Entity Matching and Disambiguation Across Multiple Knowledge Graphs

by

Michael Farag

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2019

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Knowledge graphs are considered an important representation that lie between free text on one hand and fully-structured relational data on the other. Knowledge graphs are a back-bone of many applications on the Web. With the rise of many large-scale open-domain knowledge graphs like Freebase, DBpedia, and Yago, various applications including document retrieval, question answering, and data integration have been relying on them.

In this thesis, We are primarily interested in knowledge graphs from the perspective of integrating disparate heterogeneous sources, with an eye towards applications such as document retrieval and question answering. Integrating different knowledge graphs is very important for enriching the knowledge shared among them. The core part of this integration process is matching entities across the knowledge graphs. The biggest challenge to entity matching is the ambiguity. The obvious solution is to make use of the graph structure and entity neighbourhoods for matching and disambiguating entities.

We formalize the entity matching problem and present the first large-scale dataset, AMBIGUOUS DBPEDIA-WIKIDATA, for this task based on exiting cross-ontology links between DBpedia and Wikidata, focused on several hundred thousand ambiguous entities. We propose an entity matching framework that is capable of disambiguating entities across different knowledge graphs. The framework consists of fuzzy string matcher and graph embedding-based matcher. Using a classification-based approach, we find that a simple multi-layered perceptron based on representations derived from RDF2VEC graph embeddings of entities in each knowledge graph is sufficient to achieve high accuracy, with only limited training data. The contribution of our work is both a large dataset for examining this problem and strong baselines on which future work can be based.

We also present SIMPLEDBPEDIAQA, a new benchmark dataset for simple question answering over knowledge graphs that was created by mapping SIMPLEQUESTIONS entities and predicates from Freebase to DBpedia. We show how entity matching using manual annotations can be used for migrating datasets across knowledge graphs. Although this mapping is conceptually straightforward, there are a number of nuances that make the task non-trivial, owing to the different conceptual organizations of the two knowledge graphs.

Finally, if manual annotations are scarce, we show how our entity matching framework can be used to generate *free* annotations to train our model and then use it for disambiguation. In that essence, we introduce SIMPLEQUESTIONS++, a new question answering benchmark that have all questions linked to Freebase, DBpedia, and Wikidata.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Ihab Ilyas for the continuous support during my study and research, for his patience, motivation, and immense knowledge. His guidance and expertise helped me throughout my Master's degree.

I would also like to thank Prof. Jimmy Lin for his insightful comments and encouragement, for his guidance and clear vision. He was always available whenever I needed his advice.

I would also like to thank Prof. Grant Weddell for his valuable comments and questions during writing my thesis.

I am grateful to all of those with whom I have had the pleasure to work during my Master's degree, especially Peng for being an excellent collaborator in my research.

I would like to extend my gratitude to all friends I met in Waterloo. To Abdelrahman and all our endless debates, heartfelt laughter, and cooking moments. To Mina, Salam, Ahmed, Hamouda, Mohsen, Hameedo, Omar, Hemant, Royal, Melica, George, Chathura, Amine, Siddhartha and all colleagues and friends who made my stay in Waterloo enjoyable and feel like home.

Nobody has been more important to me in the pursuit of this degree than my family. I would like to thank my parents whose love and support are always with me in whatever I pursue. My mother has always been my source of encouragement, her love and care never ceased to help me during the hardest moments. To my brother, who was always in mind during my stay abroad. To my grandmother whom her weekly calls always made me cheerful. To my lovely cousins and their continuous love and support.

Dedication

Dedicated to the memory of my grandfather, Rizkallah, who always believed in my ability to be successful and impactful. You are gone but your belief in me has made this journey possible. I will be ever grateful for your unconditional love and support.

Table of Contents

List of Tables

List of Figures

Abbreviations

AMT Amazon Mechanical Turk 16

BiGRU Bidirectional Gated Recurrent Unit 42

BiLSTM Bidirectional Long Short-Term Memory 41, 42

CBOW Continuous Bag-of-Words xi, 9, 11

CNN Convolutional Neural Network 17, 42

CRF Conditional Random Field 17, 41, 42

GRU Gated Recurrent Unit 17

IRI Internationalized Resource Identifier 5

LOD Linked Open Data 5, 6

LR Logistic Regression 17, 18, 21, 28, 42

LSTM Long Short-Term Memory 17

MLP Multi Layer Perceptron 2, 18, 19, 21, 26, 28, 29

MRR Mean Reciprocal Rank ix, xi, xii, 22, 24–30, 51, 53, 58, 60

NLP Natural Language Processing 42, 44

OAEI Ontology Alignment Evaluation Initiative 14, 15

RDF Resource Description Framework 5, 6, 8, 12–15, 22, 32, 33

URI Uniform Resource Identifier 32

W3C World Wide Consortium 4, 5

Chapter 1

Introduction

Knowledge graphs form an important representation that lie between free text on one hand and fully-structured relational data on the other. Knowledge graphs have proven useful for many applications, including document retrieval [16] and question answering [31]. We are primarily interested in them from the perspective of integrating disparate heterogeneous sources, with an eye towards applications such as document retrieval and question answering. In this context, there are two main challenges that need to be tackled: First, linking mentions extracted from free text to entities in a knowledge graph. As there already exists many large-scale efforts such as Freebase [9], DBpedia [2], Wikidata [46], and YAGO [42], to support interoperability there is a need to match entities across multiple resources that refer to the same real-world entity. Addressing this challenge would, for example, allow mentions in free text that have been linked to entities in one knowledge graph to benefit from knowledge encoded elsewhere.

Ambiguity, of course, is the biggest challenge to this problem; for example, there are 21 persons named Adam Smith in DBpedia and 24 in Wikidata. The obvious solution is to exploit the context of entities and leverage the structured graph nature of the knowledge graphs for matching and disambiguating entities.

In this thesis, we propose an entity matching[1] framework for matching and disambiguating entities across multiple, and possibly heterogeneous knowledge graphs. The proposed framework consists of a fuzzy string matcher and a graph embedding-based matcher. Together, they provide a simple yet effective solution for the entity matching problem. We show how graph embeddings can be exploited to boost the disambiguation effectiveness

[1] We use entity mapping and entity matching interchangeably througout the thesis

using a simple Multi Layer Perceptron (MLP) model. We also show how our framework can work without the need of handcrafted annotations.

Furthermore, we provide the community with a large-scale dataset for entity matching that focuses on ambiguous cases to pave the way for future work. We extensively evaluate our model on the new benchmark and examine the effects of varying the number of needed *free* annotations.

We also present SimpleDBpediaQA, a new dataset that we have created by mapping entities and predicates that comprise the answers to SimpleQuestions dataset from Freebase to DBpedia. Unlike Freebase, DBpedia is actively maintained by a dedicated community. We describe how this dataset migration is accomplished via high-quality alignments between entities in the two different knowledge graphs, and explain many of the nuances that make the creation of this dataset non-trivial.

Furthermore, we introduce SimpleQuestions++, a new question answering benchmark that is also based on the popular SimpleQuestions dataset. The introduced benchmark consists of a subset of questions that we linked to Freebase, DBpedia, and Wikidata. This enables future benchmarking of question answering models on different knowledge graphs to show how the model performs in different settings and different knowledge graphs structures.

Finally, We showcase how our proposed entity matching framework can be used for generating free question answering benchmarks on multiple knowledge graphs given any dataset that is only linked to one knowledge graph. In essence, that enables training multiple question answering models for multiple knowledge graphs using only one seed benchmark.

1.1 Contributions

The main contributions of the thesis are summarized in the following:

- We formulate the problem of entity matching across knowledge graphs and propose a simple yet strong framework that can effectively match and disambiguate entities across different knowledge graphs.
- We provide the community with the first, to our knowledge, large scale dataset that focuses on the ambiguous entity matching problem and evaluate our proposed framework on it.

- We introduce SimpleDBpediaQA, a high-quality large-scale dataset which we created by mapping the original SimpleQuestions dataset from Freebase to DBpedia.
- We also introduce SimpleQuestions++, a unified SimpleQuestions dataset that include a big subset of questions that are linked to Freebase, DBpedia, and Wikidata.
- We use SimpleQuestions++ as a downstream use case for our proposed entity matching framework and show how we can exploit a question answering dataset linked to a knowledge graph to create another question answering model on another knowledge graph.

1.2 Thesis Organization

Chapter 2 reviews some of the needed background concepts and fundamentals for understanding the covered topics in the thesis and presents an overview for the relevant related work on entity matching across multiple knowledge graphs as well as factoid simple question answering over knowledge graphs. Chapter 3 describes in detail the proposed framework for entity matching and disambiguation across multiple knowledge graphs and presents a novel large-scale dataset for ambiguous entity mapping across DBpedia and Wikidata. Chapter 4 introduces the SimpleDBpediaQA dataset and describes in details the migration process. Chapter 5 explains SimpleQuestions++ dataset which is the updated version of the original SimpleQuestions dataset and includes entities linked into Freebase, DBpedia, and Wikidata. Finally, Chapter 6 concludes the thesis.

Chapter 2

Background and Related Work

In this chapter, we briefly revise some of the topics that are necessary for understanding the work done in this thesis. After that, we discuss some of the previous work done on entity matching in general and entity matching across knowledge graphs in particular. We also discuss previous work done on question answering over knowledge graphs and the popular datasets used for benchmarking question answering models in order to lay foundation to our contribution.

2.1 The Semantic Web and Knowledge Graphs

The Semantic Web[1], proposed by Sir Tim Berners-Lee in [6], provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by World Wide Consortium (W3C)[2] with participation from a large number of researchers and industrial partners. Originally, the Semantic Web aimed to facilitate the integration and combination of data collected from various resources by defining common formats for interchanging documents on the web. Recently, it has also been extended as a language for describing how the data relates to real-word entities.

> The Semantic Web is not just about putting data on the web. It is about making links, so that a person or machine can explore the web of data. With

[1] https://www.w3.org/2001/sw/
[2] https://www.w3.org/

linked data, when you have some of it, you can find other, related, data. (Tim Berners-Lee)

2.1.1 Resource Description Framework

Since the beginning, the Semantic Web has promoted a graph-based representation of knowledge, e.g., using the Resource Description Framework (RDF) standard[3]. As defined by the W3C recommendation for RDF[4], it is a framework for representing information in the Web. RDF graphs are sets of *subject-predicate-object* triples, where the elements may be Internationalized Resource Identifier (IRI), blank nodes, or data-typed literals. They are used to express descriptions of resources. In such a graph-based knowledge representation, *entities*, which are the nodes of the graph, are connected by *predicates*, which are the edges of the graph (e.g., `Alexander the Great founded Alexandria`), and *entities* can have *types*, denoted by *is a* predicate (e.g., `Alexandria is a City`). The sets of possible *types* and *predicates* are organized in a *schema* or *ontology*, which defines their interrelations and restrictions of their usage.

More formally, we adopt the definition in [36].

Definition 2.1.1. RDF An RDF graph is a labeled graph $G = (V, E)$, where V is a set of vertices, and E is a set of directed edges, where each vertex $v \in V$ is identified by a unique identifier, and each edge $e \in E$ is labeled with a label from a finite set of edge labels.

2.1.2 Linked Open Data

As the Semantic Web emerged, the necessity of organizing and sharing data over the web has increased. Linked Data [7] has been proposed as a way of using the *Web* to connect related data that wasn't previously linked, or using the Web to lower the barriers to linking data currently linked using other methods[5]. More specifically, Wikipedia defines Linked Data as "a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF."

To further facilitate these efforts, Linked Open Data (LOD)[6] was introduced in [7]. LOD is an open, interlinked collection of datasets in machine-interpretable form, covering

[3] https://www.w3.org/RDF/
[4] https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
[5] http://linkeddata.org/
[6] https://www.w3.org/DesignIssues/LinkedData.html

multiple domains from life sciences to government data [39]. As a result of interlinking different datasets together, the collection is conceptually seen as one large knowledge graph, although very heterogeneous. Currently, the LOD cloud contains 1,234 interlinked datasets with 16,136 links (as of June 2018). Figure 2.1 depicts the structure of the LOD cloud[7].

2.1.3 Knowledge Graphs

As mentioned in [34], the term *Knowledge Graph* was coined by *Google* in 2012, referring to their use of semantic knowledge in Web Search ("Things, not string"), and is recently also used to refer to Semantic Web knowledge bases DBpedia, Freebase, Wikidata, YAGO. From a broader perspective, any graph-based representation of knowledge can be considered a knowledge graph (e.g., any RDF dataset). Nonetheless, there is no definition that is agreed upon regarding what a knowledge graph is. To refrain from defining a formal definition, we can outline the desired characteristics of a knowledge graph. A knowledge graph:

- mainly describes real world entities and their interrelations, organized in a graph.
- defines possible classes and relations of entities in a schema.
- allows for potentially interrelating arbitrary entities with each other.
- covers various topical domains.

Freebase

Freebase [10] is a popular public, editable open knowledge graph with schema templates for most kinds of possible entities (i.e., persons, cities, movies, etc.). The data in it is collaboratively structured and maintained. Freebase uses the notions of *objects*, *facts*, *types*, and *properties*. Each object is uniquely identified by a *MID* (Machine ID). An object has one or more types, and uses properties with these types to represent facts. According to [34], the last version of Freebase contains roughly 50 million entities and 3 billion facts. Freebases schema comprises roughly 27,000 entity types and 38,000 relation types. Freebase was launched by *Metaweb* in 2007 and acquired by Google in 2010. Unfortunately, in 2014 Google decided to shutdown Freebase and it is now defunct.

[7] https://lod-cloud.net/

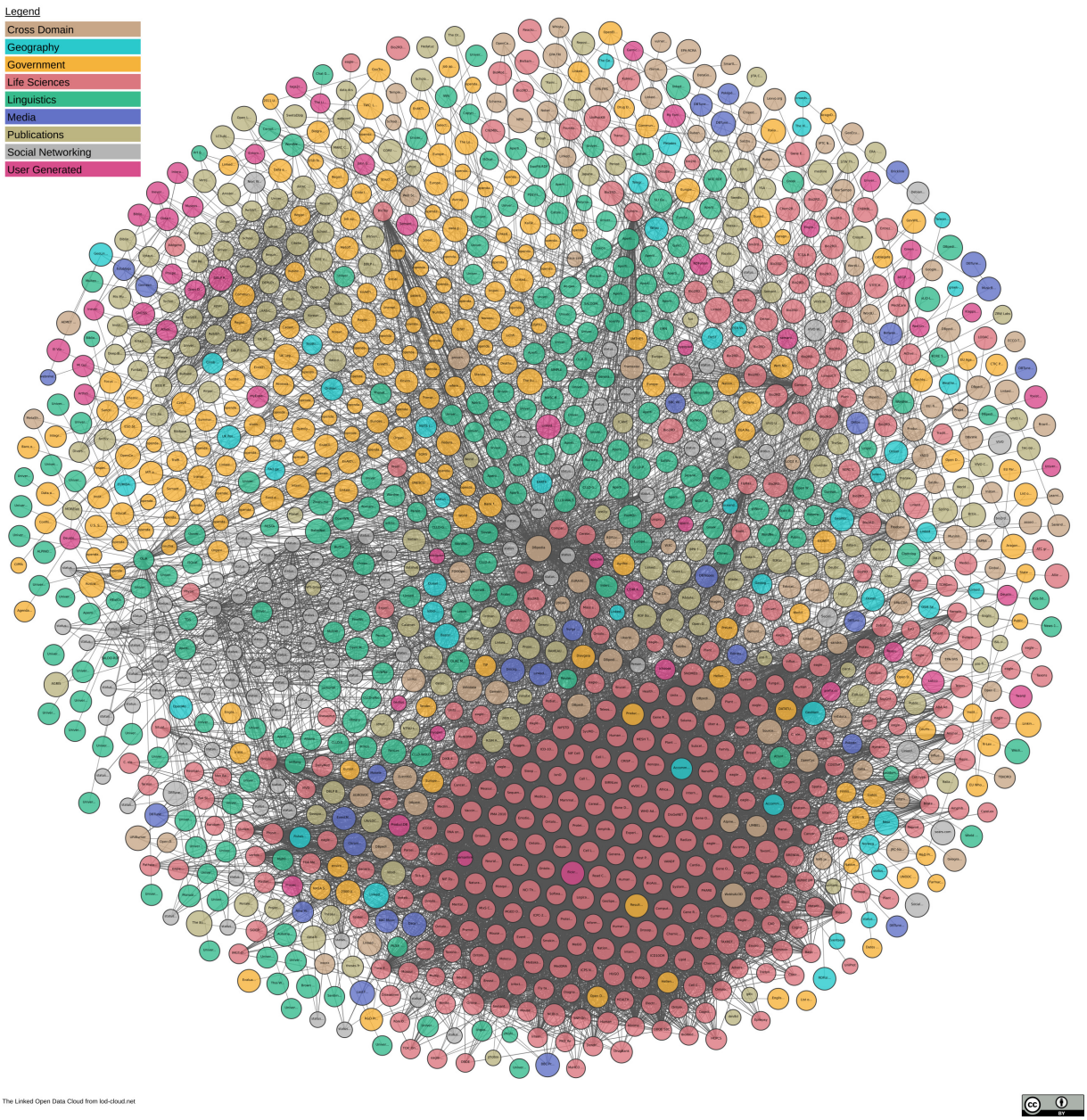


Figure 2.1: The Linked Open Data cloud

DBpedia

DBpedia [8] is another very popular knowledge graph which is extracted from structured data in Wikipedia. Typically, key-value pairs in the *infobox* tables of Wikipedia are first extracted and then mapped, in a crowd-sourced process, to DBpedia. Keys and types of the extracted infoboxes are mapped to DBpedia's properties and ontology respectively. The most recent DBpedia release (2016-10)[8] is based on updated Wikipedia dumps from October 2016. The latest release consists of 13 billion RDF triples. 1.7 billion triples were extracted from the English edition of Wikipedia, 6.6 billion triples were extracted from other language editions, and the rest was extracted from Wikipedia Commons[9] and Wikidata. The latest ontology includes 760 classes, and 2859 properties.

