B.Sc. in Computer Science and Engineering Thesis

Building A Knowledge Graph Using Twitter Data

Submitted by

Sukanta Kundu
201205068

Anindya Biswas
201405006

Qazi Fahim Farhan
201405052

Supervised by

Dr. Muhammad Masroor Ali

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

Dhaka, Bangladesh

February 2018

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis, titled, "Building A Knowledge Graph Using Twitter Data", is the outcome of the investigation and research carried out by us under the supervision of Dr. Muhammad Masroor Ali.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Sukanta Kundu
201205068

Anindya Biswas
201405006

Qazi Fahim Farhan
201405052

CERTIFICATION

This thesis titled, **"Building A Knowledge Graph Using Twitter Data"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in February 2018.

Group Members:

Sukanta Kundu

Anindya Biswas

Qazi Fahim Farhan

Supervisor:

Dr. Muhammad Masroor Ali
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

ACKNOWLEDGEMENT

We are indebted to our supervisor Professor Dr. Muhammad Mashroor Ali got introduing us to the Knowledge Graph. We've learned how to think critically like a researcher, carry on a research, read scientific papers, and how to write , speak and present properly. We show our gratitude for his dedication and patience in correcting our mistakes, reviewing our ideas, methodologies, proofs, etc. We couldn't have done it without him.

Dhaka
February 2018

Sukanta Kundu

Anindya Biswas

Qazi Fahim Farhan

Contents

List of Figures

List of Tables

List of Algorithms

ABSTRACT

In this thesis, we wish to work on creating a Knowledge Graph using Twitter micro-data. A Knowledge Graph is a representation of human knowledge in the form of a graph. Just like any graph, it has nodes (entities), attributes and edges (relations). Since tweets give us limited data (maximum length of 300 characters), it becomes very difficult to extract adequate information to create a knowledge graph.

Chapter 1

Introduction

The present world is a place with immeasurable data or information due to World Wide Web. Where the web is a vast repository of knowledge. Each and every day this amount is increasing .It is getting harder day by day for a person to extract exact information s/he needs to analysis them to have a desired goal.

But automatically extracting that huge knowledge at scale has proven to be a formidable challenge for computer science to organize, store and most importantly efficiently searching. Asearch data structure is anydata structurethat allows the efficient retrieval of specific items from asetof items, such as a specificrecordfrom adatabase.

A single word can represent a person, place, name of organization, restaurant. Again a word slightly different spelling can represent totally different thing. As far as the word efficient searching means searching a particular word with all its variations. It is very hard for a single person or a particular software to extracting the exact semantic data. It may be a better way for us to make machine learn the way of efficient searching.

It may be helpful for us to write grammars for machine to learn than writing all the rules of searching by ourselves. Now for machine to learn to work or search in order to search efficiently we need data or information to be organized in a way that machine can analyze those data. Here knowledge graph comes into play.

Knowledge graphs have become an increasingly crucial component in machine intelligence systems, powering ubiquitous digital assistants and inspiring several large scale academic projects across the globe. Many problems in AI require to deal with both relational structure and uncertainty. As a consequence, there is a growing need for tools that facilitate the development of complex probabilistic models with relational structure.

All these facts are interrelated, and hence, recently this extracted knowledge has been referred to as a knowledge graph . A key challenge in producing the knowledge graph is incorporating noisy information from different sources in a consistent manner. Information extraction systems operate over many source documents, such as web pages, and use a collection of strategies to generate candidate facts from the documents, spanning syntactic, lexical and structural features of text. Ultimately, these extraction systems produce candidate facts that include a set of entities, attributes of these entities, and the relations between these entities which we refer to as the extraction graph.

Recent evaluation efforts have focused on automatic knowledge base population [1,2], and many well-known broad domain and open information extraction systems exist, including the Never-Ending Language Learning (NELL) project [3], OpenIE [4], and efforts at Google [5], which use a variety of techniques to extract new knowledge, in the form of facts, from the web.

With the goal of teaching machines to understand human conversations, one of the most fundamental components of a conversational understanding system is the semantic parser. Conversational semantic parsers map natural language (NL) to a formal representation of meaning, typically defined by the intent of the user and the associated arguments of the intent (slots or concepts) [1].

Considerable advancements in semantic parsing have been made possible by the availability of massive volumes of data from social media. With the recent emergence of very large-scale semantic knowledge graphs (KGs) [7], it is now possible to add structure to the machine learning procedures developed above. Specifically, we have developed methods to enrich KGs with automatically annotated training data through unsupervised data mining methods.

