a person’s speech might change depending on whom they are addressing and does so with a user matrix. This matrix represents each speaker with a K dimensional vector by linearly combining user vectors v_i and v_j to model the speech of user i towards user j . Both models outperform the Seq2Seq baseline but still suffer some inconsistencies in responses to questions referring to the same user attribute (e.g., “How old are you?” and “What is your age?”).

Reinforcement Learning

Another way to improve the responses from dialog models is to apply reinforcement learning. Li et al. [55] first use mutual information score to reward less generic responses during training. They then apply reinforcement learning by simulating a conversation between virtual agents, using policy gradient methods to reward sequences that are more diverse and informative. An issue with this model is that the dialog can sometimes get stuck in cycles of length greater than one, which is a general issue with models simulating virtual conversations since increasing the number of simulated turns grows the number of cases to consider exponentially.

3.3.2 Dialog Models

Retrieval-Based

Retrieval-based (or deterministic) dialog models work by outputting a response from a repository of predefined responses. Given a new input context, they score their

repository and output the response with the highest probability. Retrieval-based models work well on closed-domain problems like the Ubuntu Dialog Corpus [65]. Kadlec et al. [66] achieved state-of-the-art performance on the dataset by creating an ensemble model consisting of 10 CNNs, 11 LSTMs, and 7 bi-directional LSTMs trained with different meta-parameters.

Generative

Generative models differ from retrieval-based ones in that they are able to create new sentences that were not necessarily present in the training data. Generative models are also better positioned to handle new (out-of-domain) input sentences [67], but do not achieve the performance of retrieval-based models in a closed domain. Google’s SmartReply [57] uses a combination of both models, generating a list of new outputs which are scored and chosen by the retrieval model.

In their work on conversational modeling, Vinyals and Le [2] present a simple approach for creating a language model using the sequence to sequence framework. For their experiments, they use two different datasets: the closed-domain IT Helpdesk Troubleshooting dataset, and the open-domain OpenSubtitles dataset. The model works by treating a conversation as a machine translation problem and predicting a sentence given the previous one. This simple model produces decent results on its own but can be extended in many ways to produce potentially better results (Figure 3.2 shows the related works which are extensions of the sequence-to-sequence model).

Attention

The attention mechanism is an approach introduced by Bahdanau et al. [59] to improve the basic encode-decoder architecture by allowing it to automatically search for parts of a source sentence that are relevant for predicting a target word. In this model, the decoder emulates searching through a sequence of annotations, created by the encoder, which uses their weighted sum to create the fixed-length context vector. Their model also addresses the problem that RNNs have with longer sentences by using a bidirectional RNN as the encoder. Luong et al. [53] further improve the attention mechanism by introducing two different classes of attention: global and local. Global attention is similar to the previous method where every word is considered, while local attention focuses only on a small context window. While previously used mostly for NMT (and image captioning/generation), Shang et al. [52] imported the

attention mechanism to response generation and proposed a hybrid model which combined local and global attention. The proposed model uses two encoders to generate local and global context, which are then merged into a single context vector.

The attention with intention (AWI) model proposed by Yao and Zweig [54] adds a third RNN, called intention, to the encoder-decoder architecture, while keeping the attention mechanism in the decoder. The intention network serves as a middle man between the encoder and decoder, tracking the dialog as it moves between turns, allowing better modeling of multi-turn conversations.

Gu et al. [61] extend the attention model by adding copying, where certain segments of the source sentence are directly copied into the target. This feature attempts to simulate the human tendency to repeat entity names or phrases while communicating. The model is also better at handling words outside of its vocabulary, since it can make use of them by copying the unknown words into the target sentence. This model was evaluated on both text summarization and dialog generation.

Evaluation

A comparison of the evaluation methods for the NMT and generative dialog systems referenced in Section 3.3 and Figure 3.2 is presented in Table 3.2.

Automatic Metrics

Automatic evaluation of generative dialog systems is still an open problem. Most of the recent works in this field use metrics adopted from machine translation and text summarization to generate a score for their model’s responses. In their survey of current available evaluation metrics, Liu et al. [68] found that these methods are largely unsuitable and correlate weakly or not at all with human judgment. Currently, the most popular automatic evaluation metrics after perplexity are word overlap-based metrics such as BLEU [69] and METEOR [70], both of which designed for machine translation. Although the dialog generation task (and models) looks similar to machine translation, the space for valid responses to a given context is much larger, and responses can be distinct enough from each other that they do not share common words or semantics.

Word perplexity measures how well the model is at predicting a sample, and is often used alongside other metrics when evaluating dialog systems. It is defined as the

Table 3.2: Evaluation metrics used for the surveyed NMT and generative dialog systems.

Lead Author	Year		Concept	Perplexity	Word Overlap	Human	Other
Ritter	2011	[60]	SMT to Response Generation		✓	✓	
Cho	2014	[22]	SMT to Encoder-Decoder		✓		
Sutskever	2014	[23]	Sequence-to-Sequence		✓		
Bahdanau	2014	[59]	Attention		✓		
Luong	2015	[53]	Local/Global Attention		✓		
Shang	2015	[52]	Hybrid Attention			✓	
Yao	2015	[54]	Attention with Intention	✓			
Vinyals	2015	[2]	Neural Conversation	✓			
Serban	2015	[1]	Hierarchical Encoder-Decoder	✓			
Sordoni	2015	[58]	Context Sensitivity		✓	✓	✓
Li	2016	[56]	Neural Personality	✓	✓	✓	
Kannan	2016	[57]	Smart Reply	✓			✓
Gu	2016	[61]	Copying				✓
Li	2016	[55]	Reinforcement Learning			✓	
Li	2017	[62]	Adversarial Learning			✓	

probability that the model will generate the ground truth next utterance. However, this metric is model dependent, as the model generating responses is also the one evaluating it, and is usually avoided when the goal is evaluating the success of a longer conversation with the dialog model [55]. Perplexity is used in the evaluation of dialog systems built for this thesis research.

Human Evaluation

In order to get a better idea of their system’s performance, many researchers turn to human evaluators; however, finding consistent criteria to evaluate utterances can be difficult [63], and there are no established standards that can be used to compare across systems. Human evaluation often relies on crowd-sourced judges [62] [55] [56] [58] [60] or experienced annotators [61] to perform a simple pairwise comparison (“Which is better?”) between responses from the test set and responses generated by the model. Judges are sometimes asked to give ratings on several categories in order to provide more insight on the performance of the model. How the results of these comparisons and ratings are handled varies, with the simplest result being a percentage score to give a general idea of the quality of the model. The same pair of responses can be given to several different judges in order to achieve a more reliable result [71] [62]. Evaluation using pairwise comparisons is usually preferred to the traditional Turing Test [9] method since it is faster and does not rely on the judge’s ability to ask proper questions [10].

3.3.3 Dialog Corpora

Recent progress in dialog systems can be attributed to the availability of large public datasets, increased computing power, and developments in neural models and machine learning architectures [63]. In order to facilitate training of these systems, Serban et al. [63] conducted a survey of the available corpora. For open-domain systems, datasets most often fall into either of these two categories: datasets mined from social media platforms such as Twitter or Sina Weibo, or datasets derived from one or more of the movie subtitle datasets.

To clarify conflicting definitions, a turn of dialog here refers to a single utterance in the conversation (with no response), two turns of dialog is a message-response pair with different speakers, and higher turn dialog alternates speakers for the duration

Table 3.3: Dialog corpora used for the surveyed NMT and generative dialog systems. Starred (*) quantities are estimated based on the average number of words and sentences in related data.

Lead Author	Year		Dataset Name	Turns (avg)	Sentences	Words	Notes
Ritter	2011	[60]	Twitter	2	650M	6.5B*	First two utterances in a conversation.
Cho	2014	[22]	WMT '14	2	28M*	766M	Processed subset.
Sutskever	2014	[23]	WMT '14	2	24M	652M	Processed subset.
Bahdanau	2014	[59]	WMT '14	2	13M	348M	Processed subset.
Luong	2015	[53]	WMT '14	2	9M	226M	Processed subset.
Shang	2015	[52]	Sina Weibo	2	64K*	800K	Chinese social media platform.
Yao	2015	[54]	Helpdesk	9.7	97K	4.6M	In-house, closed-domain.
Vinyals	2015	[2]	Helpdesk	2	75K*	33M	In-house, closed-domain.
			OpenSubtitles	2	88M	1.3B	Processed and doubled.
Serban	2015	[1]	SubTle	3	11M	93M	Pre-training set.
Serban	2015	[1]	MovieTriples	3	736K	13M*	Pre-trained on SubTle.
Sordoni	2015	[58]	Twitter	3	87M	1B*	Triples containing a frequent bigram.
Li	2016	[56]	Twitter	3	74M	740M*	Sample of “frequent speakers”.
			IMSDb	3	70K	-	Pre-trained on OpenSubtitles
Kannan	2016	[57]	Emails	1-2	238M	-	Includes 153M messages with no response.
Gu	2016	[61]	Sina Weibo	2	4.8M	-	Chinese social media platform.
Li	2016	[55]	OpenSubtitles	2	800K	12M	Subset with less vague responses.
Li	2017	[62]	OpenSubtitles	2	70M	1B	No details provided, full dataset.

of the conversation. Table 3.3 presents the datasets used for Section 3.3 (see also Figure 3.2 and Table 3.2) and gives a general idea of the size and amount of filtering done on each dataset. For example, all NMT works surveyed used the WMT '14 dataset, but each filtered more and more sentences in an effort to remove more noise in the data. The same is true with OpenSubtitles. One of the problems with that dataset is that the speaker is not indicated, which led to the assumption that each turn changed speaker [2]. Another problem is the presence of questions that lead to vague answers such as “I don’t know” [55] and removing these shortens the size of the dataset considerably (1B to 12M).

