Enhance Knowledge Graphs / Knowledge bases like Wikidata

Team No - 9

Team Members: Mentor: Tushar Abhishek

- Pushpa Yadhav (2019900034)
- Vijaya Lakhsmi (2018900071)
- Karthikeyan Arumugam(2018900074)

Demo Video Link : **https://youtu.be/ybX_TofQQ24**
Git Hub Link: **https://github.com/Team9-CS4-501-S21CourseProject/EnhanceWikiData/**
Google Drive Link : **https://drive.google.com/drive/folders/1nz_Mz0dYWBb2j1Y21MIoZm-6XAk8vIfH?usp=sharing**

Project Abstract

Human knowledge provides a formal understanding of the world. Knowledge graphs are excellent tool to represent world knowledge and it effective way to represent semantic information and relations to machines. Knowledge graphs are usually network of interconnected entities such as event, persons, objects etc. where edges connecting such entities are the actual relationship between those entities. Knowledge graph that represent structural relations between entities have become an increasingly popular research direction towards cognition and human-level intelligence. Aids downstream NLP task such as Question and Answering systems.

Some of widely used Knowledge graphs in NLP related research are Freebase, YAGO and WikiData. In this project we would be exploring ways to make knowledge graphs (such as wikidata) denser. Currently the amount of facts present in Knowledge graphs (KG) may not capture most of the world knowledge and publicly available Knowledge graphs can be enriched with more information. Pretrained Language models such as BERT are usually trained over large corpus and it inherently holds vast world knowledge. In this Project we are aim it to enrich the existing knowledge graphs using Pre trained knowledge graphs and wikipage corpus

Problem Statement

The world knowledge and all facts exist in Knowledge graphs (KG) does not capture all of the world knowledge. The Quality of the Knowledge graph can be illustrated by density of edges (relations) and volume of information captured in it. There are various possibilities to enrich and improve the quality of the KG s.

Some of possible challenges at present are

- Knowledge graphs contains entities but not all relations are captured
- Not all real-world entities are captured in Knowledge graphs hence information pertaining to those entities are potentially missing
- Augmenting different KGs/Sources to create enrich KG is tedious task.

Some important question in front of us is

• How to capture more information and enrich existing KGs?

• How to create more edges connecting entities which are already captured in Wikidata ?

• how to Ingrain more world knowledge into existing KGs Effectively ?

• is it possible to Augment different KGs and scattered information to enrich KGs such as wiki data. ?

Scope of this project goals

1) We will be Using WikiData as our Base Knowledge Graph
2) We will be Focusing on specific domains for this project. Particularly domains which has relatively less volume of data in wikidata as of 2021.
3) We will be using Wikipedia pages as source to enrich domain we choose

Assumptions

1. Pretrained Language models such as BERT will contain vast amount of world knowledge that could help us in extracting entities and relationships from our source documents.
2. By using principles of transfer learning, we will be reusing existing pretrained models and resources to reduce complexity and cost of computation.
3. Content generated from this project will Aid data addition to wikidata submission process and it is not replacement to existing mechanisms.
4. Our models will be semi/un-supervised in nature to reduce human intervention required to enrich existing knowledge graphs.

Literature Survey

A Survey on Knowledge Graphs: Representation, Acquisition and Applications , Ji et al.,2021

This research conducted a comprehensive survey on 4 scopes

a. knowledge graph embedding

b. knowledge acquisition of entity discovery

c. temporal knowledge graph representation learning

d. real-world knowledge-aware applications

This paper discussed deeper into theory and methods involves in various aspects of Knowledge graphs , they covered technical aspects such as knowledge representation, techniques to acquisition of new knowledge , various models and their comparison in detail. K-BERT,LSTM-CNN,RNN, BiLSTM , reinforcement learnings ,and various other methods were discussed in detail and how it can be used in various stages of Knowledge graph lifecycle. This gave us lot of insight in state-of-art research effort going in Knowledge graph related area.