Wikidata

Wikidata [46] is a collaboratively edited knowledge base hosted by the Wikimedia Foundation[10]. Wikidata is interlinked to many languages that are featured in WIkipedia which makes it very rich. After the shutdown of Freebase[11], the data contained in Freebase is subsequently moved to Wikidata. Currently, Wikidata contains roughly 54 million instances[12] and 673 million statements[13]. Its schema defines roughly 23,000 types[14] and 5,816 relations[15].

2.2 Word Embedding

Natural language models have been proposed to overcome the drawbacks of traditional language models (e.g. bag of words) such as high dimensionality and data sparsity. Neural language models avoid this problem by representing words in a distributed way, as non-linear combinations of weights in a neural network. The goal of these approaches is estimating the likelihood of a specific sequence of words appearing in a corpus by analyzing the context.

[8] http://wiki.dbpedia.org/blog/new-dbpedia-release-%E2%80%93-2016-10
[9] https://commons.wikimedia.org/wiki/Main_Page
[10] https://en.wikipedia.org/wiki/Wikimedia_Foundation
[11] http://plus.google.com/109936836907132434202/posts/3aYFVNf92A1
[12] https://www.wikidata.org/wiki/Wikidata:Statistics
[13] https://tools.wmflabs.org/wikidata-todo/stats.php
[14] http://tools.wmflabs.org/wikidata-exports/miga/?classes#_cat=Classes
[15] https://www.wikidata.org/w/index.php?title=Special:ListProperties

More specifically, a word embedding, $W : w \rightarrow \mathbb{R}^d$ is a parameterized function that maps a word, w, to a d-dimensional vector space. Typically, the function is a lookup table parameterized by a matrix, θ, with a row n for each word [33]:

$$W_\theta(w_n) = \theta_n$$

Word embeddings encapsulate latent semantics of words by exploiting the observation that semantically-similar words appear in similar contexts. Word embeddings are usually trained on large corpora in an unsupervised fashion.

One of the most popular and widely-adopted neural language models is word2vec [29, 30]. Word2vec is a computationally-efficient two-layer neural network model for learning word embeddings. It comes with two model architectures: (a) CBOW model, and (b) Skip-Gram model. Figure 2.2 shows the architecture of the models.

2.2.1 Continuous Bag-of-Words Model

CBOW model is used to predict a target word from its context within a given window. The input layer consists of all the words before and after the target word w_t with the given window. The input vectors retrieved from the input weight matrix are averaged and then projected. After that, the output weights which are retrieved from the output weight matrix are used to compute a score for each word in the vocabulary. The score is the probability of the word being a target word. Given a sequence of words $w_1, w_2, \ldots, w_T$, and a context window c, then the objective of CBOW is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T} \log P(w_t|w_{t-c}w_{t-1}\ldots w_{t+1}w_{t+c}), \tag{2.1}$$

where the probability $P(w_t|w_{t-c}w_{t-1}\ldots w_{t+1}w_{t+c})$ is calculated using the softmax function:

$$P(w_t|w_{t-c}w_{t-1}\ldots w_{t+1}w_{t+c}) = \frac{\exp(\bar{v}^T v'_{w_t})}{\sum_{w=1}^{|V|}\exp(\bar{v}^T v'_{w_t})}, \tag{2.2}$$

where v'_w is the output vector of the word w, V is the complete vocabulary of words, and $\bar{v}$ is the averaged input vector of all the context words:

$$\bar{v} = \frac{1}{2c} \sum_{-c \leq j \leq c, j \neq 0} v_{w_{t+j}} \tag{2.3}$$

2.2.2 Skip-Gram Model

Inversely, the skip-gram model is used to predict the context words given the target word. Given a sequence of words $w_1, w_2, \ldots, w_T$, and a context window c, the objective of the skip-gram model is to maximize the average log probability:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{t+j}|w_t), \tag{2.4}$$

where the probability $P(w_{t+j}|w_t)$ is calculated using the softmax function:

$$P(w_o|w_i) = \frac{\exp(v'^T_{w_o} v_{w_i})}{\sum_{w=1}^{|V|} \exp(v'^T_{w_o} v_{w_i})}, \tag{2.5}$$

where v_w and v'_w are the input and the output vectors of the word w, and V is the complete vocabulary of words.

Since computing the softmax function for both models is inefficient due to the dependence on the size of the vocabulary; two optimization techniques have been introduced: (a) hierarchical softmax, and (b)negative sampling. Emperical analysis have shown that negative sampling yields better performance in most cases.

2.3 Graph Embedding

Graphs are a natural way of representing diverse real-world phenomena. The ubiquity of graphs has been manifested in different scenarios. Social media networks are clear examples of how graph models can be useful. Other applications include citation networks in research, protein-protein interaction networks in biology, and knowledge graphs. Analyzing different graphs can help us understand how networks interact in a systematic way. Graph analytics have enabled various applications on top of graphs such as friendship or followers recommendations and community detection in social networks, content recommendations in multimedia, and interactions between proteins.

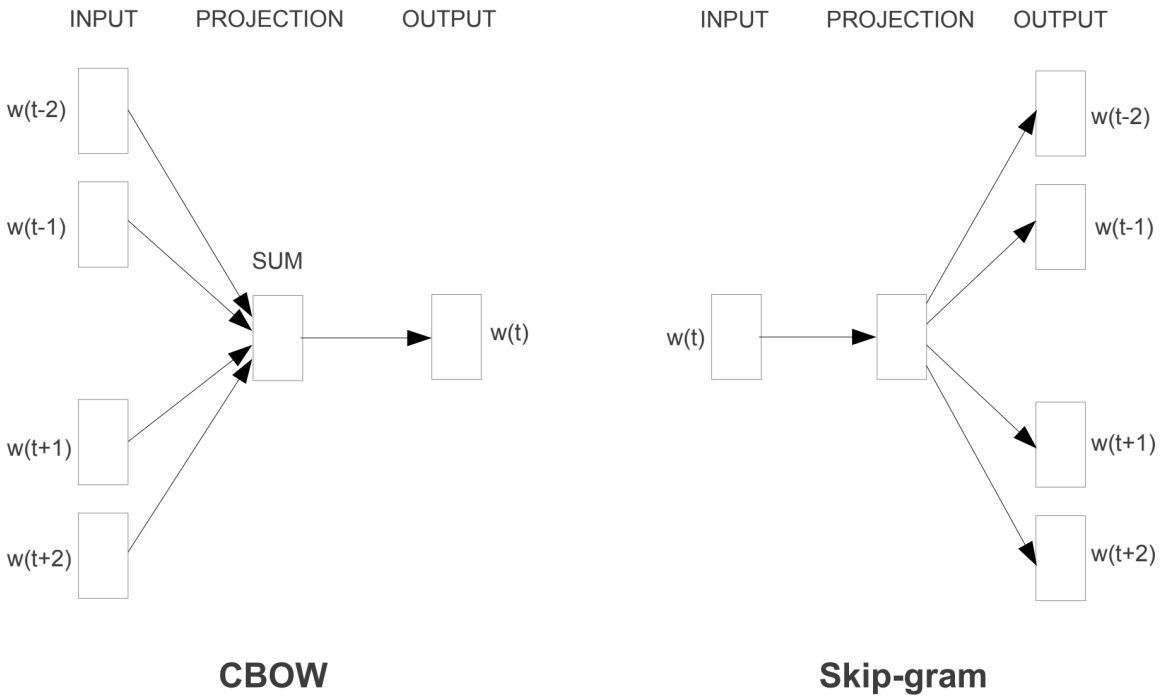


Figure 2.2: Architecture of CBOW and Skip-gram models [29]

Graph analytics can be classified according to [23] into node classification, link prediction, clustering, and visualization. Many techniques have been proposed to tackle these tasks either by directly working on the original graph or on a vector representation of it.

With the rise of machine learning as an effective technique for solving different problems, obtaining a vector representation of graphs has become essential. This vector representation is usually referred to as an embedding. Graph embedding has proven to be advantageous for machine learning and has gained much attention and popularity recently. In particular, graph embedding takes a graph as input and outputs a low-dimensional vector representation of part or whole of the graph while preserving its information [12].

The output of the graph embedding can be of different granularity as follows:

- **Node embedding**: Each node in the graph is represented as a vector in low-dimensional space such that nodes that are similar in the graph have similar vector representations. Differences between graph embedding techniques arise from how they define the similarity of two nodes and how they encode this information in vector representations.
- **Edge embedding**: Each edge in the graph is represented as a vector in low-dimensional space.
- **Hybrid embedding**: It is a combination of node-edge embedding.
- **Whole graph embedding**: It is usually used to encode small graphs such as proteins or molecules. Two similar graphs should have similar embedding.

2.3.1 RDF2VEC

RDF2Vec [37] is a popular graph embedding technique for RDF graphs. It adopts neural language models which was first introduced in [29, 30] to embed natural language text into vector space word embeddings. Word embeddings were proposed to create low-dimensional vector representations of words that satisfy two essential properties: (a) similar words are close in the vector space, and (b) arithmetic operations on relations between words can also be represented as vectors.

In case of RDF graphs, there are entities and predicates between entities in contrast to words in natural language text. To adapt the neural language models to RDF, the graph has to be first transformed into sentences comprising sequences of entities with their relations to other entities. After that, the same neural language models can be trained on

the generated sentences to output a vector representation for each entity preserving the same properties.

RDF Graph Sub-Structures Extraction

To generate sentences (sequences) from the RDF, two approaches are used: (a) graph walks, and (b) Weisfeiler-Lehman Subtree RDF graph kernels.

Definition 2.3.1. Graph Sequence It is a sequence of vertices v_i and edges e_i such that for $1 \leq i \leq k$, the edge e_i has endpoints v_{i-1} and v_i, and the length of the sequence is the number of edges in it.

For each vertex $v \in V$, the goal is to generate a set of sequences S_v such that for each sequence $s \in S_v$, the first vertex is v.

Graph Walks Given a graph $G = (V, E)$, for each vertex $v \in V$, we generate all graph walks P_v of depth d rooted at vertex v. A breadth-first algorithm is used to generate these walks. In the first iteration, the direct outgoing edges of root vertex v_r are generated. In the next iterations, the same procedure is applied to each previously visited vertex and so on. The algorithm terminates after d iterations. The output of this algorithm is the set of graph sequences $P_G = \bigcup_{v \in V} P_v$.

Weisfeiler-Lehman Subtree RDF Graph Kernels The Weisfeiler-Lehman sub-tree graph kernel [40] is an efficient, state-of-the-art, kernel for graph comparison. The approach is to count the number of shared sub-trees between two or more graphs using Weisfeiler-Lehman test of graph isomorphism. To adapt this approach to the RDF graph, [18, 17] did two modification on the original Weisfeiler-Lehman algorithm. First, since the RDF graph has directed edges, this means that the neighbours of a vertex v are only the vertices reachable by its outgoing edges. Second, in the original algorithm, the labels of two previous iterations might be different while representing the same sub-tree. To handle this, the labels from a previous iteration is kept in track and if they are equivalent to the labels in the current iteration they are reused.

Converting the RDF graph to a sequence of entities and relations is as follows: (1) given a graph $G = (V, E)$, the parameters needed for the Weisfeiler-Lehman algorithm are chosen where h is the number of iterations, and d is the vertex subgraph depth which defines the subgraph in which the subtrees will be counted for a vertex v. (2) after each

iteration, for each vertex v, all the paths of depth d are extracted from the relabeled graph. The original label of the vertex v is set to be the first token of each generated sequence. (3) step (2) is repeated h times. The output of this algorithm is a set of graph sequences of all iterations $P_G = \bigcup_{i=1}^{h} \bigcup_{v \in V} P_v$.

Neural Language Models After that, the generated sentences using both techniques are used to train a Word2vec model as discussed in section 2.2.

2.4 Entity Matching

Research related to knowledge graph integration comes from the database community and focuses on ontology matching—referred to as record-linkage, entity resolution, or deduplication. Examples include MAGELLAN [26], DEEPER [20], and the work of [32]. The primary difference between these work and ours is that they assume relational data and furthermore that the tables to be matched have been already aligned using schema matching techniques.

For example, MAGELLAN [26] is an entity matching system that is built in Python data-science eco-system to cover the entire entity matching pipeline (e.g. blocking, matching, debugging, sampling). Also DEEPER [20] is an entity resolution system that captures syntactic and semantic similarities of tuples using deep learning models. In [32], they categorize the entity matching problem space into structured, textual, and dirty entity matching. However, these systems are tailored for relational data and they assume that the tables to be matched have been already aligned using schema matching techniques. These systems cannot be directly applied to entity matching across knowledge graphs due to the difference in structure of both the relational model and RDF model.

The Semantic Web community has studied the problem of matching entities across knowledge graphs. For example, researchers have introduced the Ontology Alignment Evaluation Initiative (OAEI) to focus on ontology matching of knowledge graphs. However, the benchmarks used in the evaluation are quite small in scale. For example *spimbench* benchmark[38] has a total of only 1800 instances and 50,000 triples. In comparison, our dataset is much bigger on the scale of hundreds of thousands.

According to [14], entity matching techniques in knowledge graphs are classified into: (1) value-oriented techniques that define the similarity between instances on the attributes level and an appropriate matching technique is used according to attribute type and (2)

record-oriented techniques which includes learning-based, similarity-based, rule-based, and context-based techniques.

The best approaches that performed well in OAEI 2017 either rely on logical reasoning as in [24] or on textual features as in [1]. In constrast, Our problem formulation and solution are different for the following reasons: (1) we use graph embedding to capture the semantics and structure of the knowledge graphs without the need for hand-crafted features, (2) our system does not require any schema mappings, (3) our approach can make use of the graph nature of the RDF which has extra knowledge in terms of the connectivity between nodes and how they relate to one another.