3.4 Summary

This chapter covers the work related to the improvement of dialog systems and the development of translation-based, word, and joint embeddings. Translation-based embedding methods, such as TransE and TransH embed a knowledge graph to a vector space to infer common sense knowledge. The resulting entity embeddings have been used jointly with word embeddings to improve applications such as recommender systems. Smaller knowledge graphs in the form of user personalized knowledge graphs have also been directly queried by dialog systems to gather information about its user. Although entity embeddings have been used on a variety of tasks, they have not, to the best of my knowledge, been applied to dialog system nor applied directly to a neural model once converted to word embeddings. This thesis uses the following work directly:

- The Freebase knowledge graph [5].
- The TransE algorithm for knowledge graph embedding [6].
- The TransH algorithm for knowledge graph embedding [7].
- GloVe word embeddings [4].
- Word2Vec word embeddings [3].
- The OpenSubtitles dataset [72].
- The global/local attention sequence to sequence model [53].

The questions from Section 1.3 (restated below) were motivated in part by the following work:

1. How can knowledge graphs be used to improve dialog systems?
 - Inspired by personal knowledge graph population [8].
 - Relates closely to the work on knowledge graph embedding (Section 3.2.1).
 - Addressed in Section 4.2.1.
2. How can the knowledge graph be preprocessed in order to conserve information and be within hardware constraints?
 - Relates closely to the preprocessing techniques used by Wang et al. [45].
 - Relates closely to the Freebase parsing scripts by Chah [73].
 - The following contribution extends the above works: “Development of an adaptable knowledge graph processing pipeline with independent component filters”.
 - Addressed in Section 4.1.
3. How can knowledge graph (entity) embeddings be converted to word embeddings?
 - Inspired by Socher et al. [44].
 - Relates closely to the work on joint text embeddings (Section 3.2.1).
 - The following contribution extends the above works: “Proposal of a baseline knowledge graph embedding to word embedding conversion method”.
 - Addressed in Section 4.2.2.
4. How can the dialog model be tuned to generate sufficiently different responses required for evaluation?
 - Relates closely to the beam search extension by Li et al. [74].
 - The following contribution extends the above work: “Development and proposal of beam search post-processing techniques for dialog response diversity”.
 - Addressed in Section 4.3.6.

5. How can each model be evaluated in a way which minimizes bias?

- Inspired by retrieval-based dialog systems (Section 3.3.2).
- Driven by the lack of consensus on evaluation methods for generative dialog systems (Section 3.3.2).
- Related closely to evaluation of the retrieval-based dialog system by Lowe et al. [65]
- The following contribution extends the above work: “Development and proposal of a new evaluation method for comparing generative dialog systems”.
- Addressed in Chapter 5.

Chapter 4

Implementation

A basic generative dialog system built using the sequence-to-sequence framework has a vast amount of room for improvement, with just some of these improvements covered in Chapter 3. Word embeddings are a feature learning technique for improving the dialog system by replacing word IDs with a lower dimension vector representation, allowing each sentence to contain more information. A common way to utilize these word embeddings is with pre-trained embeddings trained from unstructured datasets, such as a news corpus. These embeddings form the embedding layer of the dialog system, which translates the word IDs to vectors. Knowledge graphs provide structured information about its entities with relation to other entities, and can also be embedded into a similar vector space. This chapter covers the implementation required to answer the following questions:

1. Does a dialog system trained using word embeddings derived from a knowledge graph outperform a dialog system trained using GloVe or Word2Vec word embeddings?
2. Do the different methods of embedding a knowledge graph have a significant impact on the quality of responses from the dialog system?

Question 1 can be generalized to: “Is a structured knowledge source better than an unstructured knowledge source for creating word embeddings?”, with dialog systems used as way to evaluate the two in practice. Since these are established methods with proven results (see Section 3.1), GloVe and Word2Vec will be referred to as the baseline word embedding methods. Question 2 asks if the translation embedding methods discussed in Section 3.2.1 perform significantly better or worse when applied

Figure 4.1: High level overview of the implemented components, detailed in Sections 4.1, 4.2, and 4.3 respectively.

to dialog systems. These methods were already compared each other in the related works, but on a much smaller sample of Freebase, and evaluated based mostly on their graph completion capabilities. In order to answer these questions the following steps require implementation:

1. Preprocess the knowledge graph to extract a set of training triples.
2. Preprocess the dialog corpus.
3. Embed the knowledge graph in a vector space using translation embedding.
4. Convert the embeddings to word vectors.
5. Train the dialog system with the word embeddings.
6. Repeat steps 3-5 for each translation embedding method implemented.
7. Repeat step 5 for each baseline word embedding method.

Figure 4.1 provides an overview of the steps above, grouped into subsystems, Section 4.1 explains the preprocessing pipeline used to extract triples from the knowledge graph (step 1.), Section 4.2 explains the reasoning and process for using knowledge graphs and translation embedding methods with dialog systems (step 3.), Section 4.2.2 covers the process of converting entity embeddings to word embeddings (step 4.), Section 4.3.1 explains the preprocessing of the dialog corpus (step 2.), and Section 4.3 explains the implementation of the dialog system (step 5.). Chapter 5 contains the

Figure 4.2: Overview of the knowledge graph subsystem pipeline.

evaluation of the dialog systems trained in steps 6-8, and answers the two questions asked at the beginning of this chapter. All code written for this thesis is available in a public repository¹.

4.1 Knowledge Graph Subsystem

Large raw datasets have to be carefully pruned or modified in order for a neural network to make the most out of its information. Since the steps taken are largely dependent on the data itself, preprocessing techniques vary significantly from dataset to dataset. Details of the exact steps taken are often vague, which makes reproducing results much harder, especially when the data is not included with the research. This could be due to the use of proprietary data, or from file size limits of the code’s hosting

¹https://github.com/briancarignan/dialog_thesis

service (commonly GitHub²). Processing extremely large datasets presents many of its own challenges as well. This section presents a broad strategy for handling the preprocessing of large datasets as well as details for better reproducibility of results.

Freebase was selected as the knowledge graph³ to be used for creating word embeddings, using the latest snapshot of the entire dataset as of June 2017. Freebase was chosen due to several factors including: size, the resources dedicated to it, and its use in related works (see Section 3.2). The preprocessing technique described in this section is inspired by Wang et al. [45] with some code ported from the freebase-triples repository⁴. All code for this section is written in Java. The goals of this section are as follows:

1. Reduce the data to a manageable size in order to be usable within time and memory constraints at later stages (see Section 4.2).
2. Ensure the remaining data is high quality by removing information that is less likely to be useful for later stages.
3. Create a reusable information extraction process with independent components which can be adapted to alter the processed data and fit different knowledge graphs.

The raw Freebase dataset poses numerous challenges common to machine learning applications. Some of these challenges are:

1. Due to size:
 - Getting a clear overview of the data present in the original 3.1B lines to understand what to remove.
 - Handling the data within memory constraints, for preprocessing and for training.
2. Due to noise:
 - Identifying entities which are not well represented by their English labels (e.g., song names).

²<https://github.com/>

³<https://developers.google.com/freebase/>

⁴<https://github.com/nchah/freebase-triples>

Table 4.1: Results from preprocessing the Freebase knowledge graph.

Step	Uncompressed Size	Lines	Entities	Relations
Raw	425.2GB (100%)	3,130,753,066 (100%)	86,054,153 (100%)	14,828 (100%)
Trim	194.9GB (45.8%)	3,130,753,066 (100%)	86,054,153 (100%)	14,828 (100%)
Coarse	9.5GB (2.2%)	204,379,257 (6.5%)	45,282,683 (52.6%)	10,464 (70.6%)
Fine	8.6GB (2.0%)	188,226,857 (6.0%)	39,036,120 (45.4%)	3,094 (20.9%)
Threshold	2.1GB (0.5%)	45,241,279 (1.4%)	9,666,330 (11.2%)	2,843 (19.2%)

- Identifying relations describing events which are hard to predict (e.g., date of death).
- Identifying unrelated data such as metadata (e.g., user information) or entity properties (e.g., age, height).

In order to make use of the knowledge graph with the available memory and within a reasonable amount of time, the size of the dataset needs to be reduced significantly. The data is reduced in steps in order to analyze the impact that each has with respect to the size, line count, entities, and relations in the dataset. Related works dealing with Freebase (see Section 3.2.1), as well as repositories, such as freebase-triples, help with identifying noise in the data. Additionally, the filters in the sections below were updated incrementally, with the effects on the resulting processed dataset analyzed during preliminary testing. This method also addresses memory concerns, as each step is saved to disk and the data is streamed line by line. Steps which require storing entities in memory are performed after a significant amount of the data is removed. Table 4.1 presents the reduction of the data for each preprocessing step relative to the original raw dataset.

Section 4.1.1 explains the raw dataset format, and Sections 4.1.2, 4.1.3, 4.1.4 describe the filters used to reduce the size of the data. Figure 4.2 displays an overview of the implementation for this section.

4.1.1 Raw Dataset

The raw dataset is the latest available version of Freebase, and although currently deprecated, it is still available for download [75]. The total size of the dataset is

Table 4.2: Sample triples from the raw Freebase dataset. Columns are separated by a tab character.

$t[0]$	$t[1]$	$t[2]$	$t[3]$
<http://rdf.freebase.com/ns/m.03h14zy>	<http://rdf.freebase.com/key/en>	“timisat.canal”	.
<http://rdf.freebase.com/ns/m.03h14zy>	<http://rdf.freebase.com/ns/geography.river.length>	“24.0”	.
<http://rdf.freebase.com/ns/m.03h14zy>	<http://rdf.freebase.com/key/wikipedia.en_id>	“14906694”	.
<http://rdf.freebase.com/ns/m.03h14zy>	<http://rdf.freebase.com/key/wikipedia.ro_id>	“197859”	.
<http://rdf.freebase.com/ns/m.0374mmq>	<http://www.w3.org/2000/01/rdf-schema#label>	“+/-”@en	.

32.2 GB compressed (452.2 GB uncompressed), and contains 3.1B RDF triples. To conserve hard drive space the data is kept compressed and decoded during run time. The raw triples consist of 4 tab separated fields: head, relation, tail, “.”. These fields are also referred to as $t[0]$, $t[1]$, $t[2]$, $t[3]$ respectively in the tables and algorithms below. Table 4.2 shows some sample raw triples.