Accurate Text-Enhanced Knowledge Graph Representation Learn-ing, Bo An et al,. 2018

This paper proposes representation framework called "an accurate text enhanced knowledge graph representation framework", which enhance the knowledge representations of a triple, and effectively handle the ambiguity of relations and entities through a mutual attention model between relation mentions and entity descriptions. Their experiment results show that their method can achieve the state-of-the-art performance, and significantly outperforms previous text-enhanced knowledge representation models

Learning to Update Knowledge Graphs from Reading News,(Tang et al., 2019)

This research work proposes a novel graph based neural network method called GUpdater. GUpdater is built upon graph neural network (GNN) with a text-based attention model. This model was able to effectively perform link-adding or link-deleting operations to ensure the KG up to date according to news snippets. Experiments demonstrated that this model could handle explicit and implicit information found in news sources.

Collective Multi-type Entity Alignment Between Knowledge Graphs,Qi Zhu1 et al.,2020
This research paper presents a new method "Collective Graph neural net-work for multi-tyoe entity alignment" (CG-MuAlign). This method jointly aligns multiple types of entities, collectively leverages neighborhood infor-mation and generalizes to unlabled entite types. This experiment propose 2novel collective aggregation function tailored for reliving the incomplete-ness of the knowledge graphs via both cross-graph and self attentions,it also scales up effectively with mini-batch training paradigm and effective neighborhood sampling strategy.

Their experiments with real world knowledge graphs with millions of enitites and they observed superior performance beyond exisiting methods. Running time of this method is much less that current state-of-the-art deep learning methods. Experiments demostrated that this proposed method can handle multiple knoeledge graphs alignment simultaneously.

Language Models are Open Knowledge Graphs, Chenguang Wang et al.,2020

This paper proposes an unsupervised method to cast the knowledge contained within language models into KG s. Specifically it shows how to construct knowledge graphs (KGs) from pre-trained language models (e.g., BERT, GPT-2/3), without human supervision. This paper introduces a two-stage unsupervised approach called ***MaMa*** (Match and Mapping),which can successfully recover the factual knowledge stored in language models to build KGs from scratch. This ***MaMa*** Constructs a KG with a single forward pass of pretrained language models over a textual corpus. Further experiments with this model demonstrated that open knowledge graph features new facts when compared to WikiData and TAC KBP. This Model establishes a bridge between the deep learning and knowledge graphs, its results suggests that larger language models store richer knowledge than existing knowledge graphs

Implementation Strategy

After considering all literatures and recent advancements in the field of generating Knowledge graphs , we finalized that approaches mentioned in “language models are open knowledge graphs” will suit our goal of enriching Knowledge graphs.

The method proposed in that paper provides us better method to avoid duplication of knowledge in existing KG and help us adding new entities and relations to it. In a true sense it fits into the goal of Enrichment of Knowledge graphs.

Methodology

On a high level Model which we considered , proposes to construct Knowledge Graphs which is a structured object that's usually built by human experts. It proposes to construct Knowledge Graphs automatically by simply using a pre-trained language model together with a corpus to extract the knowledge graph without human supervision.

In Language Models are Open Knowledge Graphs paper, the authors design an unsupervised approach called ***MAMA*** that successfully recovers the factual knowledge stored in Language Models to build Knowledge Graphs from scratch. ***MAMA*** constructs a KG with a single forward pass of a pre-trained LM (without fine-tuning) over a textual corpus. Interesting thing about this model is that transferring existing capabilities from pretrained systems instead of training models from scratch.

MAMA- Corpus to Knowledge Graph

We have a corpus, a corpus is simply a bunch of text pieces, and we want to extract a Knowledge Graph out of it. The Knowledge Graph consists of 2 distinct things, it has entities(you can think of it as nouns) and relations(occupation, signed, award received, etc.) here the relations connect to the entities to form meanings. Now, let’s discuss the two stages required to build the Knowledge Graph.

Match stage

Match stage Generates a set of candidate facts by matching the facts in the textual corpus with the knowledge in the pre-trained Language Model. Candidate facts are a set of extracted string triplets(namely; head, relation, and tail respectively). The longer the sentence the harder it is to extract the triplet strings.