Furthermore, RDF ontologies define a class (type) hierarchy that can be used to provide more semantics in the matching process. As observed in Section 3.6.2, our system can make use of these typing information implicitly.

2.5 Question Answering

Question answering over knowledge graphs is an important problem at the intersection of multiple research communities, with many commercial deployments. To ensure continued progress, it is important that open and relevant benchmarks are available to support the comparison of various techniques. In this thesis, as a use case, we focus on the class of questions that can be answered by a single triple (i.e., fact) from a knowledge graph. For example, the question "What type of music is on the album Phenomenon?" can be answered via the lookup of a simple fact—in this case, the "genre" property of the entity "Phenomenon". Analysis of an existing benchmark dataset [47] and real-world user questions [15, 45] show that such questions cover a broad range of users' needs.

The SimpleQuestions dataset [11] has emerged as the de facto benchmark for evaluating these simple questions over knowledge graphs. However, there is one major deficiency with this resource: the answers draw from Freebase. Unfortunately, Freebase is defunct and no longer maintained. This creates a number of insurmountable challenges: First, because the knowledge graph is stale, it is no longer possible to build a "real-world" operational QA system using models trained on SimpleQuestions. Second, a defunct knowledge graph means that researchers must develop custom infrastructure for querying, browsing, and manipulating the graph. Thus, we are not able to leverage multiple cooperative and interchangeable service APIs that are deployed and maintained by different parties—which is the strength of the broader "open linked data" ecosystem. While it may be the case that one can apply transfer learning so that models trained on SimpleQuestions can be

re-targeted to another "live" knowledge graph, we are not aware of research along these lines.

The development and continual advance of question answering techniques over knowledge graphs require benchmark datasets that cover different aspects of the task. Quite obviously, each dataset has to target one (or more) knowledge graphs, which means that the structure of the answers are dictated by the conceptual organization of the particular knowledge graph.

Over the years, researchers have built a number of datasets based on Freebase [10]. For instance, FREE917 [13] contains 917 questions involving 635 distinct Freebase predicates. WEBQUESTIONS [5] contains 5,810 question-answer pairs collected using the Google Suggest API and manually answered using Amazon Mechanical Turk (AMT). Both contain answers that require complex, multi-hop traversals of the knowledge graph. In contrast, the SIMPLEQUESTIONS dataset focuses on questions that can be answered via the lookup of a single fact (i.e., triple). Due to its much larger size and thus support for data-hungry machine learning techniques, this dataset has gained great popularity with researchers. Unfortunately, Google shut down Freebase in 2015; a final snapshot of the knowledge graph is still available online for download, but the associated APIs are no longer available.

Like Freebase, DBpedia [8] has also been used as the target knowledge graph for multiple question answering datasets. For example, QALD[16] (Question Answering over Linked Data) is a series of evaluation campaigns focused on question answering over linked data. LC-QUAD [44] is another recent dataset that comprises 5,000 questions with answers in the form of SPARQL queries over DBpedia. These questions are relatively complex and require the integration of evidence from multiple triples. However, a more recent analysis by Singh et al. [41] found that only 3,252 of the questions returned answers using the provided queries.

We are not the first to attempt to migrate SIMPLEQUESTIONS to another knowledge graph. Diefenbach et al. [19] mapped the dataset from Freebase to Wikidata.[17] However, our migrated SIMPLEDBPEDIAQA dataset has roughly twice the number of mapped questions. DBpedia is generally considered to be more mature than Wikidata due to its longer history, and thus we believe targeting DBpedia will ultimately yield higher-impact applications.

[16] https://qald.sebastianwalter.org/
[17] https://www.wikidata.org/wiki/Wikidata:Main_Page

Question Answering Baseline

In this thesis, we use simple yet strong baselines proposed in recent work by Mohammed et al. [31], who applied techniques with and without neural networks to SimpleQuestions. We adapt their open-source code[18] as a strong question answering baseline when needed throughout the thesis.

We briefly describe their approach, which decomposes into four tasks:

- **Entity Detection**: Given a question, the task is to identify the topic entity of the question. For this task, we examined bidirectional Long Short-Term Memory (LSTM)s and Conditional Random Field (CRF)s.
- **Entity Linking**: Detected entities (text strings) need to be linked to entities in the knowledge graph (e.g., URI from DBpedia in our case). This is formulated as a string matching problem: Levenshtein Distance is used along with a few heuristics for ranking candidate entities.
- **Predicate Prediction:** Given a question, the task is to identify the predicate being queried. We examined three models: bidirectional Gated Recurrent Unit (GRU), Convolutional Neural Network (CNN), and Logistic Regression (LR). The first two are standard neural network models; for logistic regression we used as input the average of the word embeddings of each word. BiGRU was selected over BiLSTM based on the experiments of Mohammed et al. [31], where it was found to be slightly more accurate.
- **Evidence Integration:** With m candidate entities and r candidate predicates from the previous components, the evidence integration model selects the best (entity, predicate) pair based on the product of each component score as well as a number of heuristics.

[18] http://buboqa.io/

Chapter 3

Entity Matching and Disambiguation

3.1 Introduction

Knowledge graphs form an important representation that lie between free text on the one hand and fully-structured relational data on the other. Knowledge graphs have proven useful for many applications, including document retrieval [16] and question answering [31]. We are primarily interested in them from the perspective of integrating disparate heterogeneous sources, with an eye towards applications such as document retrieval and question answering. In this context, there are two main challenges that need to be tackled: First, linking mentions extracted from free text to entities in a knowledge graph. As there already exists many large-scale efforts such as Freebase [9], DBpedia [2], Wikidata [46], and YAGO [42], to support interoperability there is a need to match entities across multiple resources that refer to the same real-world entity. Addressing this challenge would, for example, allow mentions in free text that have been linked to entities in one knowledge graph to benefit from knowledge encoded elsewhere.

In this chapter, first, we offer the community the first large-scale dataset for entity matching, focusing on ambiguous entities. Second, we show that a simple model performs well on this dataset [4]. Results suggest that RDF2Vec can capture the context of entities in a low dimensional semantic space, and that it is possible to learn associations between distinct semantic spaces (one from each knowledge graph) using a simple MLP to perform entity matching with high accuracy. Experiment show that only small amounts of training data are required, but a linear model (LR) on the same graph embeddings performs poorly. Naturally, we are not the first to have worked on aligning knowledge graphs. While

more explorations are certainly needed, to our knowledge we are the first to make these interesting observations.

3.2 Problem Formulation

We begin by formalizing our entity matching problem. Given a source knowledge graph S containing entities $E^s = \{e_1^s, e_2^s, \ldots, e_m^s\}$, where e_i is a Uniform Resource Identifier (URI), for each entity we wish find the entity (or possible entities) in the target knowledge graph T containing entities $E^t = \{e_1^t, e_2^t, \ldots, e_n^t\}$ that corresponds to the same real-world entity. This correspondence appeals to the common-sense notion that these entities refers to the same person, location, etc. In our current formulation, we take a query-based approach: that is, for a given "query" entity in the source knowledge graph, our task is to determine the best matching entity (or set of entities) in the target knowledge graph.

3.2.1 Entity Ambiguity

Ambiguity, of course, is the the biggest challenge to this problem: for example, there are 21 persons named "Adam Smith" in DBpedia and 24 in Wikidata. The obvious solution is to exploit the context of entities for matching. In the next section, we present a dataset for entity matching between DBpedia and Wikidata that specifically focuses on ambiguous cases. Interestingly, experimental results show that with a classification-based formulation, an off-the-shelf graph embedding, RDF2Vec [37], combined with a simple MLP achieves high accuracy on this dataset.

3.3 Entity Matching Framework

Our entity proposed entity matching approach consists of two components: (1) Fuzzy String Matcher, and (2) Graph embedding-based Matcher. As discussed in Section 3.2, the inputs to the framework are the source and the target knowledge graphs S and T containing entities sets E^S and E^T, respectively. The overall architecture of the entity matching framework is shown in Figure 3.1.

First, all entities from the source knowledge graph are fed into the fuzzy string matcher. Then, the unambiguous entities (only one exact match found) are directly mapped to the target knowledge graph. The unambiguous entities are further used as training data for

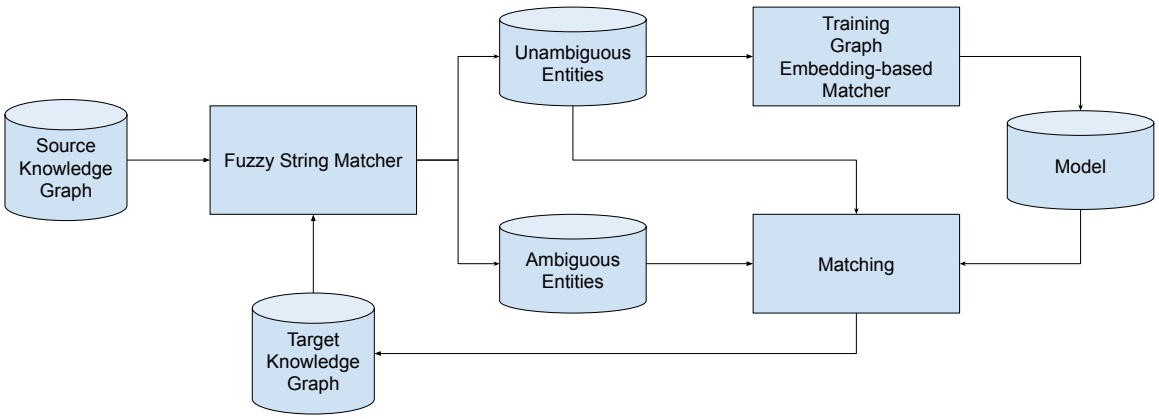


Figure 3.1: Architecture Overview for the Entity Matching Framework

the graph embedding-based matcher. Finally, the trained model is used to disambiguate the ambiguous data and match them to the target knowledge graph.

3.3.1 Fuzzy String Matcher

First, for each entity $e^S \in E^S$, the fuzzy string matcher is used to find the set of the entities $matches(e^S) \subseteq E^T$ that have the exact label of entity e^S. This set acts as a candidates set for e^S, where there is only one entity e^T that is equivalent to e^S, which is what we need to find.

If the candidates set has only one candidate (i.e. $|matches(e^S)| = 1$), then this means there is no ambiguity for entity e^S and we declare that candidate as e^T. Otherwise, if the candidates set has more than one candidate (i.e. $|matches(e^S)| > 1$), then e^S is considered to be an ambiguous entity that needs further investigation.

Let $U^S \subseteq E^S$, $A^S \subseteq E^S$ denote the set of all *unambiguous* and *ambiguous* entities in E^S, where $E^S = U^S \cup A^S$. After that stage, we are left with the set A that needs to be disambiguated to find the actual corresponding target entities. This is the reason we use the graph embedding-based matcher.

Free Positive Examples

It is worth mentioning that the set U can be used as a *free* source of positive *sameAs* mappings. This will be discussed shortly in the next section.

3.3.2 Graph embedding-based Matcher

We propose a classification approach to tackle the entity matching problem across knowledge graphs. Here, we use point-wise training strategy: a classifier is trained on each source–target entity pair and the probability of predicting a match is used for candidate ranking. As mentioned in the previous section, the set of unambiguous entities U can be used as a *free* training set for the classifier without the need of any handcrafted annotations which means that the whole matching process can be done automatically.

As discussed in Section 2.3, a graph embedding is used to represent a graph in a low dimensional semantic space while preserving its structure. Different graph embedding techniques have been introduced recently to capture different aspects of the graph structure. In our case, we need to preserve the structural as well as the semantic features of the nodes (entities) so that semantically-similar nodes are close to each other in the embedding space. For this reason, we decided on using one of best graph embedding algorithms, RDF2Vec [37], as our graph embedding technique.

The RDF graph is firstly unfolded into a set of k sequences of entities with predicates connecting them forming natural language sentences. This is typically performed using two approaches: graph walks and Weisfeiler-Lehman Subtree RDF graph kernels. After that, the generated sentences are used to train a Word2Vec [29, 30] over the natural language output. The outcome of this step is a d-dimensional vector for each entity.

After the above process, the embedding of the query entity in the source knowledge graph and candidate entity in the target knowledge graph are concatenated into one feature vector and then fed into a multi-layer perceptron with one hidden layer using ReLU activation function, followed by fully-connected layer and softmax to output the final prediction. The model is trained using Adam optimizer [25], and negative log-likelihood loss function is used. Each pair of training example is associated with the ground truth from our dataset described in the previous section. We rank the candidates by the match probability for evaluation. As a baseline, we compare our MLP with a simple LR model over the exact same input vectors.

3.4 Infrastructure

3.4.1 Knowledge Graphs Store

We use Apache Jena[1] as an RDF store for efficiently querying the knowledge graph using SPARQL when needed. We also deployed Apache Jena Fuseki[2] for exposing SPARQL endpoints via REST API. We deployed services for DBpedia and Wikidata.

3.4.2 Labels Index

In our framework, we need to be able to quickly get all the entities URIs of a certain label. This query can be processed in SPARQL by it takes a lot of time. To be able to get the URIs instantly, we created labels index using sqlitedict[3], a persistent dictionary backed by sqlite3 and pickle, where each key is a label and the value is a list of entities URIs that have that label. We created two indices for DBpedia and Wikidata.

3.4.3 Embeddings Index

Since the number of the entities in both DBpedia and Wikipedia is huge, loading the embeddings of all entities in the knowledge graphs in memory is inefficient. Therfore, we use sqlitedict as a persistent key-value store for quickly accessing embeddings when needed. We created two indices for DBpedia and Wikidata.

3.5 Experiments

We evaluate our framework on two large-scale datasets sampled from DBpedia and Wikidata. One has DBpedia as the source knowledge graph and Wikidata as the target knowledge graph and vice versa. Table 3.2 shows the number of entities in each dataset. The percentage of ambiguous entities in the dataset is shown in Table 3.1.

We evaluate our end-to-end framework on both datasets. The Top 1 accuracy of the entity matching is shown in Table 3.3 and Table 3.5. MRR is shown in Table 3.6 and Table 3.6.

[1] https://jena.apache.org/
[2] https://jena.apache.org/documentation/fuseki2/
[3] https://github.com/RaRe-Technologies/sqlitedict

Dataset	Ambiguous Entities (%)
DB to WD	34%
WD to DB	33%

Table 3.1: Dataset Statistics

Dataset	Number of Entities
DB to WD	112,820
WD to DB	98,797

Table 3.2: Dataset Ambiguity

It is clear that the framework is effective in matching and disambiguating entities without the need of any manual annotation. The framework exploit the unambiguous entities in the knowledge graphs as training data in a weakly supervised fashion to train the graph embedding-based classifier to disambiguate the other ambiguous entities.

Model	**Unambiguous**	**Ambiguous**	**All**
Fuzzy	0.99	0.299	0.767
LR	-	0.291	-
MLP	-	0.714	-
Fuzzy + LR	-	-	0.763
Fuzzy + MLP	-	-	0.904

Table 3.3: Top-1 Accuracy on DBpedia to Wikidata Dataset.

3.6 Ambiguous DBpedia-Wikidata Dataset

Since ambiguity is the biggest challenge for entity matching problem, we create a benchmark dataset that focuses on ambiguous entities in DBpedia and Wikidata. That is, for each entity in the dataset, there exists more than one entity in the target knowledge graph that has exactly the same label of the entity to be mapped.

We then show the performance of our entity matching approach on the benchmark and we study the effects of having more ambiguity in terms of the number of possible candidates and the effects of changing the number of manual annotations used in training.

Model	Ambiguous	All
LR	0.523	-
MLP	0.820	-
Fuzzy + LR	-	0.841
Fuzzy + MLP	-	0.940

Table 3.4: MRR Accuracy on DBpedia to Wikidata Dataset.