4.1.2 Trim Filter

The goal of the Trim filter is to reduce the overall size of the dataset to a more manageable size by removing excess characters but leaving in all potentially usable data. This is done by removing text matching certain regular expressions relating to: URL identifiers, brackets, type specifiers, and the last column ($t[3]$) (see Algorithm 4). The result is a reduction of over 50% of the original dataset. Table 4.3 shows the effects of the trim filter on the raw triples from Table 4.2.

This leaves a dataset which can be used for other tasks, such as converting entities to English labels, without having to reuse the raw dataset. A vocabulary of 46.6M entities is created by scanning for “type.object.name” relations involving an entity as the head and an English label as the tail (represented by “@en”). Entities consist of IDs of the format: “m.” followed by letters, numbers, or underscores (e.g., m.03h14zy). This vocabulary, EL , represents the complete set of English entity labels in the dataset, and is used as a lookup table in later steps. Each processing step below this one is independent and can be performed in any order.

Algorithm 4: The Trim filter reduces the size of a raw Freebase dataset without reducing the number of lines.

```

1 function RawToTrim ( $R$ ) ;
   Input  : A raw Freebase dataset  $R$ 
   Output: A trimmed dataset  $T$ 
2 foreach  $triple\ t \in R$  do
   // remove URLs from triples
3    $t.ReplaceAll('http://rdf.freebase.com/', '')$  ;
4    $t.ReplaceAll('http://rdf.freebase.com/ns/', '')$  ;
5    $t.ReplaceAll('http://www.w3.org/[0-9]*/[0-9]*/[0-9]*.*', '')$  ;
6    $Split(t, '\t')$  ;
7   for  $i \leftarrow 0$  to 3 do
8      $t[i].ReplaceAll('(.*)\#'$  ,  $'')$  ;           // remove schema indicator
9      $t[i].ReplaceAll('^<'$  ,  $'')$  ;           // remove angled brackets
10     $t[i].ReplaceAll('\$>'$  ,  $'')$  ;
11  end
12   $T \leftarrow t[0] + '\t' + t[1] + '\t' + t[2] + '\n'$  ;
13 end
14 return  $T$ 

```

Table 4.3: The same triples as Table 4.2 after running the Trim filter. All triples are significantly shorter and the last column was removed entirely.

$t[0]$	$t[1]$	$t[2]$
m.03h14zy	key/en	"timisat_canal"
m.03h14zy	geography.river.length	"24.0"
m.03h14zy	key/wikipedia.en_id	"14906694"
m.03h14zy	key/wikipedia.ro_id	"197859"
m.0374mmq	label	"+/-"@en

4.1.3 Coarse Filter

The coarse filter selects all triples in the form “entity, relation, entity” where both entities are present in the vocabulary created in the previous step, and discards the rest. The discarded data consists of metadata and other descriptive information which are not necessary for embedding. The largest possible subset of Freebase which can be used in existing translation-based embedding algorithms (all “entity, relation, entity” triples) is 338.6M triples, 86M entities, and 14.8k relations. After limiting the dataset to triples containing only entities present in the English label, EL , vocabulary, the size is reduced to 204M triples, 45.3M entities, and 10.5k relations. Algorithm 5 presents pseudocode for the Coarse and Threshold (Section 4.1.5) filters.

Algorithm 5: The Coarse filter selects all triples which start and end with an entity that has an English label and the Threshold filter removes entities outside a minimum and maximum threshold.

```

1 function TrimToCoarse ( $T, D, m, M$ ) ;
   Input : A trimmed dataset  $T$ , dialog corpus vocabulary  $D$ , minimum
           frequency threshold  $m$ , maximum frequency threshold  $M$ 
   Output: A coarse-filtered dataset  $C$ 
2  $V < entity, label > \leftarrow \text{ExtractEntityVocabulary}(T, D)$  ;
   // Threshold filter
3  $E < frequency, List < entity > \leftarrow \text{ExtractEntityFrequencies}(T)$  ;
4 foreach frequency  $f \in E$  do
5   | if  $f < m$  or  $f > M$  then
6   | | delete all  $E.get(f)$  from  $V$  ;
7   | end
8 end
   // Coarse filter
9 foreach triple  $t \in T$  do
10  |  $\text{Split}(t, "\backslash t")$  ;
11  | if  $t[0], t[2].\text{StartsWith}('m.')$  and  $V$  contains  $t[0], t[2]$  then
12  | |  $C \leftarrow t[0] + '\backslash t' + t[1] + '\backslash t' + t[2] + '\backslash n'$  ;
13  | end
14 end
15 return  $C$ 

```

Table 4.4: Sample triples after running the Coarse filter.

$t[0]$	$t[1]$	$t[2]$
m.01067m9v	award.award_nomination.nominated_for	m.0zdjzxm
m.01068f6j	film.performance.actor	m.0bx_v8d
m.01068pg	music.recording.releases	m.0314xwk
m.01068t59	user.micahsaul.advertising.ad_contribution.role	m.016g5h
m.01068v7q	freebase.valuenotation.is_reviewed	m.0y85w1d
m.01068v7q	base.schemastaging.holiday_fixed_date_observance_rule.month	m.03_ly
m.017dcd	tv.tv_program.regular_cast..tv.regular_tv_appearance.actor	m.06v8s0

4.1.4 Fine Filter

This step requires the most knowledge of the data in order to reduce the amount of noise present. The data was scanned for relations which occurred infrequently and contained little or no predictive power. Relations in Freebase can be represented by a tree, with the first word being the root and the following levels separated by a period (see column $t[1]$ of Table 4.4). Triples containing relations with the following roots, which represent Freebase metadata, were removed: “user”, “base”, and “freebase”. This reduced the number of relations by 70.1% while only reducing the number of triples by 4.3% and entities by 1.9%.

Additionally, entities were filtered out if they were not present in both the head and tail at least once in the dataset, reducing the total triples by another 4.3%, relations by 1.1%, and entities by 12.1%. By filtering this way we can see that the filter on relations drastically reduces the number of relations while minimally impacting the number of entities and triples, while the filter on entities had minimal impact on the number of triples and relations. The total reduction of this step is displayed in Table 4.1.

4.1.5 Threshold Filter

Since the occurrence of entities in Freebase is not evenly distributed, the frequency of entity appearance in the training data was analyzed to determine a natural threshold

Algorithm 6: The fine filter removes metadata-related relations and ensures that triples only contain entities which are present on both sides of a triple at some point in the dataset.

```

1 function CoarseToFine ( $C$ ) ;
   Input : A coarse-filtered dataset  $C$ 
   Output: A fine-filtered dataset  $F$ 
2 foreach triple  $t \in C$  do
3   | Split( $t$ , “\t”) ;
4   |  $L \leftarrow t[0]$  ; // add all left entities to a set
5   |  $R \leftarrow t[2]$  ; // add all right entities to a set
6 end
7 foreach triple  $t \in C$  do
8   | Split( $t$ , “\t”) ;
9   | if not  $t[1].\text{StartsWith}(\text{'user' or 'base' or 'freebase'})$  and  $L$  contains
   |  $t[0], t[2]$  and  $R$  contains  $t[0], t[2]$  then
10  | |  $t[1].\text{ReplaceAll}(\text{'(.*\.\.)*'}, \text{''})$  ; // shorten compound relations
11  | |  $F \leftarrow t[0] + \text{'\t'} + t[1] + \text{'\t'} + t[2] + \text{'\n'}$  ;
12  | end
13 end
15 return  $F$ 

```

Table 4.5: The same triples as Table 4.4 after running the Fine filter. 3 triples have been deleted and 1 modified.

$t[0]$	$t[1]$	$t[2]$
m.01067m9v	award.award_nomination.nominated_for	m.0zdzxm
m.01068f6j	film.performance.actor	m.0bx_v8d
m.01068pg	music.recording.releases	m.0314xwk
m.017dcd	tv.regular_tv_appearance.actor	m.06v8s0

Figure 4.3: Entity frequency relationship in Freebase; entities outside the thresholds (red lines) are removed.

for removing underrepresented and overrepresented entities, motivated by the desire to remove low quality entities. This method removes a significant amount of entities and triples without resorting to random sampling. Figure 4.3 shows a log-log graph for the frequency at which an entity appears and the number of entities at this frequency. Thresholds were chosen manually to isolate the linear portion of the curve. The lower threshold removes entities occurring less than 35 times in the full dataset, representing about 85% of all entities. The higher threshold removes 850 entities which together occur a total of 157M times.

4.2 Embedding Subsystem

Although not the only way to extract information from a knowledge graph, embedding one in a vector space has several advantages over more direct approaches (such as annotated logical forms). One of these advantages is not having to manually craft the queries, which can be very time consuming, requires expert knowledge, and operates on a limited domain [76]. The embeddings also work naturally with dialog systems, due to their similarity to word embeddings, which are now a common enhancement to dialog systems.

Figure 4.4: Overview of the embedding subsystem. The entire process is repeated for each translation embedding method evaluated.

4.2.1 Translation Embedding

In translation embedding, entities are represented by n -dimensional vectors and relations are represented as translations between entity vectors. This is accomplished with an implementation of TransE (detailed in Section 2.2.1) and TransH, which output the final entity vectors after training on the dataset produced in Section 4.1. The implementations for these algorithms are taken from Lin et al. [36] using their FastTransX repository⁵. Implementations of TransG, TransR, and others from Section 3.2.1 were omitted due to implementation time and the training cost (2 weeks per embedding algorithm). TransE was selected because of its role as a baseline for the rest of the translation embedding methods, and TransH was selected to determine if a method which outperformed TransE on knowledge classification tasks will produce better word embeddings. Vector size is kept at 50 dimensions to speed up training for TransE and TransH as well as the dialog models, despite lower vector sizes possibly affecting the quality of the embeddings (roughly 50% drop in accuracy when reducing dimensionality from 600 to 50 for Word2Vec [3]). Since embedding sizes are kept at the same dimension across all models and the dialog systems are not evaluated on response accuracy, this was seen as a reasonable compromise.