The Map stage

The candidate facts are mapped to a schema and these schemas are usually defined by humans. So here we’re still going to rely on humans to define the schema. This stage produces an open KG via mapping the matched candidate facts from the Match stage to both fixed KG schema and open schema. If the schema of candidate facts exists in the KG schemas (Wikidata and TAC KBP) annotated by humans, we map the candidate facts directly to the fixed KG schema. Otherwise, we reserve the unmapped candidate facts in the open schema. This results in a new type of Knowledge Graph (KG), known as open KG, with a mixture of mapped facts in fixed KG schema and unmapped facts in the open schema.

Candidate Fact-Finding Algorithm

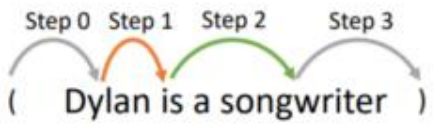
How do you come up with a Knowledge Graph from a corpus? The authors used three major steps in the map stage to build a KG from a corpus.

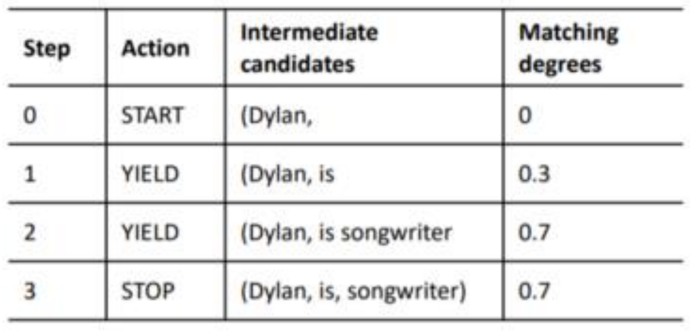
Step 1:

Run spaCy (spaCyis a free open-source library for Natural Language Processing in Python. It features NER, POS tagging, dependency parsing, word vectors, and more)to extract noun phrases/chunks. Running the spaCy library through the corpus will find the head and tail of the extracted candidate facts. But the downfall to that spaCy is probably going to miss a lot of words that are not recognized as noun phrases (in huge sentences) and the authors also say that spaCy annotations are sometimes error-prone.

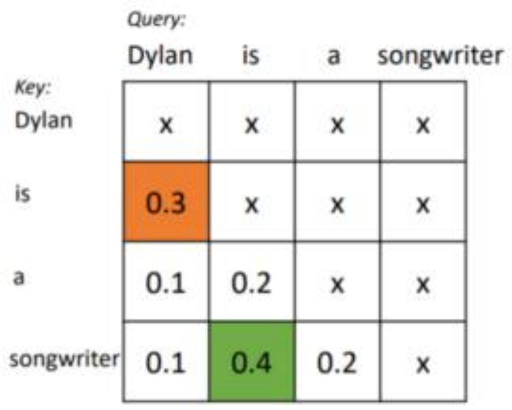
Step 2:

This is where their system/method comes in. Now you have found the head and tail in the text, somewhere in between there might be a relation and the system needs to figure out where that is. So how does this method figure it out? The algorithm is going to do something which is similar to dynamic programming and search algorithms (Beam search). The algorithm starts from the head of the string (there could be text before it) we’re simply gonna locate the head and then we’re gonna look at its attention matrix. Now the attention matrix is basically how much each token attends to each other token, i.e, how much information is sent from each other token to a particular token. Since we are locating a relation between the head and the tail, we’re gonna disregard all the token behind the query and only look ahead in the sentence. So that’s why the upper half of the attention matrix is crossed out.





Step	Action	Intermediate candidates	Matching degrees
0	START	(Dylan,	0
1	YIELD	(Dylan, is	0.3
2	YIELD	(Dylan, is songwriter	0.7
3	STOP	(Dylan, is, songwriter)	0.7



Key: \ Query:	Dylan	is	a	songwriter
Dylan	x	x	x	x
is	0.3	x	x	x
a	0.1	0.2	x	x
songwriter	0.1	0.4	0.2	x

From the token ‘Dylan’ we can only look at the tokens ahead of it (is, a, and songwriter). Where do we go next?