Model	Unambiguous	Ambiguous	All
Fuzzy	0.98	0.311	0.768
LR	-	0.334	-
MLP	-	0.701	-
Fuzzy + LR	-	-	0.777
Fuzzy + MLP	-	-	0.900

Table 3.5: Top-1 Accuracy on Wikidata to DBpedia Dataset.

3.6.1 Dataset Construction

To further evaluate our framework, we created a benchmark dataset exploiting `OWL:sameAs` predicates that link entities between DBpedia (2016-10)[4] and Wikidata (2018-10-29)[5]. These predicates are manually curated and can be viewed as high quality ground truth. The total number of mappings we obtained by querying DBpedia and Wikidata using SPARQL was 6,974,651. We removed all mappings referring to Wikipedia disambiguation pages.

Although in principle entities with different labels in two knowledge graphs may refer to the same real-world entity, we wish to focus on the ambiguity problem and hence restrict our consideration to entities in knowledge graphs that share the same label—more precisely, the `foaf:name` predicate in DBpedia and the `rdfs:label` predicate in Wikidata. Furthermore, to make our task more challenging, we only consider ambiguous cases. To accomplish this, we first built two inverted indexes of the labels of all entities of DBpedia and Wikidata. Our problem formulation leads to the construction of two datasets, each corresponding to name matching in each direction:

[4] https://wiki.dbpedia.org/downloads-2016-10

[5] https://dumps.wikimedia.org/wikidatawiki/entities/20181029/

Model	Ambiguous	All
LR	0.560	-
MLP	0.815	-
Fuzzy + LR	-	0.853
Fuzzy + MLP	-	0.938

Table 3.6: MRR Accuracy on Wikidata to DBpedia Dataset.

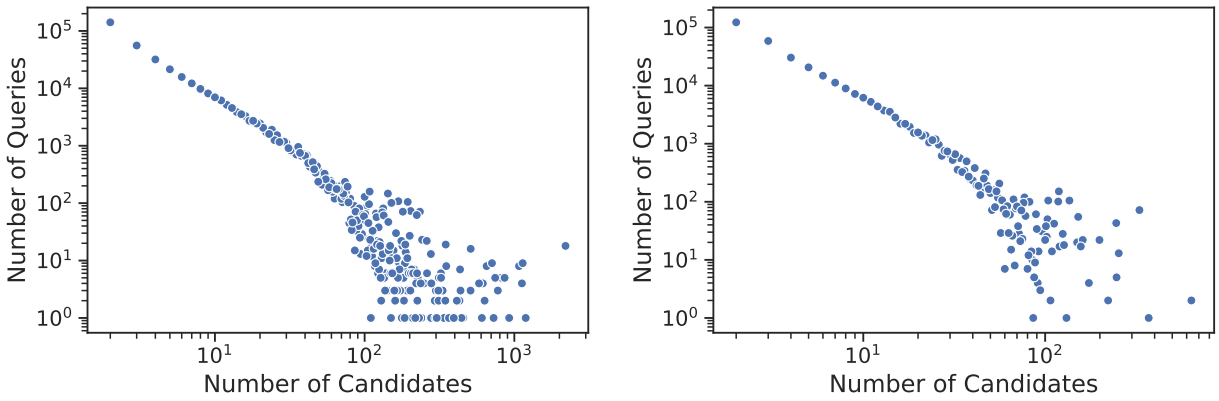
DBpedia to Wikidata Dataset

Here, we take DBpedia as the source knowledge graph and Wikidata as the target. For each entity in DBpedia, we queried the above index to retrieve entities with the same name in Wikidata, which forms a candidate set for disambiguation. Our focus is on ambiguous entities, and so we discard source DBpedia entities in which there is only one entity with the same name in Wikidata. For example, there are several people with the name "John Burt": John Burt (footballer), John Burt (rugby union), John Burt (anti-abortion activist), and John Burt (field hockey). Of these, only one choice is correct, which is determined by the `owl:sameAs` predicate: this provides our positive ground truth label. Thus, in each candidate set there is *only* one positive candidate and many negative candidates. The dataset contains 376,065 unique DBpedia URIs comprising the queries with a total of 232,757 unique names, and 967,937 unique Wikidata URIs as candidates.

Wikidata to DBpedia Dataset

We can apply exactly the same procedure as above to build a dataset with DBpedia as the source and Wikidata as the target. The resulting dataset contains 329,320 unique Wikidata URIs comprising the queries with a total of 293,712 unique names and 523,517 unique DBpedia URIs as candidates.

Finally, we shuffled and split the data into training, validation, and test set with a ratio of 70%, 10%, and 20% respectively. The dataset statistics are summarized in Table 3.7. Figure 3.2 shows the number of query (source) entities with different numbers of target entities. We see Zipf-like distribution: although most query entities have only a moderate number of possible candidates (e.g., less than 10), there exist outliers with hundreds or even more candidates. Note that by construction, our name matching problem *cannot* be solved by any more NLP techniques on text alone, since the source entities and target



(a) DBpedia to Wikidata

(b) Wikidata to DBpedia

Figure 3.2: Log-log plot showing the number of candidate URIs for each corresponding query URI

Dataset	Training	Validation	Testing	**Total**
DB to WD	263,245	37,607	75,213	**376,065**
WD to DB	230,523	32,933	658,64	**329,320**

Table 3.7: Dataset Statistics

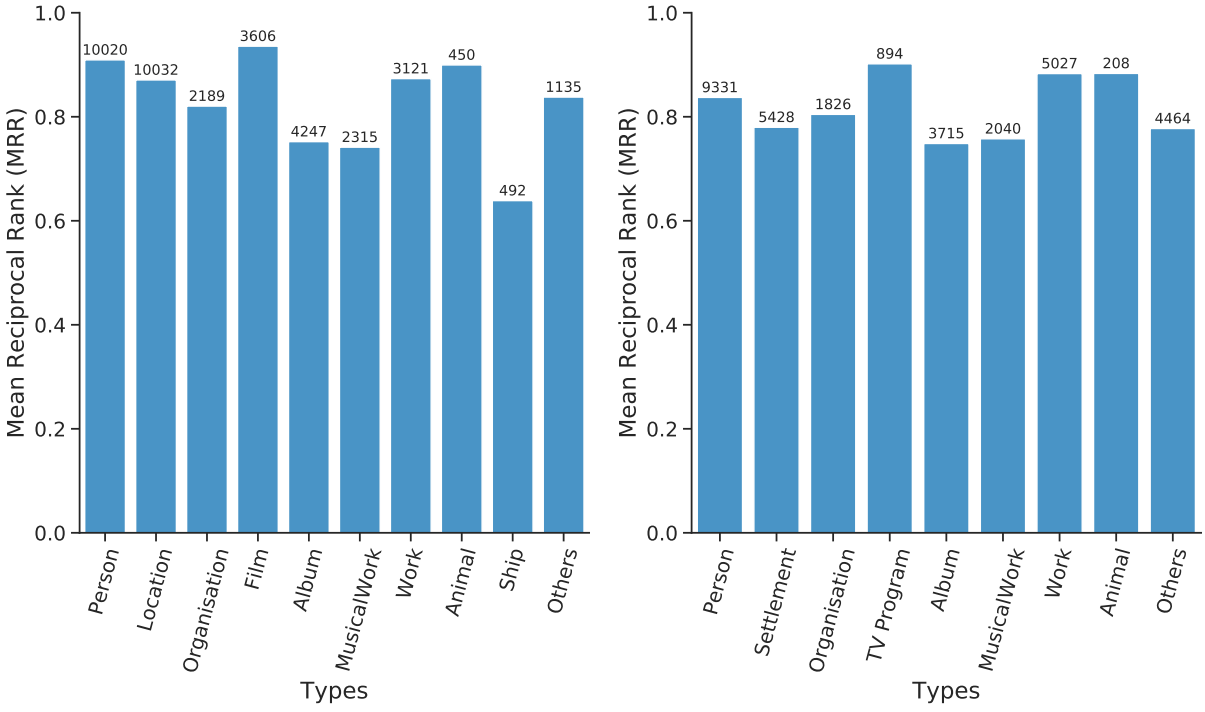
entities share exactly the same name. Thus, the context for disambiguation must come from a non-text source.

3.6.2 Experimental Evaluation

We use the benchmark introduced in Section 3.6 to evaluate our model. For each query in the dataset, we have one positive candidate and several negative candidates. Entities in each of DBpedia and Wikidata are embedded independently. As a result, each entity has two different embeddings, one in each knowledge graph. We use pretrained embeddings[6] that used $k = 200$ walks with depth $l = 4$ as hyper-parameters. The embeddings were trained using the skip-gram model with $d = 500$ which showed better results in [37]. If an entity has no pretrained embedding, a randomly-initialized embedding is used.

We evaluate the model on the test set with the best configuration tuned on the validation set, using the entire training set, The MLP hidden layer has size 750, with a learning rate of 0.001 and a batch size of 1024. MRR is used for evaluation.

[6] http://data.dws.informatik.uni-mannheim.de/rdf2vec/



(a) DBpedia to Wikidata

(b) Wikidata to DBpedia

Figure 3.3: MRR on validation set for different types of queries. Number on top of each bar indicates the size of examples

Overall, on test set, MLP model achieves 0.85 MRR mapping DBpedia entities to Wikidata and 0.81 MRR mapping Wikidata entities to DBpedia. In comparison, without the non-linear feature transformation,[7] a linear model like LR fails to learn a good decision boundary and achieves only 0.64 MRR and 0.62 MRR, respectively. Note that the embeddings of each knowledge graph are learned separately, which means that our model is *not* simply learning to match words in semantic relations—but actually learning correspondences between two semantic spaces. For reference, a random guessing baseline yields 0.25 MRR and 0.32 MRR, respectively.

Figure 3.3 breaks down performance according to entity type. We observe that types *Album* and *MusicalWork* yield worse results than the others, primarily there are greater ambiguity—these two types have larger candidates sizes, averaging 12.1 and 11.0 respectively, compared to persons, whose average candidates size is only 6.5.

Based on error analysis, we observe that our model lacks the fine-grained ability to disambiguate entities in the same type/domain in some case.

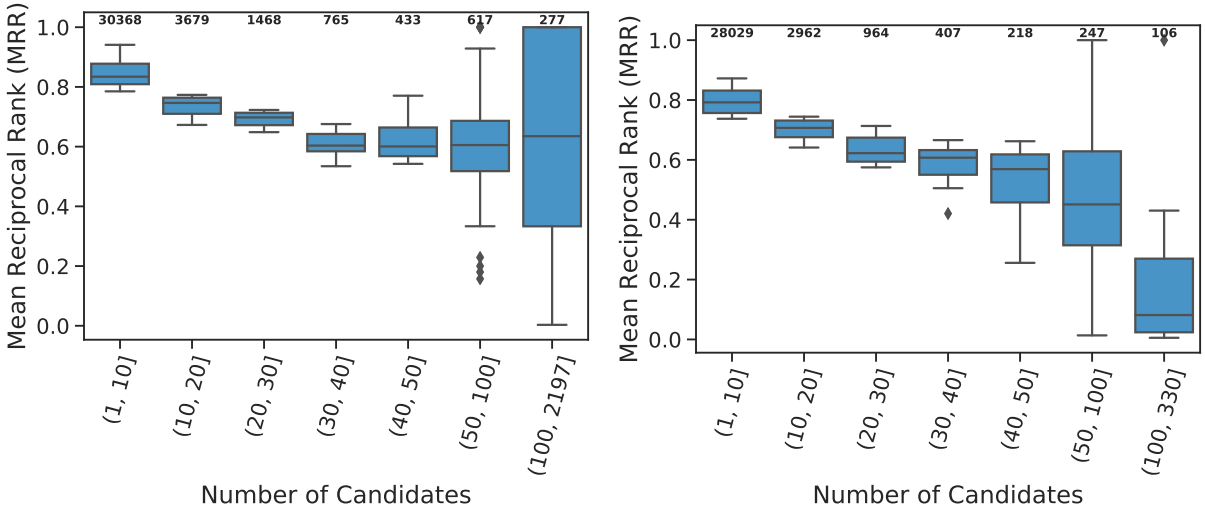
For example, in *music* domain, our model cannot differentiate the record company *Sunday Best* and the single *Sunday Best* by Megan Washington. Overall, for cases where the model fails to place the correct entities at rank one but succeeds at second rank instead, 79.5% of them have same type entities in these two positions in the validation set of DBpedia-to-Wikidata dataset.

For example, given a query entity *Sunday_Best_(song)* from DBpedia, which is a song by Australian musician Washington, the model ranked *Q7639339*, a British record company, in the first position, and ranked correct entity *Q7639338*, a single by Megan Washington, in second position.

Effects of increasing number of candidates: In our next analysis, we investigate the effects of the candidates size against the effectiveness. We measure the MRR of the MLP model on the validation set for different number of candidates. As shown in Figure 3.4, the MRR decreases for the queries that have large number of candidates. We calculate MRR for each unique candidate size and then use boxplot to aggregate MRR in each bin.

Effects of increasing training set size: Finally, we wanted to examine the effects of changing training set size. Figure 3.5 shows the effect of changing the size of the training set with percentages of $\{0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100\}$ on the MRR evaluated on the validation set. We sample from the training and validation set respectively while preserving

[7] A MLP can be regarded as a LR classifier where the input is transformed with non-linear function, which projects the feature into linear-separable.



(a) DBpedia to Wikidata

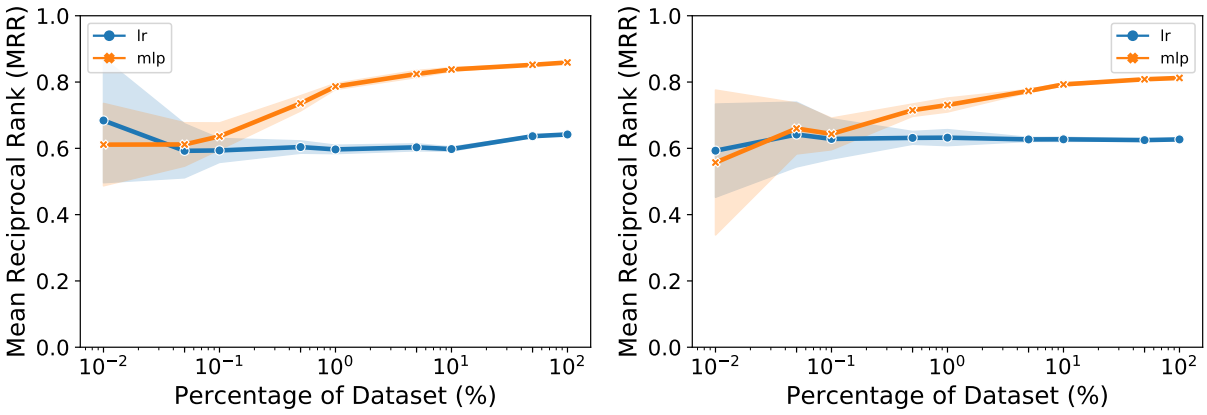
(b) Wikidata to DBpedia

Figure 3.4: MRR on validation set for different number of candidates (numbers on top are the total number of queries in each bin)

the 70:10 ratio. We repeat the sampling and run the models 10 times for the first 4 points and 5 times for the rest. 95% CI are shown in the graph in shaded colors to capture the variance of the samples. We observe that with a small fraction of the whole training set, the MLP model can achieve reasonable MRR. For example, with only 0.5% of the training set, the MLP model can achieve 0.74 MRR compared to 0.81 MRR on the whole training set.

3.7 Conclusion

We explore the problem of entity matching across knowledge graphs, sharing with the community two benchmark datasets. We introduces entity matching framework that combines a fuzzy string matcher with a graph embedding-based matcher for disambiguating ambiguous entities. Although the framework is quite simple, our models reveal some insights about the nature of this problem and paves the way for future work.