4.2.2 Embedding Conversion

Entities in Freebase are represented by a unique identifier (e.g., “m.0kpv11”) and can be represented by their English label (e.g., “Musical Recording”) which may or

⁵<https://github.com/thunlp/Fast-TransX>

may not be unique. This means that the entity embeddings calculated in Section 4.2.1 could contain embeddings represented by multiple words and could have the same word representation as another entity. Word vectors, on the other hand, consist of a vocabulary of unique unigrams (e.g., “Musical” and “Recording”) with their associated vector. Due to the difference in representation, transforming entity vectors to word vectors directly requires answering the following questions:

1. Can entity vectors with labels consisting of more than one word be split into components?
2. How should entities with non-unique labels be disambiguated?

The transformation of entity vectors to word vectors can be modeled with the function, $f: \mathbb{R}^{d \times n} \rightarrow \mathbb{R}^{d \times k}$, transforming a matrix E representing a set of n entity vectors $e \in E = (e_1, e_2, \dots, e_d)$ to a matrix V representing a set of k word vectors $v \in V = (v_1, v_2, \dots, v_d)$. Both v and e have an associated label $w_1, w_2, \dots, w_j$ consisting of j English words, and $j = 1$ for each v .

Baseline Solutions for Questions 1 and 2

The baseline entity to word transformation function is to answer both these questions with the simplest solution. For question 1 above, multi-word labels are ignored and the unigram entities are matched to the vocabulary from Section 4.3.1. For question 2, entities are disambiguated arbitrarily. Using this method, 30k entities are matched to a possible 50k words in the dialog corpus vocabulary.

$$v = \begin{cases} e & \text{if } j = 1 \\ \emptyset & \text{if } j > 1 \end{cases} \quad (4.1)$$

Expanding the Solution to Question 1

Socher et al. [44] represent entity vectors as an average of its word vectors (e.g., $e_{homosapiens} = 0.5(v_{homo} + v_{sapiens})$). This could potentially be extended by assuming that $(e_{w1}, e_{w2}, \dots, e_{wj})$ are linearly independent, and the baseline solution can be extended to include multi-word labels by representing entities as a linear combination of word vectors:

$$e = \beta_1 v_1 + \beta_2 v_2 + \dots + \beta_j v_j \quad (4.2)$$

Figure 4.5: Overview of the dialog subsystem. The Training-Response Generation loop is repeated for each embedding method evaluated.

where $\beta_1, \beta_2, \dots, \beta_j$ are the scalar coefficients of the j word vectors. This method has not been implemented and is left for future work.

Expanding the Solution to Question 2

There are several possible methods for improving the solution to question 2. One method to disambiguate entities is to select the one which appears more frequently in the knowledge base. This method is not necessarily an improvement over selecting arbitrary entities, as some entities (such as music albums) appear very frequently due to amount of the information associated with them, but may not be referenced at all in the dialog corpus. This method has been implemented and selected over the baseline.

4.3 Dialog Subsystem

The dialog system is an adaptation of OpenNMT⁶ [77], an open-source neural machine translation system built on top of Torch⁷. OpenNMT contains an implementation of a sequence-to-sequence with attention model and many different configuration options. Each step in this section is performed using a script from OpenNMT and configured using either defaults or adjustments found to be an improvement in training speed or quality of responses, after preliminary evaluation. The subsections below detail the five scripts used from OpenNMT and the reasoning for the chosen configuration options.

Although OpenNMT contains a good implementation of a sequence-to-sequence model, some features listed in the related works are missing. Notably, the model does not track the dialog’s state, and would need to be modified to train on multi-turn dialogs (3 or more) in order to better simulate longer conversations (see Section 3.3.1).

4.3.1 Dialog Corpus

The selected dialog corpus is a subset of the OpenSubtitles⁸ corpus initially created by Tiedemann [72], and trimmed to only question/answer pairs⁹. This sample of OpenSubtitles has 14M sentences, from the 338M in the original dataset, and has the following properties:

- Source sentence ends with a question mark.
- Target sentence does not end with a question mark.
- Target sentence is uttered within 20 seconds of the source sentence.

However, target sentences are not guaranteed to be a direct answer to the source questions, which could be a factor in the performance of a dialog system trained on it. Target sentences are sometimes uttered by the same speaker, after a change of scene, or after a change in topic.

Preliminary testing with this sample dataset showed that generated target sentences often contained unknown (out of vocabulary) tokens, with a single unknown

⁶<http://opennmt.net/>

⁷<http://torch.ch/>

⁸<https://www.opensubtitles.org/>

⁹<http://forum.opennmt.net/t/english-chatbot-model-with-opennmt/184>

Table 4.6: Comparing two tokenization methods applied to the dialog corpus: aggressive (A) vs. conservative (C).

Property	A (source)	A (target)	C (source)	C (target)
Total tokens	276,452	330,877	338,336	428,524
Unknown tokens	0.9%	0.6%	1.1%	0.8%
Average sentence length	6.0	7.3	6.0	7.3
Sentences $\leq$ 20 tokens	98%	96%	98%	97%

token being a common response to many questions. For example, given the question “What is your name?” the dialog system would respond with “<unk>” (the unknown token). In order to reduce the number of unknown tokens in the dataset, all sentences containing non-printable ASCII characters were removed. Additional preprocessing is performed later using scripts described in Sections 4.3.2 and 4.3.3. After running these scripts the total number of unknown tokens is reduced from 4% to 1%, and 13.3M sentences remain in the training set, with 10K sentences held out for validation.

4.3.2 Tokenization

This script tokenizes the dialog corpus created in Section 4.3.1, converting words to a simpler form and reducing the total number of unique words in the corpus. Tokens are created by scanning for sequences of letters or numbers, and splitting on anything else. The “conservative” tokenization mode allows certain characters such as ‘-’ to be present in the middle of a sequence (for example “e-mail”, “good-bye”, “9-1-1”). Other characters, such as ‘,’ are allowed to separate only sequences of numbers (for example “8,000”). This method lets the tokenization capture compound words but creates unnecessary tokens when a hyphen is used to emulate natural speech patterns such as a stutter or a double-take. For example, the vocabulary contains 106 variations of the word “what” including: “w-what”, “w-w-what”, “wh-what”, “wha-what”, “what-what”, “what-”, and “what–what”. The implementation uses a more aggressive tokenization, which does not allow these separators. This removes 35 variants of “what”, with some of the remainder being legitimate words (e.g., “whatsoever”) and the rest being words which were not properly spaced in the

dialog corpus, giving tokens such as: “butwhat”, and “whathappened”. Table 4.6 displays the total token count, number of unknown tokens, and additional sentence information for both tokenization methods.

All tokens are converted to lower case, and upper case letters are preserved using word features. For example, “Hello” becomes “hello|C” to capture first letter capitalization, ‘L’ represents lowercase words, ‘U’ for uppercase, ‘M’ for medial capitals, and ‘N’ for no case (punctuation and symbols).

4.3.3 Preprocessing

In an effort to improve dialog quality, speed up training of the dialog model, and because sentences up to 20 tokens make up 96-98% of the dataset (Table 4.6), sentences over 20 tokens are discarded. The vocabulary consists of the top 50k most frequent tokens of the 330k total, and 4 special tokens. Sentences are shuffled and sorted into batches with other sentences of the same size, removing excess padding and speeding up training.

4.3.4 Embedding

The entities and associated vectors from Section 4.2.2 are first converted to GloVe format: lines consisting of an entity name followed by its vector components, each separated by a space. This file is used as input to the embedding script, which converts it to Torch serialized tensors corresponding to the vocabulary of the dialog corpus. Out of 50k possible words in the vocabulary, 30.7k embeddings have matches to knowledge graph embeddings, with the remaining 20k randomly initialized. An embedding layer is attached to both the encoder and decoder.

4.3.5 Training

The model is configured with the following properties: 2 layers, 256 size LSTM with a global attention model (see Section 3.3.2), and trained for 13 epochs. The size of the model (2 layers, 256 hidden nodes) was chosen because increasing the size up to 3 layers and up to 2048 hidden nodes did not significantly improve the quality of responses, but increased training time drastically. The hidden nodes are comprised of LSTM cells due to their performance in the related work, and the model was found to converge within 13 epochs across many different configurations.

4.3.6 Response Generation

“Chatting” with the system is simulated by using the model to translate a source question by predicting the next 20 tokens (or until an end-of-sentence tag $</s>$ is generated) using beam search. A disadvantage of this implementation is that it will always output the same response to any particular question. Section 3.3.1 presents many other ways of improving the quality and diversity of responses, with possibilities for future improvements discussed in Section 6.2.

A beam of size K generates N best responses from the beam search decoder on a trained model, where $K \geq N$ (See Section 2.5.2). Each token in the tree has a score of S representing its suitability as a response. The search tree is post-processed to increase response diversity, adapted from Li et al. [74]. Every token at level l in the search tree is given a new score S' :

$$S'_l = S_l - (p_{l-1} + \alpha p_l) \quad (4.3)$$

where p_l is the rank of the token with respect to its siblings and p_{l-1} is the penalty of its parent. In addition, each token receives a penalty βf where f is the number of times the word has appeared so far in the traversal. Finally, a list of answers is generated by following a path from the root to each end-of-sentence tag, and responses which appear a number of times above a minimum threshold *min* across all questions are discarded (see Algorithm 7).

Algorithm 7: Generating response diversity through penalties on lower ranked siblings and word frequency, and using a response frequency threshold.

```

1 function PostProcessBeamSearchTree ( $Q, \alpha, \beta, min$ ) ;
   Input : A set of questions  $Q$  with their own beam search tree  $B_q$  for the top
            $N_q$  responses, a sibling penalty  $alpha$ , a word frequency penalty  $\beta$ ,
           and a minimum frequency threshold  $min$ .

   Output: Top 10 responses for each question in  $Q$ 
2 Run BFS on  $B_q$  with ordering determined by token score ;
3 foreach  $Question\ q \in Q$  do
   | // Starting with the root, advance through BFS( $B_q$ ) ordering  $i$ .
4   | while  $i.HasNext()$  do
5   |   |  $v \leftarrow i.Next()$  ;
6   |   |  $r \leftarrow 0$  ;
   |   | //  $freq$  keeps track of frequencies for unique tokens or
   |   | responses.
7   |   |  $freq \leftarrow \emptyset$  foreach  $c \in v.Children()$  do
8   |   |   |  $c.p \leftarrow v.p + \alpha r$  ;
9   |   |   |  $r \leftarrow r + 1$  ;
10  |   |   |  $c.p \leftarrow c.p + \beta + freq[c]++$  ;
11  |   | end
12  | end
13  | Get list  $L_q$  of responses by tracing a path from the root to each end tag ;
14  | foreach  $r \in L_q$  do
15  |   |  $freq[r]++$  ;
16  | end
17 end
18 Sort lists by score - penalty ;
19 foreach  $Question\ q \in Q$  do
20 |   Remove  $r \in L_q$  if  $freq[r] > min$  times ;
21 |   Remove responses after the top 10 responses ;
22 end
23 return Lists of top 10 responses for each question.