We’re gonna take the highest scoring value in that column of the attention matrix. Look at the attention column for the token ‘Dylan’, take the highest scoring value (0.3), then we go to the token associated with the highest scoring (‘is’ in this case) and append that token (‘is’) in the candidate fact. Once the highest-scoring token is put in the candidate fact we go to the next token and again look in the corresponding attention column, take the highest scoring value (0.4), then we go to the token associated with the highest scoring (‘songwriter’ in this case) and append that token (‘songwriter’) in the candidate fact. The algorithm discovers that the token ‘songwriter’ is actually our tail and tail is the last word, hence, the algorithm stops. We always go forward with the biggest entry in the attention matrix until we reach the tail and that’s the algorithm!!

The assumption that the relation must be in between the head and tail textually and the relation must be encoded as a semi-accurate string is super constrained. The biggest flaw with this constrained problem is that the algorithm has a very very low recall (so do all the systems for this task, apparently).

Step 3:

Use the entity linking and relation linking system to construct the knowledge graph.

Constraints

Constraint# 1

The matching degree (the sum of all these attention matrix entries that were encountered during the search, all the ones we didn't skip) must be above some threshold.

Constraint# 2

The frequency of r is above a threshold. The relation itself shouldn't be too specific it should appear a bunch of times in the corpus. You go through the corpus once, extract all the candidate facts, and then you count them and go through the candidate facts again and delete all the ones which are below a certain threshold. Similar to the removal of stop words and rare words.

Constraint# 3

Relation r is a contiguous sequence in the sentence.

Mapping Mapped Facts To KG Schema

The goal is to map a candidate fact (h, r, t) to a fact (hk, rk, tk) in the KG schema The reason for mapping to an existing KG schema is to make use of the high-quality schema designed by experts (to avoid duplicated efforts of building from scratch) and enable evaluating the candidate facts with oracle KG facts contributed by human volunteers. There are two types of mapped candidate facts:

Entity linking to KG schema

The authors leveraged an unsupervised entity linker based on a mention to-entity dictionary to link the entities for scalability consideration. Besides, contextual information is crucial to link the entities correctly, the authors use the word embedding of the context to disambiguate the entities.

Relation mapping with KG schema

The authors largely follow the relation mapping method to construct an offline relation map between KG relation and relation phrases of the candidate facts. The basic idea is that the more often linked head-tail pairs (i.e., entities are with type information from the entity linking step) co-occur between the candidate facts and KG facts, the more likely the corresponding relations are mapped to each other.