(a) DBpedia to Wikidata

(b) Wikidata to DBpedia

Figure 3.5: MRR when varying the size of the training set (x-axis is in log-scale)

Chapter 4

SimpleDBpediaQA: Dataset Migration with Manual Annotations

4.1 Introduction

Question answering over knowledge graphs is an important problem of interest both commercially and academically. There is substantial interest in the class of natural language questions that can be answered via the lookup of a single fact, driven by the availability of the popular SimpleQuestions dataset. The problem with this dataset, however, is that answer triples are provided from Freebase, which has been defunct for several years. As a result, it is difficult to build "real-world" question answering systems that are operationally deployable. Furthermore, a defunct knowledge graph means that much of the infrastructure for querying, browsing, and manipulating triples no longer exists. To address this problem, we present SimpleDBpediaQA [3], a new benchmark dataset for simple question answering over knowledge graphs that was created by mapping SimpleQuestions entities and predicates that comprise the answers to SimpleQuestions from Freebase to DBpedia. Unlike Freebase, DBpedia is actively maintained by a dedicated community.from Freebase to DBpedia.

Although this mapping is conceptually straightforward, there are a number of nuances that make the task non-trivial, owing to the different conceptual organizations of the two knowledge graphs.

We describe how this dataset migration is accomplished via high-quality alignments between entities in the two different knowledge graphs, and explain many of the nuances

Dataset	Training	Validation	Test	Total
SimpleQuestions	75,910	10,845	21,687	**108,442**
SimpleDBpediaQA	30,186	4,305	8,595	**43,086**

Table 4.1: Statistics of SimpleQuestions and SimpleDBpediaQA.

that make the creation of this dataset non-trivial. Our new dataset includes a total of 43,086 questions and corresponding answers that cover 40% of the original dataset. Summary statistics of SimpleDBpediaQA and SimpleQuestions are shown in Table 4.1. The complete dataset is available at https://github.com/castorini/SimpleDBpediaQA.

In addition to the contribution of providing the community with a new evaluation resource, we provide a series of simple yet strong baselines to lay the foundation for future work. These baselines include neural network models and other techniques that do not take advantage of neural networks, building on recently-published work [31]. An additional contribution of this thesis is that having two parallel datasets allows us to examine the effects of different conceptual organizations and knowledge graph structures: For example, we notice that many single-fact triples in Freebase require two-hop traversals in the DBpedia knowledge graph, which makes them no longer "simple" questions. Finally, evaluation resources targeting different conceptual organizations of knowledge help "keep researchers honest" in guarding against model overfitting on a single dataset.

4.2 Problem Definition

We begin with a formal definition of our problem. Let $E = \{e_1, e_2, \cdots, e_r\}$ be a set of entities, where e_i is a Uniform Resource Identifier (URI) uniquely identifying each entity. Let $P = \{p_1, p_2, \cdots, p_s\}$ be a set of predicates. Let $S \subseteq E$ be a set of subjects and $O \subseteq (L \cup E)$ be a set of objects, where L is a set of literals. In this context, $t = (s, p, o)$ denotes a RDF triple, comprised of a subject $s \in S$, a predicate $p \in P$, and an object $o \in O$.

Given this formalism, Freebase [10] represents a specific knowledge graph T^b, where $T^b = \{t_1^b, \cdots, t_m^b\}$ (i.e., a set of Freebase triples). Each Freebase entity is uniquely identified by a Mid (Machine ID). Similarly, DBpedia [8] represents another knowledge graph T^d, where $T^d = \{t_1^d, \cdots, t_n^d\}$.

The SimpleQuestions dataset is a collection of natural language questions and answers based on Freebase. Formally, $Q^b = \{q_1^b, \cdots, q_l^b\}$, where $q_i^b = (\mathcal{Q}_i, t_j^b)$; $\mathcal{Q}_i$ is a natural

language question and $t_j^b \in T^b$ is a Freebase triple that supplies the answer to that question.

For example, the question "Who wrote The New Canada?" has the following answer triple:

(fb:m/02qtvzv, fb:book/written_work/author, fb:m/01hxz2)

where fb stands for the prefix http://www.freebase.com/. The subject of the above answer triple is referred to as the topic entity, and the object of the triple is referred to as the answer entity. To answer the natural language question, a system must correctly identify the topic entity and the predicate, and then consult the knowledge graph to look up the answer entity.

Given Freebase T^b, DBpedia T^d, and SimpleQuestions Q^b, our problem can be formally defined as follows: for each $q_i^b = (\mathcal{Q}_i, t_j^b) \in Q^b$, find $q_i^d = (\mathcal{Q}_i, t_k^d)$, where $t_k^d \in T^d$ is a DBpedia triple, such that t_j^b is semantically equivalent to t_k^d. The result $Q^d = \{q_1^d, \cdots, q_l^d\}$ is our SimpleDBpediaQA dataset. Although this characterizes the basic structure of the problem, there are a number of nuances that deviate from this formalism, which we describe in the following sections.

4.3 Approach

Our overall strategy for dataset migration breaks down into the following steps: entity mapping, predicate mapping, and candidate refinement. At a high level, we begin by first mapping the topic and answer entities from Freebase to DBpedia; these then serve as anchors from which we can project the Freebase predicates to DBpedia. To assist in the process, we ingest the knowledge graphs into an RDF store to facilitate querying via SPARQL. For this effort, we use the latest version of DBpedia released in 2017[1].

4.3.1 Entity Mapping

The first step is to map Freebase entities from SimpleQuestions to entities in DBpedia. Freebase Mids and DBpedia URIs are linked through the predicate:

http://www.w3.org/2002/07/owl#sameAs;

[1] https://wiki.dbpedia.org/develop/datasets/dbpedia-version-2016-10

these official mappings are released as part of DBpedia[2]. For each Freebase entity MID (topic entity or answer entity), we issue a SPARQL query to retrieve the corresponding DBpedia URI. For example, Justin Trudeau, the current Prime Minister of Canada, is mapped via the triple:

(dbr:Justin_Trudeau, http://www.w3.org/2002/07/owl#sameAs, fb:m/02b5jh).

Here and throughout the thesis we use dbr as the DBpedia prefix for http://dbpedia.org/resource/. For approximately 56% of questions in SIMPLEQUESTIONS, we can map both the topic entity and the answer entity from Freebase to DBpedia. For the remaining questions, we are only able to map the topic entity, the answer entity, or neither. The detailed breakdowns are shown in Table 4.2.

Note that the URI for Justin Trudeau can be used to uniquely identify this entity within the broader open linked data ecosystem. For example, a human-readable version of facts associated with this individual is located at http://dbpedia.org/page/Justin_Trudeau. This, as well as a variety of other libraries, toolkits, APIs, etc. provide infrastructure that simplifies the development of operational question answering systems. The existence of these resources illustrates one of the major benefits of migrating SIMPLEQUESTIONS over to a knowledge graph that is actively maintained by a dedicated community.

4.3.2 Predicate Mapping

One-Hop Predicates

Let us consider the case where we are able to map both the topic entity and the answer entity from Freebase to DBpedia. We can then issue a SPARQL query over DBpedia to enumerate the paths (sequence of one or more predicates) connecting those entities. In the simplest case, there is a single predicate connecting the topic entity to the answer entity, which yields a straightforward mapping of the triple from Freebase to DBpedia. This occurs for approximately half of the questions with successfully mapped topic and answer entities; see detailed statistics in Table 4.2.

Consider the question "Which city is McCormick Field in?" The Freebase topic entity fb:m/05_xgn is mapped to DBpedia as dbr:McCormick_Field and the answer entity is mapped from fb:m/0ydpd to dbr:Asheville,_North_Carolina. The DBpedia predicate dbo:location connects those two entities, which provides a valid and correct mapping for the Freebase predicate

[2] http://downloads.dbpedia.org/2016-10/core-i18n/en/freebase_links_en.ttl.bz2

fb:location/location/containedby. Here and throughout the paper we use dbo as the DBpedia prefix for http://dbpedia.org/ontology/.

Due to differences in the conceptual organization of the two knowledge graphs, the directionality of equivalent predicates in Freebase and DBpedia may differ. For example, the DBpedia predicate dbo:birthPlace takes a person as the subject and a location as the object, whereas the equivalent predicate in Freebase fb:location/location/people_born_here inverts the subject and object. Therefore, for a question such as "Who was born in Aguascalientes?", the subject in the Freebase triple becomes the object in the DBpedia triple.

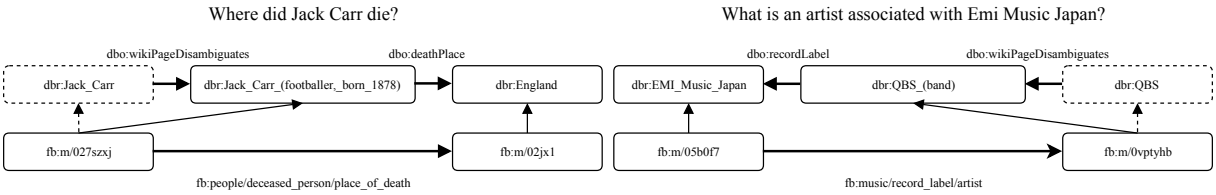
During the migration from Freebase to DBpedia, if the directionality of the mapped predicate is the same, we refer to the result as a forward predicate; if the directionality is reversed, we refer to the result as a backward predicate. We explicitly keep track of this metadata, which is necessary for the actual question answering task.

Two-Hop Predicates

Next, we consider the more complex case where the topic entity and the answer entity are *not* directly connected by a single predicate in DBpedia. That is, the results of our SPARQL query over DBpedia to enumerate the paths connecting the mapped entities contain multiple hops. In this work, we only consider two-hop traversals, as even longer paths are generally rare and spurious. These two-hop predicates can be categorized into the following:

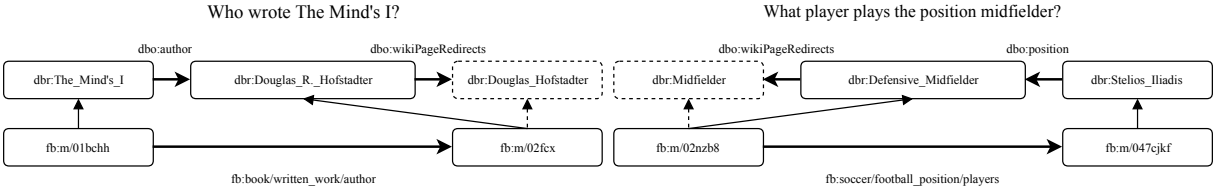
- Disambiguation Predicates
- Redirections Predicates
- Complex Predicates
- Missing Predicates

Disambiguation Predicates DBpedia uses wikiPageDisambiguates predicates to disambiguate different entities with the same name. The DBpedia sameAs links, however, might map a Freebase MID to an ambiguous URI, thus yielding a two-hop traversal from the topic entity to the answer entity. In these cases, we can "compress" the path back into a single predicate by changing the original topic entity to the disambiguated entity. Note that this disambiguation process can occur with forward predicates, as in Figure 4.1a, where dbr:Jack_Carr is disambiguated to dbr:Jack_Carr_(footballer,_born_1878), as well as backward predicates, as in Figure 4.1b, where dbr:QBS is disambiguated to dbr:QBS_(band).



(a) Forward Predicate (b) Backward Predicate

Figure 4.1: Examples of mapping disambiguation predicates from Freebase to DBpedia.

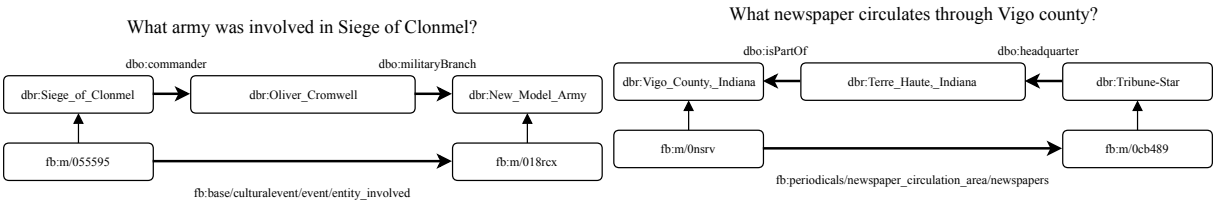


(a) Forward Predicate (b) Backward Predicate

Figure 4.2: Examples of mapping redirection predicates from Freebase to DBpedia.

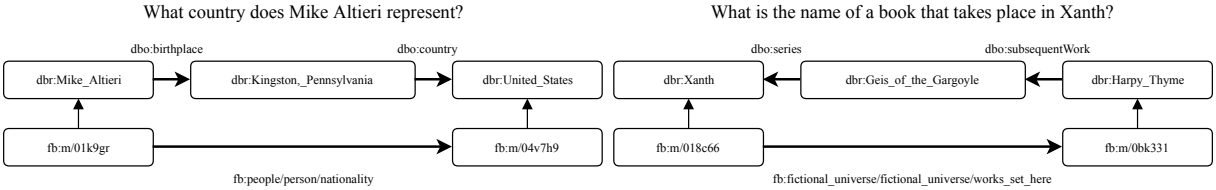
Redirections Predicates Similar to disambiguation links, DBpedia uses wikiPageRedirects predicates to redirect an entity to another entity (typically, the canonical variant of that entity). As with disambiguation predicates above, these two-hop redirection predicates can also be compressed into a single triple. For example, dbr:Douglas_Hofstadter is redirected back to dbr:Douglas_R._Hofstadter and dbr:Midfielder is redirected back to dbr:Defensive_Midfielder, as shown in Figure 4.2a and Figure 4.2b, respectively. Once again, this can occur with both forward and backward predicates.

Complex Predicates Due to differences in the conceptual organization of Freebase and DBpedia, there is no direct equivalent in DBpedia for some Freebase predicates. Instead, a chain of two predicates is necessary to capture the relationship between the topic and answer entities. An example is shown in Figure 4.3a: the question "What army was involved in Siege of Clonmel?" can be answered using the Freebase predicate fb:base/culturalevent/event/entity_involved, but in DBpedia the same fact requires a chain of two predicates, dbo:commander and dbo:militaryBranch. Note that this can also occur with backward predicates, as shown in Figure 4.3b.



(a) Forward Predicate (b) Backward Predicate

Figure 4.3: Examples of mapping complex predicates from Freebase to DBpedia.



(a) Forward Predicate (b) Backward Predicate

Figure 4.4: Examples of missing predicates in DBpedia.

Missing Predicates Some questions in DBpedia are answered using two-hop predicates even though there exists a one-hop predicate in the knowledge graph that represents a better mapping; this situation arises due to the incompleteness of DBpedia. Note that missing predicates actually represent a special case of complex predicates, which we only discovered by manual examination of the predicate mapping results.

Nevertheless, it seems appropriate to separately categorize this particular type of predicate mismatch between Freebase and DBpedia. An example is shown in Figure 4.4a: The entity dbr:Mike_Altieri should have a predicate dbo:nationality that directly links to dbr:United_States, as is typical of person entities.

However, since this predicate is missing, our SPARQL query discovered a roundabout path via dbo:birthplace then dbo:country. Figure 4.4b shows a similar case involving a backward predicate, where the entity dbr:Harpy_Thyme should have a predicate dbo:series that directly links to dbr:Xanth; instead, the answer entity is discovered via the extra hop dbo:subsequentWork. We believe that DBpedia can be enhanced by inserting these missing links, but augmenting DBpedia is beyond the scope of this work.

Detailed statistics of these two-hop predicate matches are shown in Table 4.2. As described above, there is no automatic way to differentiate between complex and missing

predicates, and thus we provide the sum of the two categories. For questions that have both topic and answer entity mappings, we are not able to find any predicate mappings for approximately 34% of them.

4.3.3 Candidates Refinement

The output of the initial entity and predicate mapping process (as described above) is then refined to produce the final SimpleDBpediaQA dataset; detailed statistics are shown in Table 4.3. In this section, we detail the candidate refinement process.