```

Chapter 5

Evaluation

Two knowledge graph embedding methods (TransE, TransH) and two word embedding methods (GloVe, Word2Vec) were chosen for evaluation. These particular methods were chosen primarily because of implementation and training time limitations, while still meeting the objective of contrasting word and knowledge graph embeddings. Each embedding method is applied to its own dialog system with all other parameters fixed (see Section 4.3). Embeddings are also evaluated as both flexible (allowed to change during training), and fixed. Fixed embeddings remain constant during training if the word is present in the vocabulary, otherwise they are randomly initialized and allowed to change during training. TransE and TransH embeddings were created using the method described in Section 4.2 while GloVe and Word2Vec embeddings were downloaded pretrained from official sources¹².

A test script was created using all 63 questions asked in the “OpenSubtitles experiments” section in [2]. The questions cover 7 topics of conversation: basic, simple Q&A, general knowledge, philosophy, morality, opinions, and work. Each model generated a top 10 list of answers to each question using the beam search and post-processing methods described in Section 4.3.6. For each list, the number of plausible responses were counted manually, with the criteria adjusted for each question. A plausible response in this context is defined as the following:

- Naturally follows the input in conversation. In the case that the input is a question, the response answers the question, correctly or incorrectly.
- Uses the right pronoun when referencing the input.

¹<https://nlp.stanford.edu/projects/glove/>

²<https://code.google.com/archive/p/word2vec/>

- Specific enough that it cannot be used to answer a large number of questions from the test script, unless specified.
- Contains specific keywords or constraints based on the input in the form of a response description.

Response descriptions were created by analyzing common responses among the models and generalizing them to a short descriptions. In order to ensure consistent response selection, the additional descriptions were sent out as a survey along with the above definition for plausible answers. Participants were chosen who had no prior knowledge of the objective and out of the 5 surveys sent, 4 responded. Participants were asked to answer yes or no to the following question for each input-description pair: “Could the response description given below be used to select a similar number of sentences from an identical set of sentences if given to a large number of people?”. The inputs and their descriptions can be found in Tables A.2 and A.3. The survey results can be found in Tables A.8 and A.9. Responses roughly indicated the degree of difficulty of the input questions, where inputs with high disagreement also tend to have low average scores in the evaluation.

The number of plausible responses generated by each model for each question can be found in Appendix A. Section 5.1 compares the embeddings used for each experiment, as well as the effects of training on variable embeddings. Section 5.2.1 covers the results of evaluating the dialog system using plausible answers.

5.1 Visualizing the Entity Vectors

To get a better understanding of the differences between traditional word embeddings and knowledge graph embeddings, a sample from each is projected into 2D and analyzed using TensorBoard³. In order to evaluate word and knowledge graph embeddings equivalently, the samples consist of the top 4k most common English words. The embeddings were projected using t-SNE with a perplexity of 14 and learning rate of 10 for 4000 iterations (Figures 5.1 and 5.2). These parameters were determined by

³https://www.tensorflow.org/get_started/embedding_viz

Figure 5.1: t-SNE projection of embeddings for the top 4k most common English words.

Figure 5.2: t-SNE projection of embeddings for the top 4k most common English words.

Figure 5.3: Highlight of the word “man” and its 20 closest neighbors in the original space.

Figure 5.4: Highlight of the word “man” and its 20 closest neighbors in the original space.

the size of the data and some fine tuning while examining the shape of the projection. In the case of knowledge graph embeddings, only 3k words are matched since the graph itself does not contain many common stop words.

To help illustrate the differences between the different embedding methods, an example word “man” is chosen with its nearest neighbors displayed on the 2D projections (Figures 5.3 and 5.4). The neighbors of “man” differ drastically between embedding methods due to the difference in the relation to other words during training. For example, traditional word embedding methods will link “man” to other words that fill a similar role in a sentence (in the Word2Vec skip-gram model), or is used in a similar context (in the GloVe model). On the other hand, word embeddings derived from knowledge graph embeddings are influenced by relations present in the graph itself, and will be related to other words that share similar relations.

Using the previous example “man”, GloVe embeddings put the word close to “woman”, “boy”, and “old” (Table 5.1). TransE has it close to “ethics”, “legislation”, and “security”, possibly making its definition closer to “mankind” rather than a specific male. Further investigation (using Freebase Easy [78]) revealed two interesting properties: most of the nearest neighbors of “man” are the tail end of a “[book] subject” relation, and “writer” and “author” are the most represented profession in Freebase (55k and 17k instances respectively). The relations between these entities and their quantity could be the reason for their close association with “man”.

Another example word, “prison”, is close to related terms such as “jailer” and “criminal” in word embeddings but close to other institutions (“university”, “bank”) in knowledge graph embeddings. A third example “yellow” has similar nearest neighbors across all embedding methods, being close to other colors and commonly yellow objects. This is the case when the relation between entities in the knowledge graph is closely tied to its use in a sentence.

5.1.1 Distribution Analysis

To further understand how the embeddings are affected during training in the variable embedding models, k nearest neighbor lists are generated for each word in each set of embeddings before and after training. Common pairs, defined as the intersection of the before and after lists, are counted and their frequency normalized by the total words in the intersection of the two embedding sets. These frequencies are displayed

Table 5.1: Top 10 nearest neighbors of the word “man” for each model.

GloVe-Fixed	GloVe-Variable	Word2Vec-Fixed	Word2Vec-Variable
woman	woman	woman	boy
boy	boy	person	fellow
old	guy	girl	mate
who	girl	child	guy
another	fellow	time	son
girl	mate	character	woman
friend	lady	race	girl
father	bull	country	buddy
him	gentleman	style	outside
himself	brother	boy	friend

TransE-Fixed	TransE-Variable	TransH-Fixed	TransH-Variable
ethics	brother	rhetoric	woman
legislation	guy	export	boy
security	woman	head	guy
liberty	son	empire	girl
head	boy	failure	lady
logic	lady	discrimination	son
consciousness	officer	flower	player
politics	girl	courage	gentleman
civilization	fool	credit	nut
religion	wolf	responsibility	brother

Figure 5.5: Number of common pairs between the top 20 nearest neighbor lists for each word before and after the embeddings are adjusted by the dialog system, with frequency in log scale.

with a log scale in Figure 5.5 for all 4 variable embedding models at $k = 20$. Figures 5.6 and 5.7 show the changes to the distributions as k increases using a linear scale for the frequencies.

The shape of the distributions help indicate that embeddings are greatly altered during training, as the majority (frequency $> 50\%$ when $k = 20$) of words will have 0 words in common with the lists after training for every model. As k increases the distributions appear to converge to 3 different gamma distributions. Word2Vec is the least skewed, with the lowest shape parameter value. The GloVe distribution is the most skewed, with the highest shape parameter value, and is the closest of the 4 to a normal distribution. From these distributions we can see that GloVe has the most in common with the embeddings after training, meaning they changed the least during training. Both TransE and TransH behaved nearly identically, and Word2Vec appears to change the most after being altered by the dialog system in a variable embedding model.

Figure 5.6: Number of common pairs between the top k nearest neighbor lists for each word before and after the embeddings are adjusted by the dialog system.

(a) $k = 250$ (b) $k = 300$ (c) $k = 350$ (d) $k = 400$

Figure 5.7: Number of common pairs between the top k nearest neighbor lists for each word before and after the embeddings are adjusted by the dialog system.

Figure 5.8: Validation perplexity for each model over 13 training epochs.

5.2 Dialog System Evaluation

Figure 5.8 displays the validation perplexity for each of the 8 models during training. This figure shows a clear split between the variable and fixed group of embedding models, with the former being significantly lower than the latter. Within either group the models are not significantly different from each other. Each model shows a sharp drop in perplexity after epoch 7 or 9 as a result of learning rate decay. This decay starts when either the difference from the last epoch is 0 or lower (as seen in models dropping after epoch 7), or when 9 epochs pass.

5.2.1 Results

The number of plausible responses generated by each model were counted for each question, resulting in a table of 8 variables with 63 observations. A Friedman test was conducted to determine whether any significant difference occurs between the models, and were found to be significantly different ($p = 6.232 \times 10^{-5}$). A Nemenyi post-hoc test was then conducted to determine pairwise difference between models (Table 5.2). All variable embedding models were found to be significantly different than the Word2Vec-Fixed model at differing levels of significance. The GloVe model had the most significant difference ($p < 0.001$), followed by TransH and Word2Vec ($p < 0.01$), and TransE ($p < 0.05$). No significant difference was found between any

Figure 5.9: Distributions for the number of plausible responses per question for each model.

Table 5.2: Pairwise comparisons for each model using Nemenyi multiple comparison test.

	GloVe	GloVe-Fixed	Word2Vec	Word2Vec-Fixed	TransE	TransE-Fixed	TransH
GloVe-Fixed	0.59482	-	-	-	-	-	-
Word2Vec	0.99557	0.95886	-	-	-	-	-
Word2Vec-Fixed	0.00047***	0.21511	0.00933**	-	-	-	-
TransE	0.96580	0.99406	0.99999	0.02659*	-	-	-
TransE-Fixed	0.75039	1.00000	0.98981	0.12507	0.99940	-	-
TransH	0.99986	0.85755	0.99999	0.00292**	0.99865	0.94208	-
TransH-Fixed	0.05756	0.94666	0.32956	0.89570	0.53169	0.86574	0.17631

other pair of models.