Mapping Unmapped Facts to Open Schema

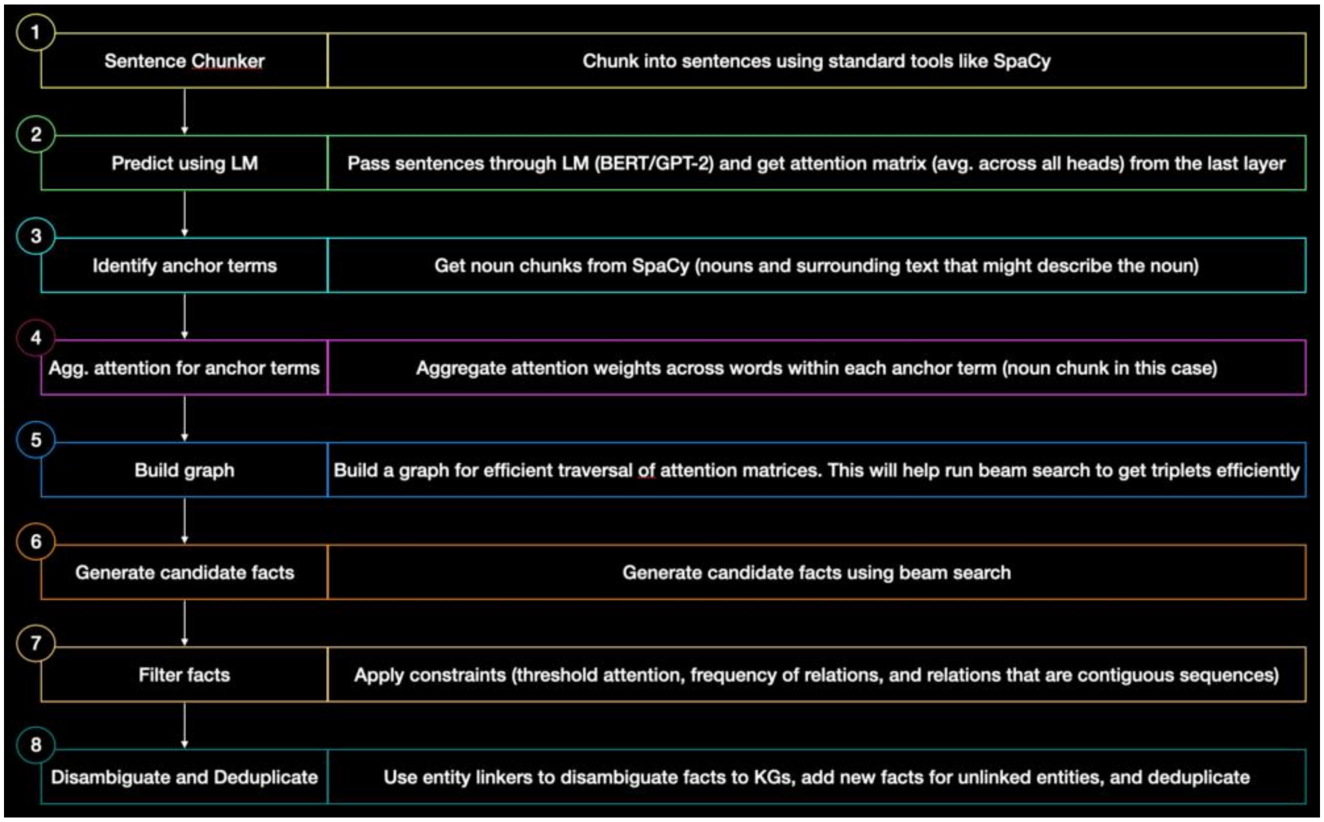
An unmapped candidate fact (h, r, t) means at least one of h, r, and t is not mapped to the KG schema. There are two types of unmapped candidate facts:

Partially unmapped facts

The authors represent at least one of h, r, and t are mapped to the KG schema. It can be h or t mapped to hk or tk based on the entity linker. It can also be r that mapped to rk using the relation mapping. This actually results in unmapped facts that are in a mixture of the KG schema and the open schema.

Completely unmapped facts

The authors indicate all h, r, and t are not mapped to the KG schema. This means neither the entity linker nor the relation mapping is able to map h, r, and t to hk, rk, tk respectively. The resulting unmapped candidate facts stay in the open schema.



1	Sentence Chunker	Chunk into sentences using standard tools like SpaCy
2	Predict using LM	Pass sentences through LM (BERT/GPT-2) and get attention matrix (avg. across all heads) from the last layer
3	Identify anchor terms	Get noun chunks from SpaCy (nouns and surrounding text that might describe the noun)
4	Agg. attention for anchor terms	Aggregate attention weights across words within each anchor term (noun chunk in this case)
5	Build graph	Build a graph for efficient traversal of attention matrices. This will help run beam search to get triplets efficiently
6	Generate candidate facts	Generate candidate facts using beam search
7	Filter facts	Apply constraints (threshold attention, frequency of relations, and relations that are contiguous sequences)
8	Disambiguate and Deduplicate	Use entity linkers to disambiguate facts to KGs, add new facts for unlinked entities, and deduplicate

As we see from the flow above, the pipeline involves a lot more than just running the sentences through an LM. While an LM may help us generate candidate facts, it is equally important to tune the other parts of the pipeline. All the other pieces of the pipeline try to leverage pre-trained models from libraries like spaCy, REL and Flair.