The need for post-processing candidate results from the output of the processes described above is apparent from manual examination. While the entity mappings are generally of high quality, some of the mapped predicates are invalid, primarily due to two reasons:

- *Semantic drift:* Some candidate predicates are not semantically correct even though the answer may be factually correct. For example, consider the question "From where does Anjali Devi claim nationality?" The predicate mapping produces dbo:deathPlace instead of the correct predicate, dbo:nationality. This is because the correct predicate is missing for this entity, and by coincidence, this person's nationality is the same as her death place.
- *Predicate constraints:* In some cases, we observe mismatches between the domains of the subjects or objects of a Freebase predicate and its corresponding DBpedia predicate. For example, the DBpedia predicate dbo:author can take as subject books, movies, etc. However, the Freebase predicate fb:book/author/works_written can only be mapped to the DBpedia predicate dbo:author (in the backward direction) if the DBpedia subject has the type dbo:WrittenWork. More generally, a predicate mapping is valid only under certain type constraints.

To tackle these challenges with minimal manual effort, we construct manual rules that map high-frequency Freebase predicates in the initial mappings to all potentially correct (at the semantic level) DBpedia predicates. Each rule includes a Freebase predicate and a list of corresponding DBpedia predicates, an associated directionality (forward or backward), and an optional type constraint. A few examples are shown in Table 4.4. The interannotator agreement of these rules based on three human annotators is 97%, where agreement is computed as the number of predicates that were identically labeled by all the annotators, divided by the count of all predicates.

				Training	Validation	Test	Total
Mapped Entities	Mapped Predicates	One Hop		22,158	3,193	6,304	**31,655**
		Two Hop	Disambiguation	173	24	59	**256**
			Redirection	1,992	247	553	**2,792**
			Complex + Missing	3,504	501	985	**4,990**
	Not Mapped Predicates			14,875	2,059	4,373	**21,307**
Sub-Total				**42,702**	**6,024**	**12,274**	**61,000**
Not Mapped Entities	Only Answer Entity Mapped			18,040	2,635	5,174	**25,849**
	Only Topic Entity Mapped			9,611	1,415	2,648	**13,674**
	Both Not Mapped			5,557	771	1,591	**7,919**
Sub-Total				**33,208**	**4,821**	**9,413**	**47,442**
SimpleQuestions **Total**				**75,910**	**10,845**	**21,687**	**108,442**

Table 4.2: Statistics from the initial mapping of entities and predicates in SimpleQuestions.

Using these mapping rules, we can filter and discard spurious one-hop mappings (including the disambiguation and redirection cases) where both the topic and answer entities are correctly mapped. Furthermore, we can expand the dataset by applying these rules to a few additional cases. Consider the case of complex and missing predicate: since these questions have two-hop predicates, making them no longer "simple questions", they would have been discarded from our dataset. However, we can issue a SPARQL query using the topic entity and the mapped DBpedia predicates from our mapping rules to search for valid answers (ignoring the answer entity). If the query returns a result, we can add the question to our dataset. Heuristically, this means that the question *does* have an answer in DBpedia, just not the same as the one provided in Freebase.

The same process can be applied to cases where we have successfully mapped the entities but not the predicates, and even to cases where we have only successfully mapped the topic entity. As a concrete example, for the question "What is a song by John Rutter?", only the topic entity is mapped. Based on our rules, the Freebase predicate fb:music/artist/track is mapped to the DBpedia predicate dbo:artist with a constraint of dbo:MusicalWork in the backward direction. Using the topic entity as an anchor, a SPARQL query returns a valid result.

Detailed statistics from the refinement process are shown in Table 4.3. The final output of entity mapping, predicate mapping, and candidate refinement is our SimpleDBpediaQA dataset, which successfully migrates SimpleQuestions from Freebase over to

				Training	Validation	Test	Total
Mapped Entities	Mapped Predicates	One Hop		19,271	2,773	5,467	**27,511**
		Two Hop	Disambiguation	84	17	32	**133**
			Redirection	1,547	191	429	**2,167**
			Complex + Missing	1,365	191	377	**1,933**
	Not Mapped Predicates			3,940	531	1,183	**5,654**
Sub-Total				**26,207**	**3,703**	**7,488**	**37,398**
Not Mapped Entities	Only Answer Entity Mapped			0	0	0	**0**
	Only Topic Entity Mapped			3,979	602	1,107	**5,688**
	Both Not Mapped			0	0	0	**0**
Sub-Total				**3,979**	**602**	**1,107**	**5,688**
SimpleDBpediaQA **Total**				**30,186**	**4,305**	**8,595**	**43,086**

Table 4.3: Final statistics of SimpleDBpediaQA following candidates refinement.

Freebase Predicate	DBpedia Predicate	Directionality	Type Constraint
fb:architecture/structure/architect	dbo:architect	forward	-
fb:location/location/contains	dbo:country	backward	-
fb:baseball/baseball_position/players	dbo:position	backward	dbo:BaseballPlayer
fb:music/album_release_type/albums	dbo:type	backward	dbo:Album
fb:book/author/works_written	dbo:author	backward	dbo:WrittenWork

Table 4.4: Examples of predicate mapping rules.

DBpedia.

4.4 Question Answering Model

As discussed in Section 2.5, we evaluate the question answering model on SimpleDBpediaQA and compare it to the original results on SimpleQuestions.

One additional detail is necessary to understand our experimental methodology for entity detection. In SimpleQuestions, the topic entity is not explicitly tagged in the natural language question at the token level; as a result, SimpleDBpediaQA does not have token-

Entity Linking	**R@1**	**R@5**
BiLSTM	78.0	88.2
CRF	75.1	85.4
Predicate Prediction	**R@1**	**R@5**
CNN	89.2	99.1
BiGRU	88.1	99.0
LR	84.2	97.6

Table 4.5: Component accuracy on validation set.

level annotations either. This presents a problem, as our formulation of entity detection as sequence labeling requires per-token labels. The solution adopted by Mohammed et al. [31] with SimpleQuestions was to "back-project" the entities onto the natural language questions to automatically derive token labels, either Entity or NotEntity. We performed exactly the same back-projection in this work. If the entity text can be matched exactly in the question, the corresponding tokens are tagged appropriately. If there is no exact match, n-grams are generated from the question (from length one up to the length of the question) and the Levenshtein Distances between these n-grams and the entity text are computed. The n-gram with the highest score is selected and the corresponding tokens are tagged appropriately. We find that 94.1% of questions have exact matches with entity strings.

4.5 Experiment Results

We evaluated the quality of our models in the same way as Mohammed et al. [31]: For entity detection, we compute F1 in terms of the entity labels. For both entity linking and predicate prediction, we evaluate recall at N (R@N). For the final end-to-end evaluation, we use accuracy (or equivalently, R@1). For evidence integration, our model considers 20 entity candidates and 5 predicate candidates. All hyper-parameters and other settings follow the original paper; we have not specifically fine-tuned parameters for this dataset.

For entity detection, on the validation set, the Bidirectional Long Short-Term Memory (BiLSTM) achieves 90.3 F1, compared to the CRF at 88.1. The top of Table 4.5 shows the entity linking results for the BiLSTM and the CRF. These results are consistent with the findings of Mohammed et al. [31]: the BiLSTM achieves a higher F1 score than the

Entity Detection	Relation Prediction	Accuracy
BiLSTM	CNN	78.5
BiLSTM	BiGRU	78.2
BiLSTM	LR	75.8
CRF	CNN	76.1
CRF	BiGRU	76.0
CRF	LR	73.5

Table 4.6: End-to-end accuracy on test set.

CRF, which translates into higher recall in entity linking (both R@1 and R@5). Predicate prediction results are shown on the bottom of Table 4.5: the CNN slightly outperforms the Bidirectional Gated Recurrent Unit (BiGRU) on R@1, but in terms of R@5 the accuracy of both are quite similar. The neural network models appear to be more effective than logistic regression.

Finally, Table 4.6 shows end-to-end accuracy on the test set. The best model combination uses the BiLSTM for entity detection and the CNN for predicate prediction, achieving 78.5% accuracy. By swapping the BiLSTM with the CRF for entity detection, we observe a 2.4% absolute decrease in end-to-end accuracy. Results from other combinations are also shown in Table 4.6. Note that using the CRF for entity detection and LR for predicate prediction, which is a baseline that does not use neural networks (with the exception of word embeddings), is also reasonably accurate. This finding is also consistent with Mohammed et al. [31], who advocate that Natural Language Processing (NLP) researchers examine baselines that do not involve neural networks as a sort of "sanity check".

4.6 Error Analysis

Following Lukovnikov et al. [28], we sampled 200 examples of errors on the test set from the BiLSTM + CNN model to analyze their causes. We manually classified them into the following categories, summarized in Table 4.7 and described below:

- *Hard ambiguity:* The context provided by the question is insufficient, even for a human, to disambiguate between two or more entities with the same name. In these cases, our model correctly identified the entity string, but linked it to an incorrect entity in the knowledge graph. For example, in the question "What is the place of birth

Error Type	# Errors	Prevalence
Hard ambiguity	42	21.0%
Soft ambiguity	21	10.5%
Entity detection error	19	9.5%
Predicate prediction error	28	14.0%
Error in both	90	45.0%
Total	200	100.0%

Table 4.7: Results of error analysis.

of Sam Edwards?", it is unclear if Sam Edwards refers to the actor dbr:Sam_Edwards or the physicist dbr:Sam_Edwards_(physicist).

- *Soft ambiguity:* The context provided by the question is sufficient to disambiguate the entity, but our model fails to identify the correct entity in the knowledge graph. In these cases, our model correctly identified the entity string, so the error is isolated to the entity linking component. For example, in the question "What kind of show is All In?", the model predicted dbr:All_In_(song) instead of dbr:All_In_(TV_series) (the correct entity). Note that in this case, it is clear to a human based on context that the question refers to a show and not a song.
- *Entity detection error:* The extracted entity string is incorrect.
- *Predicate prediction error:* The predicted predicate is incorrect.
- *Error in both:* Both the extracted entity and the predicted predicate are incorrect.

From the above analysis, we find that there is still substantial room to improve on the effectiveness of our baselines. However, these results also suggest that there is an upper bound on accuracy that lies substantially below 100%, as the cases of hard ambiguity are difficult to resolve, even for humans. In those cases, correct entity linking is more a matter of luck and other idiosyncratic characteristics of the dataset rather than signals that can be reliably extracted to understand the true question intent.

4.7 Conclusion

This chapter presents SimpleDBpediaQA, a new benchmark dataset for simple question answering over knowledge graphs created by migrating the SimpleQuestions dataset

from Freebase to DBpedia. Although this mapping process is conceptually straightforward, there are a number of nuances and complexities we had to overcome with a combination of special-case handling and heuristics. The result is a dataset targeting a knowledge graph that is actively maintained by a dedicated community. We hope that our efforts better connect existing research communities, in particular, NLP researchers with the open linked data community, and spur additional work in question answering over knowledge graphs.

Chapter 5

SimpleQuestions++: Dataset Migration without Manual Annotations

5.1 Introduction

As discussed in Section 2.5, SimpleQuestions++ dataset has emerged as the de facto benchmark for evaluating simple questions over knowledge graphs. However, SimpleQuestions dataset is solely based on Freebase, in other words, all entities in the dataset are only linked to Freebase.

Having parallel datasets allows us to examine the effects of different conceptual organizations and knowledge graph structures: For example, we notice that many single-fact triples in Freebase require two-hop traversals in the DBpedia knowledge graph, which makes them no longer "simple" questions. Furthermore, most question answering models over knowledge graphs that are released in research community are only benchmarked on datasets that are linked to a single knowledge graph.

There is a real need of benchmarking those models against datasets that are linked to multiple knowledge graphs that have different different conceptual organizations of knowledge, and different structure of graphs. These benchmarks can help "keep researchers honest" in guarding against model overfitting on a single dataset.

In this chapter, we describe SimpleQuestions++ dataset, an updated version of SimpleQuestions dataset, that has questions linked to Freebase, DBpedia, and Wiki-

data which can be used as the aforementioned benchmark. Moreover, we evaluate our entity mapping and disambiguation model on the DBpedia-Wikidata portion of the dataset and show how the model can be used to automatically migrate datasets from one knowledge graph to another. The complete dataset is available at https://github.com/MichaelAzmy/SimpleQuestionsQA.

5.2 Unified SimpleQuestions Dataset

5.2.1 Entity Mapping

We use the same approach in Section 4.3.1. We make use of http://www.w3.org/2002/07/owl#sameAs predicate that links knowledge graph URIs. We use the official mappings between DBpedia and Freebase that are released as part of DBpedia[1]. We also make use of the official mappings between Freebase and Wikidata that are released as part of Freebase[2]. In addition, we queried DBpedia SPARQL end-point to obtain mappings between DBpedia and Wikidata. The summary of the numbers of obtained mappings are shown in Table 5.1.

The goal is to map, to the best effort, entities of SimpleQuestions from Freebase to both DBpedia and Wikidata using the multiple mappings obtained.

Direct Mapping

The straightforward approach is to directly check if a given Freebase entity exists in the mappings of the target knoweldge graph. For example, when mapping entities from Freebase to DBpedia, we check the Freebase-DBpedia mappings to find a direct mapping. If a mapping exists, then the process is done. Otherwise we try transitive mapping.

Transitive Mapping

We make use of the transitive property of the mappings for all entities that were not found in the direct mappings. That is, if we are mapping an entity from Freebase to DBpedia, we check if that entity has a direct mapping between Freebase-Wikidata and Wikidata-DBpedia, where Wikidata here acts as a bridge between the source and the target knowledge graphs. This approach help find more mappings with minimal effort.

[1] http://downloads.dbpedia.org/2016-10/core-i18n/en/freebase_links_en.ttl.bz2
[2] https://developers.google.com/freebase/#freebase-wikidata-mappings

The entities that we didn't find a mapping for using both methods are declared as not mapped.

The summary of the statistics of the entity mapping of SimpleQuestions++ is shown in Table 5.2. The numbers in brackets are the total number of actual entities appearing in the original SimpleQuestions dataset. Since some entities can appear more than one time in different questions, we also show the numbers of unique entities. In addition, an entity can appear as a subject as well as an object, so we show both statistics.

DBpedia-Freebase	Freebase-Wikidata	DBpedia-Wikidata
4,389,741	6,974,361	20,967,457

Table 5.1: Statistics of mappings between Freebase, DBpedia, and Wikdiata.

Entity Ambiguity

We calcualte the number of ambiguous entities in both datasets. Table 5.3 and Table 5.4 shows the ratio of ambiguous entities in SimpleQuestions for both DBpedia to Wikidata and Wikidata to DBpedia, respectively. It is clear that Wikidata to DBpedia direction is more ambiguous, which will be reflected in a lower accuracy for the automatic mapper that we will discuss in detail in Section 5.3.

5.2.2 Predicate Mapping

After mapping entities of SimpleQuestions, the next step is mapping predicates between each subject and object. We follow the same procedure from Section 4.3.2. For questions that we were able to map both the subject and the object, We issue a SPARQL query over the target knowledge graph to enumerate the one-hop simple path connecting those entities. The summary of the statistics of the predicate mapping is shown in Table 5.5.