Chapter 6

Conclusion

The two main goals of this thesis were: to use human-made facts and real-world knowledge to improve the quality of responses generated by a neural dialog model and to evaluate these models as objectively as possible. To achieve these goals and form a complete end-to-end dialog system, a number of questions were addressed and various subsystems implemented. Section 6.1 presents a summary of these subsystems and a summary of the experiment results, and Section 6.2 discusses possibilities for future work.

6.1 Summary of Results

The sections below refer to the contributions listed in Section 1.4.

6.1.1 Knowledge Graph Pipeline

The knowledge graph pipeline was created in order preprocess Freebase to both fall within hardware constraints, and lose a minimum amount of information when reducing the size (Section 4.1). The pipeline consists of multiple independent filters which can be added, removed, or modified in order to generate a dataset of the required size. These filters and their algorithms were detailed as well as the effects they had on the knowledge graph.

6.1.2 Embedding Conversion

A simple baseline method was to convert entity embeddings to word embeddings, involving two steps: entity disambiguation and dealing with multi word labels (Section

4.2.2). This method chooses the most frequent entity in case of duplicate labels and ignores entities with multi word labels (only using entities consisting of a single word). When applied to the 50k word vocabulary, these converted entities matched 60% of words (where GloVe matched 94%).

6.1.3 Beam Search Postprocessing

Modifying beam search was found to be the biggest factor in generating interesting and different responses from the evaluated models (Section 4.3.6). A postprocessing method was proposed to attempt to extract the top 10 best responses for each question generated by a beam search of size 175 (with a best list size of 175). The method consists of 3 sub-methods with their own parameters to adjust the weight of the penalties of each sub-method. The first is a penalty applied to lower ranked siblings, the second a penalty applied based on the frequency of the word in the beam search tree, and the third is a list crop threshold and resorting the remainder by the frequency each answer occurs in the testing script. Using this method yielded much better responses than vanilla beam search.

6.1.4 Plausible Response Evaluation

An evaluation method was proposed in order to address problems with current evaluation methods for generative dialog systems. The method uses a set description for what constitutes a plausible answer for each question, which is used to select the number of plausible responses in the top 10 for each model (Chapter 5). This method was used to compare the 8 evaluated models: GloVe, Word2Vec, TransE, and TransH, trained with both variable and fixed embeddings. All of the models trained with variable embeddings performed significantly better than Word2Vec with fixed embeddings. The models trained with knowledge graph embeddings did not perform significantly better or worse than the baseline models.

6.2 Future Work

There are multiple areas which could use some improvement and require additional research which falls outside the scope of this thesis. Three important areas for future work are described below.

6.2.1 Embedding Conversion

The embedding conversion method detailed in Section 4.2.2 converts entity embeddings to word embeddings. The entity disambiguation step (Question 2) can be extended by applying entity disambiguation techniques to the dialog corpus, and tag homographs such that they appear as separate words in the vocabulary. This method increases the number of unique words in the corpus, which increases the number of unknown tokens due to the static size of the vocabulary. Despite this, it should have the most potential for improvement over the baseline, depending on the entity disambiguation technique used and the net inflation of tokens. An alternative to this method is using entity disambiguation to select the entity which appears the most frequently in the dialog corpus rather than the knowledge graph, leaving the total word count of the dialog corpus unchanged.

Handling multi-word labels (Question 1) can be achieved in a similar way, with an entity disambiguation system able to detect entities, such as “Major League Baseball”, within a sentence despite not necessarily being completely spelled out. Another solution would be to represent these entities as a linear combination of the entities contained in the label, in this example “Major”, “League”, and “Baseball” would be embeddings and the system would solve for their coefficients.

6.2.2 Beam Search Postprocessing

The current beam search postprocessing method is dependent on the questions asked in the test script and could be altered to apply most of the penalties during run time, similar to the method by Li et al. [74], which could lower the memory requirements but may impact the answers generated by the model. This could only be directly applied to the first component, which penalizes the lower ranking siblings. Since the second component applies a penalty based on the number of times a word appears in the entire beam search tree, there are a few different strategies to move this penalty to run time. The first is an increasing penalty each subsequent time a word is encountered in the search, and the second is running the beam search multiple times while applying penalties equal to the number of times each word has been used in the previous iteration. The last component sorts the top 30 answers based on the number of times this answer has occurred in the test script, and returns the top 10 answers. A similar strategy could be adopted for this component but would still require many different

questions in order to build a dataset of generic answers. These answers could then be pruned during run time.

6.2.3 Plausible Response Evaluation

The plausible response evaluation method was intended to be used with a variety of human evaluators to achieve consensus by providing clear definitions for the type of answers required for each question in order to compare different models. Selecting these answers however is a very time consuming process, since a test script with 63 question and 8 models require participants to look through 5040 answers. This number can be reduced by creating a tool which merges duplicate answers across models and also across questions with the same plausibility description. This tool could speed up the evaluation process and also eliminate errors in counting plausible answers.

List of References

- [1] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, “Building end-to-end dialogue systems using generative hierarchical neural network models,” *Computing Research Repository*, vol. abs/1507.04808, 2015.
- [2] O. Vinyals and Q. Le, “A neural conversational model,” *Computing Research Repository*, vol. abs/1506.05869, 2015.
- [3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *Computing Research Repository*, vol. abs/1301.3781, 2013.
- [4] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, vol. 14, pp. 1532–1543, Association for Computational Linguistics, 2014.
- [5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: a collaboratively created graph database for structuring human knowledge,” in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD 2008*, ACM Press, 2008.
- [6] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in *Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS’13, (USA)*, pp. 2787–2795, Curran Associates Inc., 2013.
- [7] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes,” 2014.
- [8] X. Li, G. Tur, D. Hakkani-Tür, and Q. Li, “Personal knowledge graph population from user utterances in conversational understanding,” in *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pp. 224–229, IEEE, 2014.
- [9] A. M. Turing, “Computing machinery and intelligence,” *Mind*, vol. LIX, no. 236, pp. 433–460, 1950.
- [10] A. Pinar Saygin, I. Cicekli, and V. Akman, “Turing test: 50 years later,” *Minds Mach.*, vol. 10, pp. 463–518, Nov. 2000.

- [11] J. Weizenbaum, “ELIZA—a computer program for the study of natural language communication between man and machine,” *Communications of the ACM*, vol. 9, pp. 36–45, Jan. 1966.
- [12] K. M. Colby, *Artificial Paranoia: A Computer Simulation of Paranoid Processes*. New York, NY, USA: Elsevier Science Inc., 1975.
- [13] K. M. Colby and F. D. Hilf, “Can expert judges, using transcripts of teletyped psychiatric interviews, distinguish human paranoid patients from a computer simulation of paranoid processes?,” tech. rep., Stanford, CA, USA, 1972.
- [14] V. Bastin and D. Cordier, “Methods and tricks used in an attempt to pass the turning test,” in *Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning - NeMLaP3/CoNLL 1998*, NeMLaP3/CoNLL 1998, (Stroudsburg, PA, USA), pp. 275–277, Association for Computational Linguistics (ACL), 1998.
- [15] L. Floridi, M. Taddeo, and M. Turilli, “Turing’s imitation game: Still an impossible challenge for all machines and some judges—an evaluation of the 2008 loebner contest,” *Minds Mach.*, vol. 19, pp. 145–150, Feb. 2009.
- [16] M. A. Nielsen, *Neural networks and deep learning*. Determination Press USA, 2015.
- [17] A. Nkland, “Direct feedback alignment provides learning in deep neural networks,” *Computing Research Repository*, 2016.
- [18] Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, and Z. Lin, “Towards biologically plausible deep learning,” *Computing Research Repository*, 2015.
- [19] B. Scellier and Y. Bengio, “Equilibrium propagation: Bridging the gap between energy-based models and backpropagation,” *Computing Research Repository*, 2016.
- [20] B. A. Garro and R. A. Vázquez, “Designing artificial neural networks using particle swarm optimization algorithms,” *Computational Intelligence and Neuroscience*, vol. 2015, pp. 1–20, 2015.
- [21] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.
- [22] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *Computing Research Repository*, 2014.
- [23] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Computing Research Repository*, vol. abs/1409.3215, 2014.
- [24] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

- [25] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, ACM, ACM Press, 2008.
- [26] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *Computing Research Repository*, 2016.
- [27] O. Levy, Y. Goldberg, and I. Dagan, “Improving distributional similarity with lessons learned from word embeddings,” *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.
- [28] R. Guns, “Tracing the origins of the semantic web,” *Journal of the American Society for Information Science and Technology*, vol. 64, no. 10, pp. 2173–2181, 2013.
- [29] M. Färber, B. Ell, C. Menne, and A. Rettinger, “A comparative survey of dbpedia, freebase, opencyc, wikidata, and yago,” *Semantic Web Journal*, July, 2015.
- [30] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, “DBpedia: A nucleus for a web of open data,” in *The Semantic Web*, pp. 722–735, Springer Berlin Heidelberg, 2007.
- [31] D. Vrandečić and M. Krötzsch, “Wikidata: A free collaborative knowledgebase,” *Commun. ACM*, vol. 57, pp. 78–85, Sept. 2014.
- [32] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: A core of semantic knowledge,” in *Proceedings of the 16th International Conference on World Wide Web, WWW ’07*, (New York, NY, USA), pp. 697–706, ACM, 2007.
- [33] C. Fellbaum, *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.
- [34] A. Gangemi, *Ontology Design Patterns for Semantic Web Content*, pp. 262–276. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.
- [35] X. Glorot, A. Bordes, J. Weston, and Y. Bengio, “A semantic matching energy function for learning with multi-relational data,” *Computing Research Repository*, vol. abs/1301.3485, 2013.
- [36] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, “Learning entity and relation embeddings for knowledge graph completion,” in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI’15*, pp. 2181–2187, AAAI Press, 2015.
- [37] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, “Modeling relation paths for representation learning of knowledge bases,” *Computing Research Repository*, 2015.
- [38] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, “Knowledge graph embedding via dynamic mapping matrix,” in *ACL*, 2015.
- [39] M. Fan, Q. Zhou, E. Chang, and T. F. Zheng, “Transition-based knowledge graph embedding with relational mapping properties,” in *PACLIC*, 2014.

