ECE 5831: Pattern Recognition and Neural Networks

Context Derivation for Knowledge Graph Expansion Using Commonsense Transformers

Authors: Haard Rao, University of Michigan – Dearborn; Rohit Sanjay, University of Michigan – Dearborn; Neel Khakhar, University of Michigan - Dearborn

Abstract-- Humans can speak, read, and write with the aid of language, which is a form of communication. For instance, we use natural language—more specifically, words—to think, decide, plan, and do other things. The key question, though, is whether we can converse similarly with machines in the age of AI. In other words, is it possible for people to speak naturally to computers? Because computers require organized data but human speech is unstructured and frequently confusing in nature, it is difficult for us to create NLP applications.
This makes it possible to define Natural Language Processing (NLP) as the area of computer science, particularly Artificial Intelligence (AI), that deals with teaching computers how to comprehend and use human language. Technically speaking, the main function of NLP would be to program computers to process and analyze vast amounts of natural language data.

***Index Terms*—Natural Language Processing (NLP), Artificial Intelligence (AI), process, analyze, computer science.**

I. Introduction

Data from an information extraction task can be stored in a knowledge graph. A concept known as a "triple"—a group of three items—a subject, a predicate, and an object—that we might use to hold information about something—is used in many fundamental knowledge graph implementations.

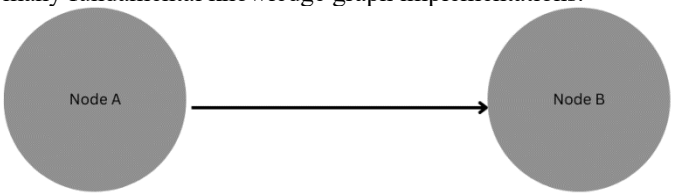


Fig1: Relationship of Node A and Node B

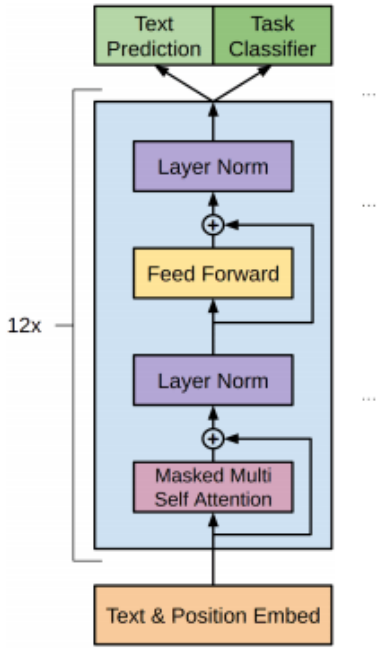
Here, Node A and Node B are two unique entities. These nodes are connected by an edge that represents the relationship between the two nodes. This is the smallest knowledge graph that we may create; it is often referred to as a triple. Knowledge Graphs can be any size or shape.

II. Project Components

A. GPT-2 based approach

GPT-2 has a wide range of capabilities, including the ability to generate conditional synthetic text samples of unprecedented quality by priming the model with an input and having it generate a lengthy continuation. Furthermore, without the use of domain-specific training datasets, GPT-2 outperforms other language models trained on specific domains (such as Wikipedia, news, or books). GPT-2 begins learning language tasks like question answering, reading comprehension, summarization, and translation from raw text, with no task-specific training data. While the scores on these downstream tasks are far from optimal, they do suggest that the tasks can benefit from unsupervised techniques if sufficient (unlabeled) data and compute are available.