The Experiments

Domains

After Considering wikidata stats page (https://www.wikidata.org/wiki/Wikidata:Statistics),

we have short listed following domains for this project.

- Film (Current Size - 294,370 or 0.4% of Wiki Data)
- Chemical compound (Current Size - 1,188,724 or 1.7% of Wiki Data)
- Through fares (Current Size - 630,794 or 0.9% of Wiki Data)
- Wiki News Articles (Current Size - 195,900 or 0.3% of Wiki Data)

Our Focus in this experiment was to enrich domains which has less volume of data when compared to mainstream such as Persons Domain.

Value Add to Language Models are Open Knowledge Graphs

- We add Pair wise confidence score
  - This Score can enable us to filter out junk content generated from source pages.
  - Score can be used as Scale to measure degree of quality of the extracted knowledge graph entities and their relations.

Methods and Tools

Development Tools/Environments: Colab, Jupiter, spyder

Project Implementation Methodology: Scaled Agile

Libraries:

- Allennlp
- wikidata
- Spacy
- Nltk
- numpy

- Pandas
- Regex
- Scipy
- Spacy entity linker
- tokenizers
- torch
- Transformers

Results

We ran our experiment with following number of wiki pages.

1) Throughfares : ~230 Pages
2) News Article : ~ 270 Pages
3) Films ; ~ 750 Pages
4) Chemical Names : ~750 pages

After running few combinations of our configurations , we generated following amount of knowledge graphs entries from all these files.

1) For Chemical compounds – ~23000 New Entities in the newly generated KG
2) For news : ~10000 New Entities in the newly generated KG
3) For Films : ~16000 New Entities in the newly generated KG
4) For Through fares : ~7000 New Entities in the newly generated KG

Even after adjusting various threshold combination , we will still get many noisy Entity names

Detailed list of generated KG is available as csv file in our Github repository under "Final Generated KG files"

Conclusion and Next steps

After running several experiments , we established that Using MAMA algorithm we can mine knowledge graph from any source file and map it against existing knowledge graphs such as wikidata. Though result is driven by Pre trained language models, Coreference resolution tools, Entity linkers etc, we have greater control over how to reuse existing pretrained machine learning models and assets to fine tune our system. By employing domain specific entity linker and coreference models , we can significantly improve the way we mine the knowledge graphs from text corpuses.

In future we would like to continue following improvements in this system

- Compare generated knowledge by running same experiment with different pretrained language models
- For domain specific corpus – use assets built specifically for the domain and compare results between domain specific entity linker, Coref Resolver vs generic assets.
- By Increasing threshold value we would like to better quality results with higher confidence score.
- Validate this proposed model in Non english/Indian language setup
- Create a better relationship linker and semantic level resolution for edges in the knowledge graphs.

References

”A Survey on Knowledge Graphs: Representation, Acquisition and Applications” , Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, Philip S. Yu, 2020 , arXiv:2002.00388 [cs.CL]

”Accurate Text-Enhanced Knowledge Graph Representation Learning”,Bo An, Bo Chen, Xianpei Han and Le Sun,2018, ”Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 , https://www.aclweb.org/anthology/N18-1068,10.18653/v1/N18-1068

“Learning to Update Knowledge Graphs from Reading News” , Jizhi Tang,Yansong Feng, Dongyan Zhao , Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International. Joint Conference on Natural Language Processing (EMNLP-IJCNLP) 10.18653/v1/D19-1265

https://www.aclweb.org/anthology/D19-1265

“Collective Multi-type Entity Alignment Between Knowledge Graphs” , Qi Zhu , Hao Wei, Bunyamin Sisman, Da Zheng , Christos Faloutsos, Xin Luna Dong , Jiawei Han,2020, Proceedings of The Web Conference 2020, https://doi.org/10.1145/3366423.3380289

https://assets.amazon.science/ff/7a/b96282984a0fbe5e31a8fcf68d17/scipub-1202.pdf

“Language Models are Open Knowledge Graphs” , Chenguang Wang, Xiao Liu, Dawn Song , 2020. arXiv:2010.11967