- [40] H. Xiao, M. Huang, and X. Zhu, “Transg : A generative model for knowledge graph embedding,” in *ACL*, 2016.
- [41] H. Xiao, M. Huang, and X. Zhu, “From one point to a manifold: Knowledge graph embedding for precise link prediction,” *Computing Research Repository*, 2015.
- [42] S. He, K. Liu, G. Ji, and J. Zhao, “Learning to represent knowledge graphs with gaussian embedding,” in *CIKM*, 2015.
- [43] A. Celikyilmaz, D. Hakkani-Tr, P. Pasupat, and R. Sarikaya, “Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems,” *AAAI - Association for the Advancement of Artificial Intelligence*, January 2015.
- [44] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, “Reasoning with neural tensor networks for knowledge base completion,” in *Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS’13, (USA)*, pp. 926–934, Curran Associates Inc., 2013.
- [45] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph and text jointly embedding,” in *The 2014 Conference on Empirical Methods on Natural Language Processing*, ACL Association for Computational Linguistics, October 2014.
- [46] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, “Representing text for joint embedding of text and knowledge bases,” *ACL Association for Computational Linguistics*, September 2015.
- [47] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, “Representation learning of knowledge graphs with entity descriptions.,” in *AAAI*, pp. 2659–2665, 2016.
- [48] H. Xiao, M. Huang, and X. Zhu, “Ssp: Semantic space projection for knowledge graph embedding with text descriptions,” *Computing Research Repository*, 2016.
- [49] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, “Collaborative knowledge base embedding for recommender systems,” in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’16, (New York, NY, USA)*, pp. 353–362, ACM, 2016.
- [50] H. Huang, L. Heck, and H. Ji, “Leveraging deep neural networks and knowledge graphs for entity disambiguation,” *Computing Research Repository*, 2015.
- [51] L. Heck and H. Huang, “Deep learning of knowledge graph embeddings for semantic parsing of twitter dialogs,” in *The 2nd IEEE Global Conference on Signal and Information Processing (DRAFT)*, IEEE Institute of Electrical and Electronics Engineers, December 2014.
- [52] L. Shang, Z. Lu, and H. Li, “Neural responding machine for short-text conversation,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Association for Computational Linguistics (ACL), 2015.

- [53] M. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *Computing Research Repository*, vol. abs/1508.04025, 2015.
- [54] K. Yao, G. Zweig, and B. Peng, “Attention with intention for a neural network conversation model,” *Computing Research Repository*, vol. abs/1510.08565, 2015.
- [55] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, “Deep reinforcement learning for dialogue generation,” *Computing Research Repository*, vol. abs/1606.01541, 2016.
- [56] J. Li, M. Galley, C. Brockett, G. Spithourakis, J. Gao, and B. Dolan, “A persona-based neural conversation model,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics (ACL), 2016.
- [57] A. Kannan, P. Young, V. Ramavajjala, K. Kurach, S. Ravi, T. Kaufmann, A. Tomkins, B. Miklos, G. Corrado, L. Lukacs, and M. Ganea, “Smart reply,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 2016*, KDD ’16, (New York, NY, USA), pp. 955–964, Association for Computing Machinery (ACM), 2016.
- [58] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan, “A neural network approach to context-sensitive generation of conversational responses,” in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics (ACL), 2015.
- [59] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *Computing Research Repository*, 2014.
- [60] A. Ritter, C. Cherry, and W. B. Dolan, “Data-driven response generation in social media,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP ’11, (Stroudsburg, PA, USA), pp. 583–593, Association for Computational Linguistics, 2011.
- [61] J. Gu, Z. Lu, H. Li, and V. O. Li, “Incorporating copying mechanism in sequence-to-sequence learning,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics (ACL), 2016.
- [62] J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky, “Adversarial learning for neural dialogue generation,” *Computing Research Repository*, vol. abs/1701.06547, 2017.
- [63] I. V. Serban, R. Lowe, L. Charlin, and J. Pineau, “A survey of available corpora for building data-driven dialogue systems,” *Computing Research Repository*, vol. abs/1512.05742, 2015.
- [64] A. Ritter, C. Cherry, and B. Dolan, “Unsupervised modeling of twitter conversations,” in *Human Language Technologies: The 2010 Annual Conference of*

- the North American Chapter of the Association for Computational Linguistics*, HLT '10, (Stroudsburg, PA, USA), pp. 172–180, Association for Computational Linguistics, 2010.
- [65] R. Lowe, N. Pow, I. Serban, and J. Pineau, “The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems,” in *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Association for Computational Linguistics (ACL), 2015.
 - [66] R. Kadlec, M. Schmid, and J. Kleindienst, “Improved deep learning baselines for ubuntu corpus dialogs,” *Computing Research Repository*, vol. abs/1510.03753, 2015.
 - [67] R. T. Lowe, N. Pow, I. V. Serban, L. Charlin, C. Liu, and J. Pineau, “Training end-to-end dialogue systems with the ubuntu dialogue corpus,” *D&D*, vol. 8, no. 1, pp. 31–65, 2017.
 - [68] C.-W. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau, “How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation,” *Computing Research Repository*, vol. abs/1603.08023, 2016.
 - [69] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: A method for automatic evaluation of machine translation,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, (Stroudsburg, PA, USA), pp. 311–318, Association for Computational Linguistics, 2002.
 - [70] A. Lavie and A. Agarwal, “Meteor: An automatic metric for mt evaluation with improved correlation with human judgments,” pp. 65–72, 2005.
 - [71] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio, “Generating sentences from a continuous space,” *Computing Research Repository*, vol. abs/1511.06349, 2015.
 - [72] J. Tiedemann, “News from opus-a collection of multilingual parallel corpora with tools and interfaces,” in *Recent advances in natural language processing*, vol. 5, pp. 237–248, 2009.
 - [73] Niel Chah, “Nchah/freebase-triples v1.1.0,” 2017.
 - [74] J. Li, W. Monroe, and D. Jurafsky, “A simple, fast diverse decoding algorithm for neural generation,” *Computing Research Repository*, 2016.
 - [75] Google, “Freebase data dumps.” <https://developers.google.com/freebase/data>, 2017.
 - [76] J. Berant, A. Chou, R. Frostig, and P. Liang, “Semantic parsing on freebase from question-answer pairs,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1533–1544, 2013.

- [77] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, “OpenNMT: Open-Source Toolkit for Neural Machine Translation,” *ArXiv e-prints*, 2017.
- [78] H. Bast, F. Baurle, B. Buchhold, and E. Haußmann, “Easy access to the freebase dataset,” in *Proceedings of the 23rd International Conference on World Wide Web*, pp. 95–98, ACM, 2014.

Appendix A

Additional Material

A.1 Experiment Environment

The main bottlenecks of the experimental environments were the amount of available memory and the power of the GPU. The experimental environment consists of two identical computers with the hardware listed in Table [A.1](#).

A.2 Plausible Response Model

Table A.1: Experiment environment hardware.

CPU	Intel i7-6700k @ 4.0 GHz
GPU	GeForce GTX 1070
GPU Memory	8192 MB
RAM	32 GB

Table A.2: Plausible responses and their descriptions. 1/2

Input	Plausible Response
Hello!	Any greeting: hi, hello, hey, etc.
How are you?	Fine, good, etc. as an adjective.
What's your name?	Any name.
When were you born?	Any static or relative time in the past.
What year were you born?	Any number or a relative year in the past.
Where are you from?	A municipality, state, country, continent, etc.
Are you a man or a woman?	Any gender or both, neither, etc.
Why are we here?	We'll see, we'll find out, etc. or begins with because, to, etc.
Okay, bye!	Okay, bye or "I'll be back" or similar.
See you later.	Agreement/disagreement or future plans.
My name is David, what is my name?	Any name.
My name is John, what is my name?	Any name.
Are you a leader or a follower?	Any profession, person, etc. or both/neither, etc.
Are you a follower or a leader?	Any profession, person, etc. or both/neither, etc.
Who is Skywalker?	Any adjective or relationship to speaker.
Who is Bill Clinton?	Any adjective or relationship to speaker.
Is sky blue or black?	Any color.
Does a cat have a tail?	Yes/no.
Does a cat have a wing?	Yes/no.
Can a cat fly?	Yes/no.
How many legs does a cat have?	Any number, a few, a lot, etc.
How many legs does a spider have?	Any number, a few, a lot, etc.
How many legs does a centipede have?	Any number, a few, a lot, etc.
What is the color of the sky?	Any color.
What is the color of water?	Any color.
What is the color of blood?	Any color.
What is the usual color of a leaf?	Any color or "it depends".
What is the color of a yellow car?	Any color.
How much is two plus two?	Any number, numeric or english (and not in dollars).
How much is ten minus two?	Any number, numeric or english (and not in dollars).
What is the purpose of life?	Any description (begins with "it's") or uncertainty "I don't know", "I'm not sure", etc.