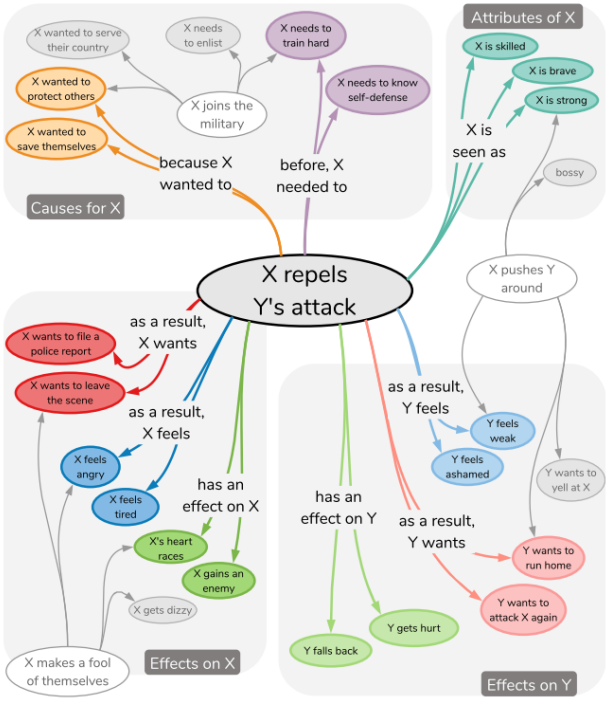
On its own, GPT-2's performance is still significantly lower than the 30 to 50% range of open domain question answering systems that combine information retrieval with extractive document question answering.



Shortcomings of using pre-trained LMs as knowledge providers: (i) insufficient coverage, (ii) insufficient precision, and (iii) limited reasoning capabilities.

Atomic KG commonsense knowledge graph with 1.33M everyday inferential knowledge tuples about entities and events.

It represents a large-scale common sense repository of textual descriptions that encode both the social and the physical aspects of common human everyday experiences.



Relations in ATOMIC2020 along with illustrative examples and their respective size.

Head	Relation	Tail	Size
	ObjectUse	make french toast	165,590
bread	AtLocation	basket; pantry	20,221
	MadeUpOf	dough; wheat	3,345
	HasProperty*	cooked; nice to eat	5,617
	CapableOf*	coat cake with icing	7,968
baker	Desires*	quality ingredients	2,737
	Not Desires	bad yeast	2,838
	IsAfter	X exercises in the gym	22,453
	HasSubEvent	become tired	12,845
X runs out	IsBefore	X hits the showers	23,208
of steam	HinderedBy	drinks too much coffee	106,658
	Causes	takes a break	376
	xReason	did not eat breakfast	334
X watches --- anyway	isFilledBy	the game; the TV	33,266
	xNeed	do something tiring	128,955
	xAttr	old; lazy; lethargic	148,194
X runs out of steam	xEffect	drinks some water	115,124
	xReact	tired	81,397
	xWant	to get some energy	135,360
	xIntent	to give support	72,677
X votes	oEffect	receives praise	80,166
for Y	oReact	grateful; confident	67,236
	oWant	thank X: celebrate	94,548

Combination of GPT2 and ATOMIC Knowledge graph gives us a COMET (Commonsense Transformers for Automatic Knowledge Graph Construction) Knowledge graph. We evaluate this model with SocialIQA dataset. Finally, through human evaluation, we show that the few-shot performance of GPT-3 (175B parameters), while impressive, remains ~12 absolute points lower than a BART-based knowledge model trained on ATOMIC20 20 despite using over 430x fewer parameters. [paper]

B. BERT based approach

BERT is the abbreviation of **B**idirectional **E**ncoded **R**epresentations from **T**ransformers.

- **Bidirectional:** You must read both the text you are looking at and the previous words in order to understand it (at the next words)
- **Transformer:** Token sequences are read in their whole by the Transformer at once. The model is non-

directional in a sense, whereas LSTMs read sequentially (left-to-right or right-to-left). Learning how words relate to one another in context is made possible by the attention process.

15% of the tokens were hidden throughout BERT's training in order to train it to estimate them. Predicting the next sentence was another goal.
Google used TensorFlow to create and train the original BERT model. The two sizes of BERT are BERTBASE and BERTLARGE.

The LARGE model generates state-of-the-art results that were presented in the research paper, whereas the BASE model is utilized to compare the performance of one architecture to another.