Our approach is large-scale multi-concept (entity, relation, fact) open domain semantic parsing. Our approach is web-scale, learning neural embedding for all the concepts of twitter. Also, while the other approaches rely on supervised training, our approach is unsupervised. We use microblogging and more particularly Twitter for the following reasons: Microblogging platforms are used by different people to express their opinion about different topics, thus it is a valuable source of peoples opinions. Twitter contains an enormous number of text posts and it grows every day. The collected corpus can be arbitrarily large. Twitters audience varies from regular users to celebrities, company representatives, politicians, and even country presidents. Therefore, it is possible to collect text posts of users from different social and interests groups. Twitters audience is represented by users from many countries.

In this thesis paper we analyze the process, algorithm for constructing knowledge graph. We try to extract information from twitter. Moreover we develop the way of accessing DBpedia database in order to enrich entity relationship in knowledge graph. It means knowledge in graph form. Here nodes are entities which are labeled with attributes typed edges between two nodes capture a relationship between entities. KG is vastly used in google, Amazon as amazon product graph, Facebook graph API, IBM Watson, Microsoft satori. Generally knowledge graph come from structured or unstructured texts and Images and videos.

In our KGs entity-relationship edge is assumed as RDF triples like ¡rdf: Subject, rdf: Predicate, rdf: Object ¿ . A starting with GATE Developer 8.4.1 for analyzing live tweets from twitter many processing resources such as ANNIE, Transducer allowed us identifying Token of different kinds. By codding grammar rule using JAPE in GATE noun, pronoun were identified which will be used as Node in KGs.

But for the seek of better semantic analysis and improved relationship between closer entity, development of a process Word Hashing has been done. Which represents a word of a string as vector of a letter n-grams to reduce the dimensionality bag of words-term vectors. Two words are compared based on the angle between two vectors representing those words. The smaller the angle, the closer the relation between the words. So , our goal is to through all this process developing a better knowledge graph based on closer entity-relationship that is efficient to extract information , process and analyze .

Chapter 2

Related Works

Previously there have been many researches conducted on knowledge graphs. They share some common as well as different approaches. Let us discuss them one by one.

2.1 Knowledge Graph Identification

In the paper 'Knowledge Graph Identification' [1], the authors collected two different data sets. In both of them, they extracted uncertain entities and their relations to create an 'extraction graph'. The resulting extraction graph was full of noise, missing information. So they removed noise, added missing informations. After that, they needed to find candidate facts. Reasoning jointly about candidate facts, extracting their confidence, identifying coreferences, imposing ontological constraints, removing duplicate entities, resolving ontological conflicts were some of the other tasks. Let us take an example.

"We can use **Binary Search Tree**, for fast lookup, addition and deletion of an item". Usually, the word **tree** means something related to plants. But in this case, **tree** refers to a data structure because the words **binary**, **search** are imposing ontological constraints. Therefore we can distinguish between the two meanings and disambiguate them. Next, they needed to create edges between the nodes. To relate a concept with similar concept, the researchers have used Probabilistic Soft Logic (PSL) [2] ontological constraints, identify coreferences etc.

2.1.1 Probabilistic Soft Logic(PSL)

We know, that boolean variables have only two values: $True$ and $False$ or 1 and 0. So boolean logics can tell only between 2 choices. In real world problems, this is not always usable. This is where PSL comes innto play. PSL assigns a value in range $[0, 1]$. For example, the word **tree** is not always related to plants. So, we cannot assign 0 or 1 to it. Instead, we can assign **tree** with a value, say 0.8 with the meaning **plan** and 0.2 with the meaning

PSL has some advantages. We can represent models using first order logic syntaxes. We can have continuously valued random variables that is convenient with uncertainty or probability. Using weights, we can control the importance of model rules.

2.2 Link Detection

One common problem in building a Knowledge Graph is to find appropriate links between various concepts. Mihalcea annd Csomai et al [3] have worked on link detection. They used machine learning on a wikipedia dump to link a concept(article) with its corresponding mention. There they have also calculated link probability of various phrases or mentions. Define link probability of a phrase as the number of wikipedia articles that use the phrase as anchor divided by number of articles that mention the phrase at all. Next they disambiguated links by taking help of the surrounding words (the words themselves and their parts of speech). Here a problem occurs on which meaning to select. The best method is to take the most common meaning first. Then calculate the relatedness with the context. Define $relatedness(a, b) = \frac{log(max(|A|,|B|))-log(|AB|)}{log|W|-log(min(|A|,|B|))}$, where a,b are the Article of interest, A , B are sets of all articles linking to a,b respectively and W is the set of all articles.

After that, by comparing the common meaning with the relatedness, the correct link was found. For example, the word 'tree' has a common meaning 'plant' and so 97% of the time, it will link to a page on plants. But if we find 'binary tree' in a document, the relatedness of 'tree' to 'binary' suggests that this phrase is related to data structure. That is how almost everyone disambiguates links. Followed by this, they did Topic indexing which aims to identify the most significant topics, summarize the document and organize it into various categories, and to answer queries fast. In topic indexing, the key problem is to find the correct significant terms and disambiguate them to appropriate topic.