Table A.3: Plausible responses and their descriptions. 2/2

Input	Plausible Response
What is the purpose of living?	Any description (begins with "it's") or uncertainty "I don't know".
What is the purpose of existence?	Any description (begins with "it's") or uncertainty "I don't know".
Where are you now?	A static location or relative location information.
What is the purpose of dying?	Any description (begins with "it's") or contains die, dying, kill, etc.
What is the purpose of being intelligent?	Any description (begins with "it's") or uncertainty "I don't know".
What is the purpose of emotions?	Any description (begins with "it's") or uncertainty "I don't know".
What is moral?	Any description (begins with "it's").
What is immoral?	Any description (begins with "it's").
What is morality?	Any description (begins with "it's").
What is the definition of altruism?	Any description (begins with "it's").
OK... so what is the definition of morality?	Uncertainty "I don't know", "I'm not sure", etc.
Tell me the definition of morality, I am quite upset now!	Yes, no, sorry, etc.
Tell me the definition of morality.	Any description (begins with "it's" or a single descriptive word).
Look, I need help, I need to know more about morality...	Contains a 3rd person or demonstrative pronoun.
Seriously, what is morality?	Contains a 3rd person or demonstrative pronoun.
Why living has anything to do with morality?	"because...", "it does", etc. or "it doesn't", "no reason", etc.
Okay, I need to know how should I behave morally...	Contains a 3rd person or demonstrative pronoun.
Is morality and ethics the same?	Yes/no.
What are the things that I do to be immoral?	Contains a 3rd person or demonstrative pronoun.
Give me some examples of moral actions...	"I have/will" or yes/no, etc. or an example action.
Alright, morality?	Any attempt at continuing the conversation excluding yes/no.
What is integrity?	Contains a 3rd person or demonstrative pronoun.
Be moral!	Yes/no or contains a 3rd person or demonstrative pronoun.
I really like our discussion on morality and ethics...	Any general agreement or disagreement.
What do you like to talk about?	Any topic or nothing, not much, etc.
What do you think about Tesla?	Any description, any gender pronoun.
What do you think about Bill Gates?	Any description, male pronoun.
What do you think about Messi?	Any description, male pronoun.
What do you think about Cleopatra?	Any description, female pronoun.
What do you think about England during the reign of Elizabeth?	Any description, gender neutral or female pronoun.
What is your job?	Any profession.
What do you do?	Any profession.

Table A.4: Number of plausible responses present in the top 10 answers generated by each model. 1/4

Input	GloVe	Glove-Fixed	Word2Vec	Word2Vec-Fixed
Hello!	4	7	3	3
How are you?	6	7	6	9
What's your name?	1	3	0	1
When were you born?	5	5	4	8
What year were you born?	1	1	4	2
Where are you from?	8	5	6	6
Are you a man or a woman?	7	4	7	7
Why are we here?	1	2	3	2
Okay, bye!	4	2	5	4
See you later.	8	7	9	9
My name is David, what is my name?	0	2	0	0
My name is John, what is my name?	1	3	1	0
Are you a leader or a follower?	7	3	8	6
Are you a follower or a leader?	6	4	8	4
Who is Skywalker?	4	7	1	4
Who is Bill Clinton?	6	4	5	5
Is sky blue or black?	8	9	7	0
Does a cat have a tail?	7	7	10	6
Does a cat have a wing?	6	8	10	8
Can a cat fly?	5	5	9	3
How many legs does a cat have?	9	8	7	9
How many legs does a spider have?	9	9	7	9
How many legs does a centipede have?	4	10	5	9
What is the color of the sky?	8	3	7	0
What is the color of water?	9	6	8	0
What is the color of blood?	10	6	9	0
What is the usual color of a leaf?	7	6	7	0
What is the color of a yellow car?	10	8	10	0
How much is two plus two?	10	5	8	2
How much is ten minus two?	10	6	6	1
What is the purpose of life?	8	5	7	3

Table A.5: Number of plausible responses present in the top 10 answers generated by each model. 2/4

Input	GloVe	Glove-Fixed	Word2Vec	Word2Vec-Fixed
What is the purpose of living?	4	5	3	3
What is the purpose of existence?	7	6	4	6
Where are you now?	10	8	8	7
What is the purpose of dying?	6	2	4	0
What is the purpose of being intelligent?	6	7	6	4
What is the purpose of emotions?	4	3	3	4
What is moral?	4	1	3	0
What is immoral?	4	5	3	1
What is morality?	7	1	3	0
What is the definition of altruism?	2	0	4	0
OK... so what is the definition of morality?	7	6	2	2
Tell me the definition of morality, I am quite upset now!	4	3	6	2
Tell me the definition of morality.	6	1	3	0
Look, I need help, I need to know more about morality...	5	6	4	4
Seriously, what is morality?	3	2	3	0
Why living has anything to do with morality?	3	2	4	0
Okay, I need to know how should I behave morally...	7	6	5	4
Is morality and ethics the same?	4	5	9	7
What are the things that I do to be immoral?	2	4	2	5
Give me some examples of moral actions...	5	7	7	4
Alright, morality?	7	1	2	0
What is integrity?	2	3	4	1
Be moral!	4	3	2	4
I really like our discussion on morality and ethics...	2	3	4	4
What do you like to talk about?	5	6	7	6
What do you think about Tesla?	6	6	9	9
What do you think about Bill Gates?	8	8	7	1
What do you think about Messi?	3	7	7	3
What do you think about Cleopatra?	9	10	10	0
What do you think about England during the reign of Elizabeth?	5	2	1	3
What is your job?	6	6	4	0
What do you do?	6	5	5	7

Table A.6: Number of plausible responses present in the top 10 answers generated by each model. 3/4

Input	TransE	TransE-Fixed	TransH	TransH-Fixed
Hello!	3	5	4	2
How are you?	8	8	8	7
What's your name?	0	3	0	2
When were you born?	5	4	5	6
What year were you born?	0	2	2	1
Where are you from?	4	2	5	5
Are you a man or a woman?	8	8	7	6
Why are we here?	2	5	2	3
Okay, bye!	4	4	4	2
See you later.	10	8	9	10
My name is David, what is my name?	0	1	0	0
My name is John, what is my name?	1	0	0	0
Are you a leader or a follower?	9	3	6	4
Are you a follower or a leader?	6	2	3	4
Who is Skywalker?	2	7	1	4
Who is Bill Clinton?	6	7	2	3
Is sky blue or black?	10	10	8	10
Does a cat have a tail?	5	7	9	8
Does a cat have a wing?	5	10	10	8
Can a cat fly?	3	5	8	3
How many legs does a cat have?	8	10	8	10
How many legs does a spider have?	8	10	9	10
How many legs does a centipede have?	9	10	9	10
What is the color of the sky?	7	7	6	6
What is the color of water?	6	7	8	9
What is the color of blood?	6	9	9	6
What is the usual color of a leaf?	8	9	9	0
What is the color of a yellow car?	10	10	7	10
How much is two plus two?	10	5	9	7
How much is ten minus two?	9	4	9	6
What is the purpose of life?	3	7	8	2

Table A.7: Number of plausible responses present in the top 10 answers generated by each model. 4/4

Input	TransE	TransE-Fixed	TransH	TransH-Fixed
What is the purpose of living?	3	8	5	3
What is the purpose of existence?	7	7	7	4
Where are you now?	8	7	7	7
What is the purpose of dying?	5	1	4	1
What is the purpose of being intelligent?	6	4	5	2
What is the purpose of emotions?	5	4	5	6
What is moral?	3	0	5	0
What is immoral?	3	0	2	1
What is morality?	3	2	3	5
What is the definition of altruism?	1	0	4	0
OK... so what is the definition of morality?	3	3	5	3
Tell me the definition of morality, I am quite upset now!	6	2	4	5
Tell me the definition of morality.	2	1	3	2
Look, I need help, I need to know more about morality...	5	3	4	5
Seriously, what is morality?	3	3	4	1
Why living has anything to do with morality?	4	5	4	2
Okay, I need to know how should I behave morally...	6	2	6	3
Is morality and ethics the same?	5	7	8	9
What are the things that I do to be immoral?	3	5	3	3
Give me some examples of moral actions...	0	7	6	8
Alright, morality?	0	0	2	0
What is integrity?	3	0	4	3
Be moral!	5	5	5	2
I really like our discussion on morality and ethics...	1	1	2	1
What do you like to talk about?	4	4	7	4
What do you think about Tesla?	8	7	6	6
What do you think about Bill Gates?	8	0	5	4
What do you think about Messi?	7	6	3	6
What do you think about Cleopatra?	9	1	8	0
What do you think about England during the reign of Elizabeth?	1	2	5	0
What is your job?	7	7	6	6
What do you do?	7	6	7	6

Table A.8: Survey results for validating the plausible response descriptions. 1/2

Input	Yes	No
Hello!	4	0
How are you?	3	1
What's your name?	3	1
When were you born?	2	2
What year were you born?	2	2
Where are you from?	4	0
Are you a man or a woman?	2	2
Why are we here?	2	2
Okay, bye!	4	0
See you later.	3	1
My name is David, what is my name?	2	2
My name is John, what is my name?	2	2
Are you a leader or a follower?	1	3
Are you a follower or a leader?	1	3
Who is Skywalker?	2	2
Who is Bill Clinton?	2	2
Is sky blue or black?	2	2
Does a cat have a tail?	4	0
Does a cat have a wing?	4	0
Can a cat fly?	4	0
How many legs does a cat have?	2	0
How many legs does a spider have?	3	1
How many legs does a centipede have?	3	1
What is the color of the sky?	3	1
What is the color of water?	2	2
What is the color of blood?	2	2
What is the usual color of a leaf?	3	1
What is the color of a yellow car?	2	2
How much is two plus two?	2	2
How much is ten minus two?	2	2
What is the purpose of life?	2	2

Table A.9: Survey results for validating the plausible response descriptions. 2/2

Input	Yes	No
What is the purpose of living?	2	2
What is the purpose of existence?	2	2
Where are you now?	3	1
What is the purpose of dying?	2	2
What is the purpose of being intelligent?	2	2
What is the purpose of emotions?	2	2
What is moral?	2	2
What is immoral?	2	2
What is morality?	2	2
What is the definition of altruism?	1	3
OK... so what is the definition of morality?	2	2
Tell me the definition of morality, I am quite upset now!	2	2
Tell me the definition of morality.	2	2
Look, I need help, I need to know more about morality...	1	3
Seriously, what is morality?	1	3
Why living has anything to do with morality?	2	2
Okay, I need to know how should I behave morally...	1	3
Is morality and ethics the same?	3	1
What are the things that I do to be immoral?	1	3
Give me some examples of moral actions...	2	2
Alright, morality?	1	3
What is integrity?	2	2
Be moral!	1	3
I really like our discussion on morality and ethics...	3	1
What do you like to talk about?	4	0
What do you think about Tesla?	2	2
What do you think about Bill Gates?	3	1
What do you think about Messi?	3	1
What do you think about Cleopatra?	3	1
What do you think about England during the reign of Elizabeth?	3	1
What is your job?	4	0
What do you do?	2	2