5.2.3 Final Dataset

After the predicate mapping phase, we have two individual datasets (i.e, Freebase to DBpedia and Freebase to Wikidata). Since the mapping is done individually, it is expected to find overlapping and non-overlapping questions in both datasets. Since our final goal is to

		Mapping	Training	Validation	Test	Total
DBpedia	Subjects	Direct	41,297 (52,313)	6,575 (7,439)	12,792 (14,922)	**60,664 (74,674)**
		Transitive	617 (985)	87 (138)	165 (250)	**869 (1,373)**
		Not Mapped	22,054 (22,612)	3,249 (3,268)	6,449 (6,515)	**31,752 (32,395)**
		Sub-Total	**63,968 (75,910)**	**9,910 (10,845)**	**19,406 (21,687)**	**93,285 (108,442)**
	Objects	Direct	25,053 (60,742)	4,767 (8,659)	8,803 (17,448)	**38,623 (86,849)**
		Transitive	315 (1,092)	52 (155)	91 (300)	**458 (1,547)**
		Not Mapped	12,027 (14,076)	1,848 (2,031)	3,515 (3,939)	**17,390 (20,046)**
		Sub-Total	**37,395 (75,910)**	**6,667 (10,845)**	**12,409 (21,687)**	**56,471 (108,442)**
Wikidata	Subjects	Direct	30,063 (40,378)	4,934 (5,775)	9,457 (11,502)	**44,454 (57,655)**
		Transitive	11,782 (12,852)	1,725 (1,797)	3,496 (3,668)	**17,003 (18,317)**
		Not Mapped	22,123 (22,680)	3,252 (3,273)	6,453 (6517)	**31,828 (32,470)**
		Sub-Total	**63,968 (75,910)**	**9,910 (10,845)**	**19,406 (21,687)**	**93,285 (108,442)**
	Objects	Direct	20,348 (52,957)	4,067 (7,588)	7,390 (15,200)	**31,805 (75,745)**
		Transitive	4,992 (8,830)	744 (1,222)	1503 (2,537)	**7,239 (12,589)**
		Not Mapped	12,055 (14,123)	1,856 (2,035)	3,516 (3,950)	**17,427 (20,108)**
		Sub-Total	**37,395 (75,910)**	**6,667 (10,845)**	**12,409 (21,687)**	**56,471 (108,442)**

Table 5.2: Statistics of entity mapping phase from Freebase to DBpedia and Wikidata in SimpleQuestions++ dataset. (The numbers without brackets are the unique number of entities)

get a unified dataset for Freebase, DBpedia and Wikidata, we have to get the overlapping questions across the mapped dataset. The statistics of the overlapped dataset is shown in Table 5.6. Furthermore, Table 5.7 shows the number of unique entities in the final unified dataset, and Table 5.8 shows the final number of questions in SimpleQuestions++ dataset.

5.2.4 Examples

Listing 1 shows an example of a question in the unified dataset. Each question has the subject and the object mapped to Freebase, DBpedia, and Wikidata. In addition, each question has a list of possible predicates for each knowledge graph. Listing 2 shows an example of a question having three Wikidata predicates that can answer the question.

	Training	Validation	Test
Ambiguity	0.425	0.449	0.432

Table 5.3: Ambiguity of entities in SimpleQuestions++ dataset (DBpedia to Wikidata).

	Training	Validation	Test
Ambiguity	0.570	0.564	0.589

Table 5.4: Ambiguity of entities in SimpleQuestions++ dataset (Wikidata to DBpedia).

5.3 Automatic Entity Mapping

Now that we have a high-quality unified dataset, SimpleQuestions++, we show the effect of using our proposed entity matching approach discussed in Chapter 3 onautomatically mapping SimpleQuestions++ dataset. The unified dataset acts as ground truth.

We use the same formulation introduced in Section 4.2. Let DB denote DBpedia knowledge graph and WD denote Wikidata knowledge graph. Let Q^{DB}, Q^{WD} denote the ground truth DBpedia and Wikidata datasets in SimpleQuestions++, respectively.

Since we have questions linked to both DBpedia and Wikidata, we use our approach to automatically map the dataset in both directions without using any manual annotations (i.e without levaraging any *SameAs* predicates). We assume that predicates are already mapped and focus on entity mapping using our framework. This is a valid assumption because predicates always exist in a small number and can always be manually mapped.

Generating Training Data

First, we use the fuzzy string matcher to map entities from source knowledge graph to target knowledge graph. For each entity in source knowledge graph, we use the inverted labels index to get the exactly matching entities from the target knowledge graph. The output of this stage are two collections as follows:

- *Unambiguous entities:* entities in source knowledge graph that has only one exact mapping in target knowledge graph.
- *Ambiguous entities:* entities in source knowledge graph that has more than one mapping in target knowledge graph.

			Training	Validation	Test	Total
DBpedia	Mapped Entities	Mapped Predicates	20,119	2,893	5,755	**28,767**
		Not Mapped Predicates	24,309	3,376	6,973	**34,658**
		Sub-Total	**44,428**	**6,269**	**12,728**	**63,425**
	Not Mapped Entities	Only Answer Entity Mapped	17,406	2,545	5,020	**24,971**
		Only Topic Entity Mapped	8,870	1,308	2,444	**12,622**
		Both Not Mapped	5,206	723	1,495	**7,424**
		Sub-Total	**31,482**	**4,576**	**8959**	**45,017**
	Total		**75,910**	**10,845**	**21687**	**108,442**
Wikidata	Mapped Entities	Mapped Predicates	22,420	3,174	6,380	**31,974**
		Not Mapped Predicates	21,897	3,086	6,340	**31,323**
		Sub-Total	**44,317**	**6,260**	**12,720**	**63,297**
	Not Mapped Entities	Only Answer Entity Mapped	17,470	2,550	5,017	**25,037**
		Only Topic Entity Mapped	8,913	1,312	2,450	**12,675**
		Both Not Mapped	5,210	723	1,500	**7,433**
		Sub-Total	**31,593**	**4,585**	**8,967**	**45,145**
	Total		**75,910**	**10,845**	**21687**	**108,442**

Table 5.5: Statistics of predicate mapping phase from Freebase to DBpedia and Wikidata in SimpleQuestions++ dataset.

The first set of entities is trivially mapped (exactly one mapping exists so there is no ambuiguity). Consequently, we use all of the unambiguous entities as training examples for the graph embedding-based matcher.

5.3.1 DBpedia to Wikidata

In this section, we consider DB and WD as our source knowledge graph and target knowledge graph, respectively. We use Q^{DB} as our source dataset and the goal is to automatically map the questions of Q^{DB} that are linked to DB to $Q^{WD'}$ that is linked to WD.

As per our formulation, we first need to map entities and then issue the SPARQL query

		Training	Validation	Test	Total
Mapped Entities	Mapped Predicates	13,665	1,947	3,877	**19,489**
	Not Mapped Predicates	15,284	2,114	4,418	**21,816**
	Sub-Total	**28,949**	**4,061**	**8,295**	**41,305**
Not Mapped Entities	Only Answer Entity Mapped	17,247	2,522	4965	**24,734**
	Only Topic Entity Mapped	8,750	1,290	2,408	**12,448**
	Both Not Mapped	5,178	720	1,490	**7,388**
	Sub-Total	**31,175**	**4,532**	**8,863**	**44,570**
Total		**60,124**	**8,593**	**17,158**	**85,875**

Table 5.6: Statistics of overlapping questions between DBpedia and Wikidata mappings in SimpleQuestions++.

to get the predicate. Obviously, if any of the subject or the object of a given question are incorrectly mapped, the question will be incorrect and might negatively affect the question answering model.

Entity Mapping

After retrieving the training set as described in the previous section, we train the graph embedding-based matcher and then use the trained model to map the ambiguous entities.

Figure 5.1 shows the Top-1 accuracy of the model on the ambiguous entities, and Figure 5.2 shows the MRR of the model on the ambiguous entities. Table 5.9 shows the top 1 accuracy for the end-to-end entity matching framework. Fuzzy string matcher is used for unambiguous cases and graph embedding-based matcher is used for ambiguous cases. It is clear that using embeddings for disambiuguation boosts the performance of the matcher. The fuzzy string matcher can only map $\sim 25\%$ of the ambiguous entities yielding an overall accuracy of $\sim 66\%$, however, when it is combined with the embedding-based matcher, it achieves $\sim 84\%$. Table 5.10 shows the final statistics of the automatically-mapped dataset.

		Training	Validation	Test
Freebase	Unique Topic Entities	12,153	1,847	3,598
	Unique Answer Entities	8,487	1,486	2,779
	Overlap Topic and Answer Entities	995	92	264
	Overall Unique Entities	**19,645**	**3,241**	**6,113**
DBpedia	Unique Topic Entities	12,152	1,847	3,598
	Unique Answer Entities	8,487	1,486	2,779
	Overlap Topic and Answer Entities	997	92	264
	Overall Unique Entities	**19,642**	**3,241**	**6,113**
Wikidata	Unique Topic Entities	12,153	1,847	3,598
	Unique Answer Entities	8,487	1,486	2,779
	Overlap Topic and Answer Entities	996	92	264
	Overall Unique Entities	**19,644**	**3,241**	**6,113**

Table 5.7: Statistics of entities in SimpleQuestions++ dataset.

Dataset	Training	Validation	Test	Total
SimpleQuestions	75,910	10,845	21,687	**108,442**
SimpleQuestions++	13,665	1,947	3,877	**19,489**

Table 5.8: Statistics of SimpleQuestions and SimpleQuestions++.

5.3.2 Wikidata to DBpedia

In this section, we consider WD and DB as our source knowledge graph and target knowledge graph, respectively. We use Q^{WD} as our source dataset and the goal is to automatically map the questions of Q^{WD} that are linked to WD to $Q^{DB'}$ that is linked to DB.

Similarly, we first need to map entities and then issue the SPARQL query to get the predicate. Obviously, if any of the subject or the object of a given question are incorrectly mapped, the question will be incorrect and might negatively affect the question answering model.

Model	Unambiguous	Ambiguous	All
Fuzzy	0.982	0.254	0.697
LR	-	0.567	-
MLP	-	0.612	-
Fuzzy + LR	-	-	0.820
Fuzzy + MLP	-	-	0.837

Table 5.9: Top-1 Accuracy on SimpleQuestions++ dataset (DBpedia to Wikidata).

	Training	Validation	Test
Both Subject and Object Mapped	4,688	687	1,340
Only Subject Mapped	4,150	604	1,176
Only Object Mapped	2,452	331	715
Both Subject and Object Not Mapped	1,659	228	441
Total Questions	**12,949**	**1,850**	**3,672**

Table 5.10: Statistics of automatically mapped entities in SimpleQuestions++ dataset (DBpedia to Wikidata).

Entity Mapping

After retrieving the training set as described in the previous section, we train the graph embedding-based matcher and then use the trained model to map the ambiguous entities.

Figure 5.3 shows the Top-1 accuracy of the model on the ambiguous entities, and Figure 5.4 shows the MRR of the model on the ambiguous entities.

Table 5.11 shows the top 1 accuracy for the end-to-end entity matching framework. Fuzzy string matcher is used for unambiguous cases and graph embedding-based matcher is used for ambiguous cases. It is clear that using embeddings for disambiuguation boosts the performance of the matcher. The fuzzy string matcher can only map $\sim 56\%$ of the ambiguous entities yielding an overall accuracy of $\sim 72\%$, however, when it is combined with the embedding-based matcher, it achieves $\sim 80\%$. Table 5.12 shows the final statistics of the automatically-mapped dataset.

Model	Unambiguous	Ambiguous	All
Fuzzy	0.951	0.566	0.725
LR	-	0.650	-
MLP	-	0.709	-
Fuzzy + LR	-	-	0.775
Fuzzy + MLP	-	-	0.809

Table 5.11: Top-1 Accuracy on SimpleQuestions++ dataset (Wikidata to DBpedia).

	Training	Validation	Test
Both Subject and Object Mapped	5,145	744	1,435
Only Subject Mapped	3934	579	1,135
Only Object Mapped	2,379	326	687
Both Subject and Object Not Mapped	1,491	201	405
Total Questions	**12,949**	**1,850**	**3,672**

Table 5.12: Statistics of automatically mapped entities in SimpleQuestions++ dataset (Wikidata to DBpedia).

5.4 Conclusion

With exploring the problem of entity matching across knowledge graphs, sharing with the community a new large-scale benchmark and new baseline models which leverages the structure information from graph embedding, and offer detailed analysis on this problem at the same time. Although quite simple, our model reveals some insights about the nature of this problem and paves the way for future work.

```
{
  "ID": "00007",
  "Question": "what movie is produced by warner bros.",
  "Subject": {
    "Freebase": "www.freebase.com/m/086k8",
    "DBpedia": "http://dbpedia.org/resource/Warner_Bros.",
    "Wikidata": "http://www.wikidata.org/entity/Q126399"
  },
  "Object": {
    "Freebase": "www.freebase.com/m/0278x5r",
    "DBpedia": "http://dbpedia.org/resource/Saving_Shiloh",
    "Wikidata": "http://www.wikidata.org/entity/Q7428297"
  },
  "Predicates": {
    "Freebase": [
      {
        "Predicate": "www.freebase.com/film/production_company/films",
        "Direction": "Forward"
      }
    ],
    "DBpedia": [
      {
        "Predicate": "http://dbpedia.org/ontology/distributor",
        "Direction": "Backward"
      }
    ],
    "Wikidata": [
      {
        "Predicate": "http://www.wikidata.org/prop/direct/P750",
        "Direction": "Backward"
      }
    ]
  }
}
```


Listing 1: Example of a single-predicate question in SimpleQuestions++

```
{
  "ID": "75905",
  "Question": "What airport is near the city elista",
  "Subject": {
    "Freebase": "www.freebase.com/m/02fd8s",
    "DBpedia": "http://dbpedia.org/resource/Elista",
    "Wikidata": "http://www.wikidata.org/entity/Q3977"
  },
  "Object": {
    "Freebase": "www.freebase.com/m/0273xkh",
    "DBpedia": "http://dbpedia.org/resource/Elista_Airport",
    "Wikidata": "http://www.wikidata.org/entity/Q2276543"
  },
  "Predicates": {
    "Freebase": [
      {
        "Predicate": "www.freebase.com/location/location/nearby_airports",
        "Direction": "Forward"
      }
    ],
    "DBpedia": [
      {
        "Predicate": "http://dbpedia.org/ontology/location",
        "Direction": "Backward"
      }
    ],
    "Wikidata": [
      {
        "Predicate": "http://www.wikidata.org/prop/direct/P138",
        "Direction": "Backward"
      },
      {
        "Predicate": "http://www.wikidata.org/prop/direct/P131",
        "Direction": "Backward"
      },
      {
        "Predicate": "http://www.wikidata.org/prop/direct/P931",
        "Direction": "Backward"
      }
    ]
  }
}
```


Listing 2: Example of a multi-predicate question in SimpleQuestions++

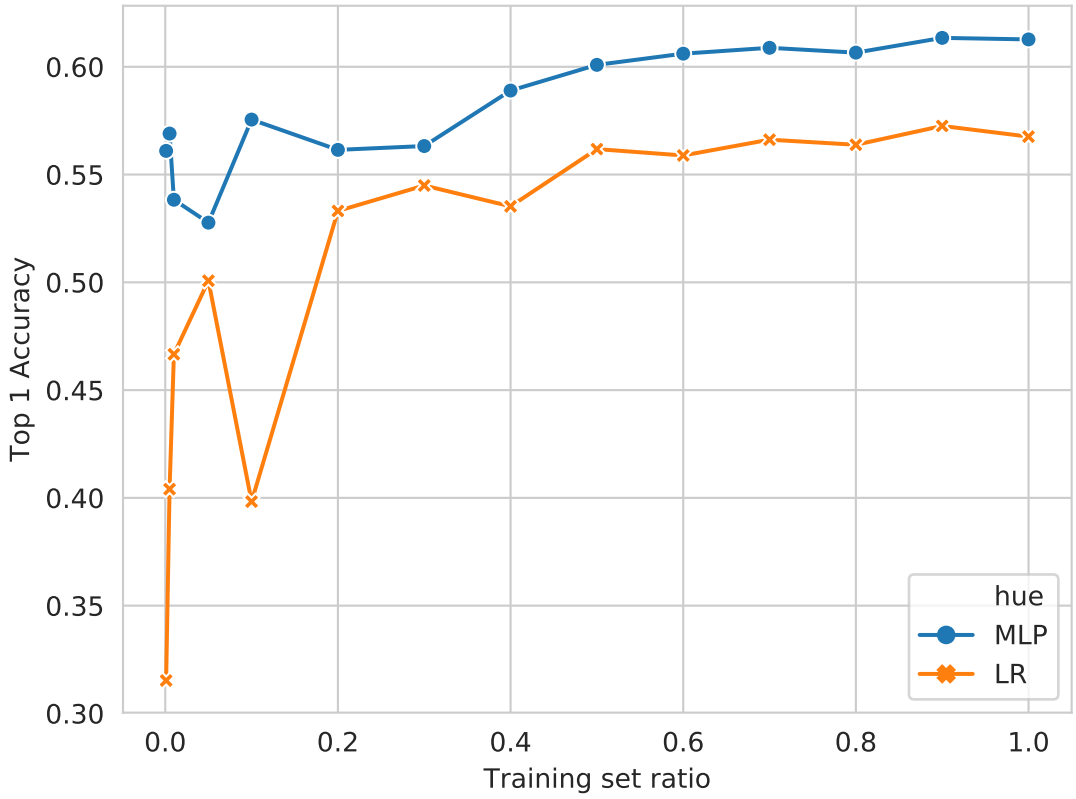


Figure 5.1: Top-1 Accuracy on ambiguous entities in SimpleQuestions++ dataset (DBpedia to Wikidata)