The Next work deals with a semi-supervised graph regularization method on twitter data wikification [4]. There the authores described the problems as unlinkability (absense of a valid concept from knowledge base), ambiguity and prominence. Some more major problem for working with twitter data are informal writing style, shortness and noisiness. To deal with these problems, the authors developed a graph-based semi-supervised learning algorithm for wikification. Their model at the same time, detects mentions and disambiguates them at both local and global levels. They have also developed meta path-based unified framework to detect relevant mentions. Their method works in this way: given a set of tweets $< t_1, t_2, ..., t_n >$, first find candidate concept mentions $< m, c1, c2, ..., ck >$. From there, we find the most probable concept c related to mention m and so we get $< m, c >$. After that , they constructed a relational graph $G =< V, E >$.

Here set of nodes $V =< v_1, ..., v_n >$, and set of edges $E =< e_1, ..., v_e >$. Here $v_i =< m_i, c_i >$.

2.3 A Deep Neural Networking Approach

Finally, we have an important work of Larry Heck and Hongzhao Huang [5]. First they developed a robust way to represent concepts. Many NLP tools use string concatenation to represent ideas. But most of the time it fails to represent appropriate concepts. For example 'Madrid' is a place, 'Real' means something authentic or true, but 'Real Madrid' refers to a football club. This is a counter example where string concatenation fails. A possible alternative is to use *Word Hashing* with n-gram (tri-gram in this case) word representation. This way turns words into concept-vectors. This representation is robust because it can be used even for unseen words. So for example, *dog* is a word. First put # before and after the word, so it becomes *#dog#*. Then we get these trigrams: #do , dog , og#. Thus $\vec{(}m) = \vec{(}\#dog\#) = \widehat{(}\#do) + \widehat{(}dog) + \widehat{(}og\#)$. Since tweets donot have adequate data, one needs external source of knowledge to make sense of the tweets. In this case, they took a wikipedia dump, converted the documents into concept vectors $\vec{(}c)$ and simply took a dot product of the $\vec{(}m)$ and $\vec{(}c)$. If the tweet and the concept are similar, the two vectors will have smaller angle, ie, $\cos(\theta) = \vec{(}m).\vec{(}c)/mc$. This is defined as the semantic relatedness of two concepts, $R(m, c)$. After that they did neural embedding of knowledge graph. To do so, they tracked a concept and its corresponding subgraph, encode the knowledge as featured vector. then they trained Deep Neural Network to get semantic relationships among various concepts. After that, they took tweets and used their deep neural network on those tweets.

Chapter 3

Tentative Methodology

In order to create a graph $G =< V, E >$, where set of vertices is $V =< v_1, v_2, ...v_n >$ and set of edges $E =< e_1, e_2, ..., e_m >$. Here vertex $v_i =< m_i, c_i >$ where m_i is a tweet and c_i is a concept from dbpedia. So each node is a pair of tweeter mention m_i and its corresponding concept c_i.

To create the nodes, we need to collect tweets. But the tweets donot give adequate data. Since a single tweet doesnot give us adequate data, we can use a set of tweets discuss about same topic leading us to some data to disambiguate meaning. In order to work with tweets, we need to perform some Natural Language Processing. For this we will use GATE Developer [6] , a text analysis or language processing toolkit. It helps us to annotate, gazetize, corefer etc in the tweets.

Next we need to collect data from dbpedia because the tweets themselves donot provide adequate data. After collecting the dbpedia data, we will convert them into n-grams. For each acticles in dbpedia, we will create its corresponding concept vector, c. After that, we need to use machine learning to train how to detect a concept and find relations between them. We also need to train on matching twitter mentions m with their corresponding concept vectors c.

Thus we will have nodes $< m_i, c_i >$. After that, we need to connect the nodes with edges. We will use link detection tecniques as discussed in [3] to find possible links and disambiguate links.

Chapter 4

Methodology

4.1 Proposed Methodology

In order to build a knowledge graph from tweets, we can divide our problem into these sub-problems. They are:

- Extracting information / tuples from tweets
- Merging those tuples with External Knowledge graphs
- disambiguating after Merging

These two hypothesis are elaborated below:

Extracting Information or Tuples from tweets We know that tweets have very limited data. for example, *@ImRo45 is back in the Test squad for Indias tour of Australia.* is a tweet. Someone with adequate knowledge in cricket may assume that this is about a game but otherwise, there is very little information in this tweet. Also, words like prepositions (of, in, for) , articles (a, the) have almost no value. On the contrary, 'India', 'tour', 'Australia' are inpoetant words here. So, we have to extract these *important* words into *tuples* of the form $< u, e, v >$.
Define tuple of a graph $< u, e, v >$ where u is starting node, v is ending node and e is an edge between u and v.