Semi-Supervised Learning was one of the primary factors in BERT's successful completion of several NLP tasks. This indicates that the model has been trained for a particular task that enables it to comprehend the linguistic patterns. Once trained, the model (BERT) has the ability to process language, which may be utilized to strengthen other models that we create and train using supervised learning.
BERT is just a transformer architecture with an encoder stack. An encoder-decoder network using self-attention on the encoder side and attention on the decoder side is known as a transformer architecture.
The Encoder stack in BERTLARGE contains 24 layers compared to BERTBASE's 12 layers. These go beyond the Transformer design as it was originally defined in the paper (6 encoder layers).
In addition, LARGE and BASE BERT architectures have more attention heads (12 and 16 respectively) and larger feedforward networks (768 and 1024 hidden units) than the Transformer architecture proposed in the original paper. It has 8 attention heads and 512 hidden units. While BERTLARGE has 340M parameters, BERTBASE only has 110M.

Why do we need BERT?

The absence of sufficient training data is one of the main problems facing NLP. Although there is a vast quantity of text data available overall, we must divide it up into the many different fields in order to build task-specific datasets. We only have a few thousand to a few hundred thousand human-labeled training examples after doing this. Unfortunately, deep learning-based NLP models need a lot more data to function well; they significantly improve when trained on millions or billions of annotated training instances.
Researchers have created a variety of methods for training general purpose language representation models using the massive amounts of unannotated content on the web in order to fill up this data gap (this is known as pre-training). When working with issues like question answering and sentiment analysis, for example, these general purpose pre-trained models can subsequently be fine-tuned on smaller task-specific datasets. When compared to starting from zero and training on the smaller task-specific datasets, this method significantly increases accuracy. In the deep learning field, BERT, a relatively new addition to these techniques for NLP pre-training, generated a stir because it demonstrated cutting-edge outcomes in a range of NLP tasks, including question answering.

Idea behind BERT

What exactly does language modeling entail? What issue are language models attempting to address? In essence, their job is to use context to "fill in the blank." For instance, given: “The woman went to the store and bought a _ of shoes.”
The word "cart" would be used to replace the void in this sentence 20% of the time, while the word "pair" would be used 80% of the time, according to a language model.
A language model would have studied this text sequence during training from either left to right or from a combination of left to right and right to left in the pre-BERT era. For creating sentences, this one-directional method works well. We may anticipate the subsequent word, add it to the sequence, and then predict the subsequent word until we have a complete sentence. BERT, a bilingually trained language model, now joins the conversation. As opposed to the single-direction language models, we can now perceive the context and flow of language more deeply.
BERT employs a cutting-edge method called Masked LM (MLM), which randomly masks words in the phrase before attempting to predict them. This method is used in place of traditional word prediction. When a word is "masked," the model uses both left and right surrounds, as well as both directions of the sentence, to anticipate the hidden word. It considers the previous and following tokens simultaneously, in contrast to earlier language models. This "same-time portion" was absent from the existing combined left-to-right and right-to-left LSTM based models. (BERT may be more accurately described as non-directional.)
Language representations that have already been trained can either be context-free or context-based. Consequently, context-based representations may be unidirectional or bidirectional. For each word in the lexicon, context-free methods like word2vec provide a single word embedding representation (a vector of numbers).
The term "bank," for instance, might be represented as both "bank account" and "bank of the river" without regard to context. The representation of each word in context-based models, on the other hand, is based on the meanings of the other words in the phrase. A unidirectional contextual model, for instance, would represent "bank" based on "I accessed the" but not "account" in the sentence "I accessed the bank account." BERT, on the other hand, deeply bidirectionally depicts "bank" utilizing both its previous and next context — "I accessed the... account" — starting from the very bottom of a deep neural network.
The information goes from left to right solely in the unidirectional GPT, while BERT is bidirectional and ELMO is shallowly bidirectional that can be understood from the above example.

.