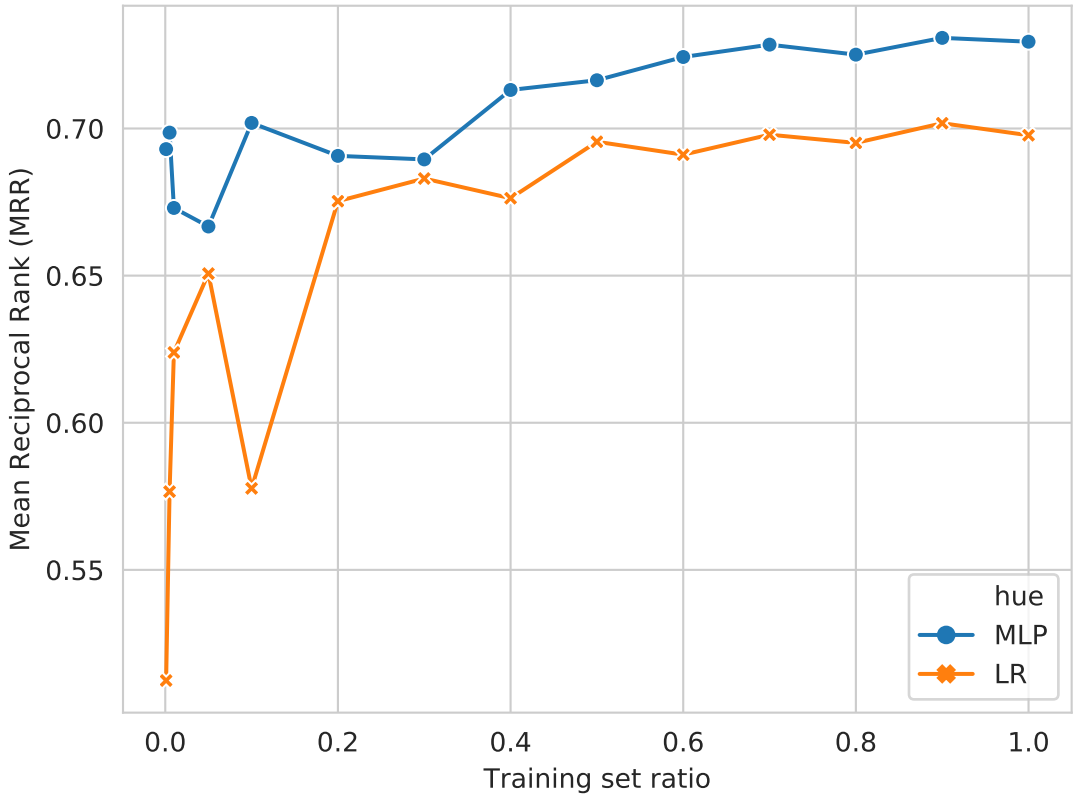


Figure 5.2: MRR on ambiguous entities in SimpleQuestions++ dataset (DBpedia to Wikidata)

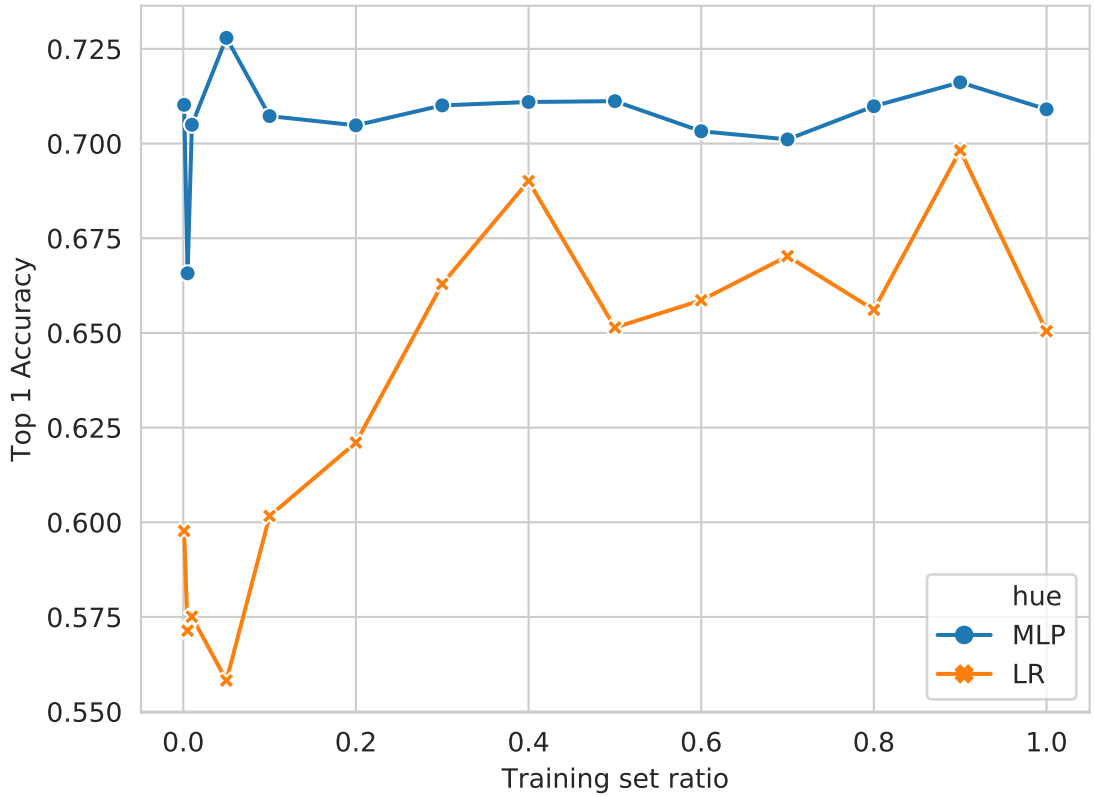


Figure 5.3: Top-1 Accuracy on ambiguous entities in SimpleQuestions++ dataset (Wikidata to DBpedia)

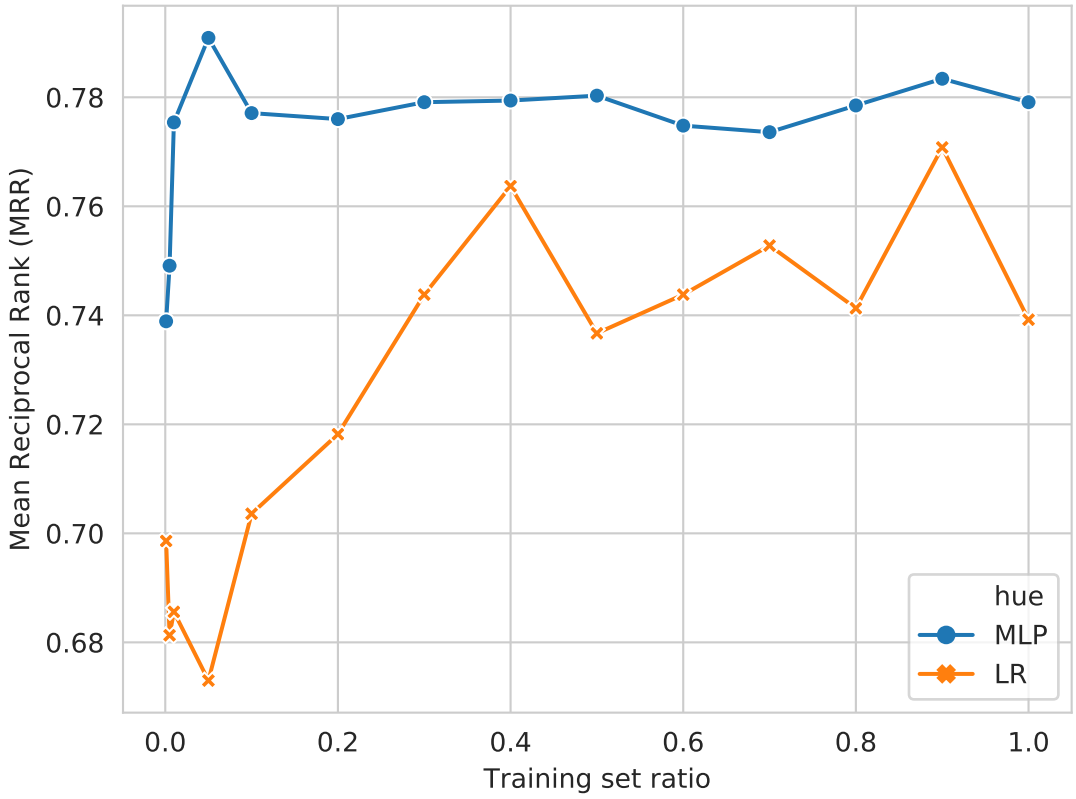


Figure 5.4: MRR on ambiguous entities in SimpleQuestions++ dataset (Wikidata to DBpedia)

Chapter 6

Conclusion

In this thesis, we proposed a novel framework for entity matching across multiple knowledge graphs. We combine a fuzzy string matcher along with a graph embedding-based model for disambiguating entities that have same labels. The proposed model achieves good results given its simplicity and it paves the way for future work to enhance its capability.

We also created a large-scale dataset for entity matching that focuses on ambiguous cases to make it easier for researchers to evaluate their entity disambiguation models.

We presented SimpleDBpediaQA, a new benchmark dataset for simple question answering over knowledge graphs created by migrating the SimpleQuestions dataset from Freebase to DBpedia.

Finally, we introduced SimpleQuestions++ , a unified simple questions dataset that includes questions from SimpleQuestions dataset along with the entity and predicate mapping in Freebase, DBpedia, and Wikidata. This benchmark can help evaluate the performance of question answering models against different knowledge graphs. We also evaluate the performance of our entity matching framework on automatically mapping SimpleQuestions++ dataset.

References

[1] Manel Achichi, Zohra Bellahsene, and Konstantin Todorov. Legato: Results for oaei 2017. In *OM-2017: Proceedings of the Twelfth International Workshop on Ontology Matching*, page 146, 2017.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[3] Michael Azmy, Peng Shi, Jimmy Lin, and Ihab Ilyas. Farewell freebase: Migrating the simplequestions dataset to dbpedia. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2093–2103, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[4] Michael Azmy, Peng Shi, Jimmy Lin, and Ihab F Ilyas. Matching entities across different knowledge graphs with graph embeddings. *arXiv preprint arXiv:1903.06607*, 2019.

[5] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question–answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1533–1544, Seattle, Washington, 2013.

[6] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

[7] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI Global, 2011.

[8] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia — A crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.

[9] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

[10] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1249, Vancouver, British Columbia, Canada, 2008.

[11] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv:1506.02075v1*, 2015.

[12] Hongyun Cai, Vincent W Zheng, and Kevin Chang. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[13] Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433, Sofia, Bulgaria, 2013.

[14] Silvana Castano, Alfio Ferrara, Stefano Montanelli, and Gaia Varese. Ontology and instance matching. In *Knowledge-driven multimedia information extraction and ontology evolution*, pages 167–195. Springer, 2011.

[15] Zihang Dai, Lei Li, and Wei Xu. CFO: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 800–810, Berlin, Germany, 2016.

[16] Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2014)*, pages 365–374, Gold Coast, Australia, 2014.

[17] Gerben Klaas Dirk De Vries and Steven de Rooij. Substructure counting graph kernels for machine learning from rdf data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 35:71–84, 2015.

[18] Gerben Klaas Dirk De Vries, Steven De Rooij, et al. A fast and simple graph kernel for rdf. *DMoLD*, 1082, 2013.

[19] Dennis Diefenbach, Thomas Pellissier Tanon, Kamal Deep Singh, and Pierre Maret. Question answering benchmarks for Wikidata. In *Proceedings of the 2017 International Semantic Web Conference (Posters, Demos & Industry Tracks)*, 2017.

[20] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018.

[21] Michael Färber, Basil Ell, Carsten Menne, and Achim Rettinger. A Comparative Survey of DBpedia , Freebase,OpenCyc,Wikidata,And YAGO. *Semantic Web*, 1:1–5, 2015.

[22] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Ann Arbor, Michigan, 2005.

[23] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[24] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer, 2011.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Pradap Konda, Sanjib Das, Paul Suganthan GC, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, et al. Magellan: Toward building entity matching management systems. *Proceedings of the VLDB Endowment*, 9(12):1197–1208, 2016.

[27] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sren

Auer, and Christian Bizer. DBpedia A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web*, 1:1–5, 2012.

[28] Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1211–1220, Perth, Australia, 2017.

[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[31] Salman Mohammed, Peng Shi, and Jimmy Lin. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 291–296, New Orleans, Louisiana, 2018.

[32] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34. ACM, 2018.

[33] Christopher Olahn. Deep learning, nlp, and representations. http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/, 2014.

[34] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

[35] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. In *Proceedings of the 25th international conference on world wide web*, pages 1419–1428. International World Wide Web Conferences Steering Committee, 2016.

[36] Petar Ristoski. *Exploiting semantic web knowledge graphs in data mining.* PhD thesis, University of Mannheim, 2018.

[37] Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *International Semantic Web Conference*, pages 498–514. Springer, 2016.

[38] Tzanina Saveta, Evangelia Daskalaki, Giorgos Flouris, Irini Fundulaki, Melanie Herschel, and Axel-Cyrille Ngonga Ngomo. Pushing the limits of instance matching systems: A semantics-aware benchmark for linked data. In *Proceedings of the 24th International Conference on World Wide Web*, pages 105–106. ACM, 2015.

[39] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. Adoption of the linked data best practices in different topical domains. In *International Semantic Web Conference*, pages 245–260. Springer, 2014.

[40] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

[41] Kuldeep Singh, Arun Sethupat Radhakrishna, Andreas Both, Saeedeh Shekarpour, Ioanna Lytra, Ricardo Usbeck, Akhilesh Vyas, Akmal Khikmatullaev, Dharmen Punjani, Christoph Lange, Maria Esther Vidal, Jens Lehmann, and Sören Auer. Why reinvent the wheel: Let's build question answering systems together. In *Proceedings of the 2018 World Wide Web Conference*, pages 1247–1256, Lyon, France, 2018.

[42] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[43] Thomas Pellissier Tanon, Denny Vrande, San Francisco, Sebastian Schaffert, and Thomas Steiner. From Freebase to Wikidata : The Great Migration. *Proceedings of the 25th International Conference on World Wide Web*, pages 1419–1428, 2016.

[44] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. LC-QuAD: A corpus for complex question answering over knowledge graphs. In *Proceedings of the 16th International Semantic Web Conference*, pages 210–218, Vienna, Austria, 2017.

[45] Ferhan Ture and Oliver Jojic. No need to pay attention: Simple recurrent neural networks work! (for answering "simple" questions). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 2856–2862, Copenhagen, Denmark, 2017.

[46] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[47] Xuchen Yao. Lean question answering over Freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations (NAACL/HLT 2015)*, pages 66–70, Denver, Colorado, 2015.