this can be done in two ways:

- Machine Learning
- JAPE

In machine learning approach, one has to divide the dataset into ‘training dataset’ and ‘test dataset’ . After that, one has to manually annotate the test dataset by marking possible tuples. After that, some machine learning algorithm can be trained on the test dataset and finally use it on the test dataset.
JAPE stands for ‘Jolly And Pleasant Experience’. It is a special kind of pattern matching language in GATE developer. By writting rules in JAPE , one can fine-tune GATE developer to extract necessary information which is in this case, finding the appropriate tuples.
In our methodology, we took the liberty to use JAPE. The reason is that, in previous works, other researchers have already used machine learning approaches in this stage. Besides, we know that a machine learning algorithm is only as good as its training dataset. Using JAPE, detect tuples better. How we can use JAPE is expained below:

CASE 1:
First we start by writing rules for simple sentences. For example, *CR7 has joined Juventus* . Our target is to convert this sentences into tuple form like $< u, e, v >=< CR7, join, juventus >$. So our initial target is to write JAPE rules for these kinds of sentences or tweets.

CASE 2:
Practically, sentences like case 1 donot provide much information. So now we need to extract information from the surrounding context. For example, *segment tree is a good data structure.* The previous rule may create tuples like ¡tree, is data¿. This is ambiguous as we cannot differentiate But in this case we want ¡segment tree, is, data structure¿. So we have to extract words like ‘segment’ that will allow us to disambiguate between ‘plants’ and ‘data structure trees’.

CASE 3:
Though tweets are very small, we can occationally come across tweets with multiple (two or three) sentences. In that case, we have to Extract informations from multiple sentences.

4.2 Merging Tuples with External Knowledge Graphs

Since we have very limited data in our tweets, we need the help of an external knowledge graph to build up our version of knowledge graph. Suppose, in the previous step, we got this tuple: Alice loves Bob here $< u, e, v >=< Alice, love, Bob >$. Now the problem is that we need to identify who Alice and Bob really are, ie, if Alice is an actor , a fictional character or someone else.

For this can use Freebase, DBpedia etc as a source of external knowledge graph. In our case, we took the liberty to use DBpedia. How to extract data from DBpedia is given below:

4.2.1 SPARQL

One way to get data from DBpedia is to use the SPARQL. We can write queries in this language, such as

In our case , we have nodes u and v, so we can fetch datas that are related to both of them and make an intersection of those. But practically, this intersection returns a huge list of tentative tuples. We can set a limit in our query to restrict our results' size. But still we need to disambiguate the results. For this, the next section comes into play.

4.3 disambiguating after Merging

For each tuple ¡u,e,v¿ , we have a set of tentative candidates $< a_1, b_1 >, < a_2, b_2 >, < a_n, b_n >$ from dbpedia. We have to select only one of them. For this there are several ways:

4.3.1 Calculating Relatedness

We can compute $relatedness$ of two articles , a , b using the formula $relatedness(a, b) = \frac{\log(max(|A|,|B|))-\log(A \cap B)}{\log(|W|)-\log(max(|A|,|B|))}$ where a , b are the articles of our interest, A = set of all articles linking to a, B = set of all articles linking to b, and W = set of all links in DBpedia. The formula generates a score for each candidate tuples and we select te one with the highest score. This is the easiest way to perform disambiguation.

4.3.2 Vector Analysis

In this way, we can we take a candidate $< a_i, b_i >$, then we read the articles by converting them into $n-grams$, then we calculate the angle between them. The closer the articles are, the better result we can get.

4.3.3 Deep Neural Network

One can also use a dump from DBpedia, and follow the detailed instructions provided here [5] to create a deep neural network that can be used then to link tweet tuples with a corresponding article.

References

[1] L. G. W. C. Jay Pujara, Hui Miao, “Knowledge graph identification,” *International Semantic Web*, pp. 0–1, 2013.

[2] M. B. B. H. L. G. Angelika Kimming, Stephen H. Batch, “A short introduction to probabilistic soft logic,” pp. 0–1, 2012.

[3] I. H. W. David Milne, “Learning to link with wikipedia,” *CIKM*, pp. 0–1, 2014.

[4] X. H. H. J. H. Huang, Y. Cao and C. Lin, “Collective tweet wiki-fication based on semi-supervised graph regularization,” 2014.

[5] H. H. Larry Heck, “Deep learning of knowledge graph embeddings for semantic parsing of twitter dialogs,” *Signal and Information Processing*, pp. 0–1, 2014.

[6] “Gate developer.”

Generated using Undegraduate Thesis LaTeX Template, Version 1.4. Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

This thesis was generated on Wednesday 17th April, 2019 at 11:10am.