C. spaCy

SpaCy is a Python package for high-level Natural Language Processing (NLP), which is open-source and free. It is created

to produce information extraction or natural language processing systems and is built in Cython. It offers a clear and approachable API and is designed for use in production.
For instance, what's the topic? What do the words in the sentence mean? Who is treating whom with what? Which businesses and goods are mentioned? What books share similarities with one another?
SpaCy enables one to create applications that handle and "understand" massive amounts of text because it is made primarily for usage in production environments. It can be used to create systems for information extraction, NLU, or text pre-processing before deep learning.
SpaCy is not "an API" or a platform. SpaCy does not offer software as a service or a web application, in contrast to a platform. It's an open-source library, not a product, made to assist you in creating NLP applications.
SpaCy isn't a standard chatbot engine. While spaCy can be used to power conversational apps, it is not intended to be used to power chat bots explicitly; rather, it just offers the text processing basics.
The research software spaCy is not. Although it is based on the most recent research, it is made with efficiency in mind. As opposed to NLTK or CoreNLP, which were developed as platforms for teaching and research, this necessitates somewhat different architectural choices. The primary distinction is that spaCy is opinionated and integrated. SpaCy makes an effort to avoid presenting the user with a choice between several algorithms that offer the same functionality. SpaCy can provide usually better performance and developer experience by keeping the menu short.
Some of spaCy's properties allow it to anticipate linguistic annotations, such as whether a word is a verb or a noun, whereas others need statistical models to be loaded. Currently, spaCy provides statistical models that may be installed as separate Python modules for many different languages. Models might vary in terms of their accuracy, size, speed, memory requirements, and data content. Always consider your use case and the texts you'll be working with while selecting a model. The simple, default models are always an excellent place to start for a general-purpose use case. Usually, they consist of the following elements:

- Binary weights to forecast such annotations in context for the named entity recognizer, dependency parser, and part-of-speech tagger.
- The vocabulary's lexical items are words and their contextually irrelevant characteristics, like their shape or spelling.
- Lemmatization rules and lookup tables are examples of data files.
- Word vectors, or multidimensional representations of a word's meaning, allow you to assess how similar two words are to one another.
- Options for configuring spaCy so that it is in the right state when the model is loaded, such as language and processing pipeline parameters.

D. Natural Language Tool Kit (NLTK)

The automatic software manipulation of natural language, such as speech and text, is known as natural language processing, or NLP for short.
With the advent of computers, the study of natural language processing, which has been around for more than 50 years, evolved from the study of linguistics.
Let's first attempt to respond to the following query before we delve further into NLP.
What is natural language, and how does it vary from other kinds of information?
Natural language refers to the spoken and written forms of communication that humans use to interact with one another.
All around us, there is text. Think about how much text you see each day:

- Signs
- Menus
- Email
- SMS
- Web Pages

And so much more. The list is endless.
Now consider speech. As a species, we might speak to one another more frequently than write. Speaking may possibly be simpler to learn than writing. Our primary forms of communication are voice and text. We must have tools to comprehend and analyze natural language, just as we do for other sorts of data, given the significance of this kind of data.
The NLTK module is a sizable toolkit created to aid in the Natural Language Processing (NLP) process as a whole. NLTK will help with everything from separating sentences from paragraphs to breaking words into their component parts, emphasizing the major ideas, and even assisting machines in understanding the content of the text. We'll discuss sentiment analysis, often known as opinion mining, in this tutorial.
In order to do sentiment analysis, NLTK uses:

- Tokenizing - Splitting sentences and words from the body of text.
- Part of Speech tagging
- Machine Learning with the Naive Bayes classifier
- Scikit-learn (sklearn) with NLTK
- Training classifiers with datasets

III. OUR GOAL

We anticipate that our system will use the provided dataset as its input and produce a knowledge graph as its output. All of the nodes' (entities) connections with each other will be shown in the knowledge graph.

I. Data representation in knowledge graph

For example, consider this line: ***London is the capital of England. Westminster is located in London.***

After basic processing, we would get 2 triples: *(London, be capital, England), (Westminster, locate, London)*

As a result, there are three distinct entities (London, England, and Westminster) and two relations in this

example (be capital, locate). We simply need two associated nodes in the graph with the entities and vertices with the relations to construct a knowledge graph, and we will obtain something similar to this:
It is not scalable to create a knowledge graph manually. Nobody is going to sift through hundreds of pages to extract every entity and their relationships!
Machines are therefore better suited to handle this activity because they can easily sort through hundreds or even thousands of pages. The fact that machines cannot grasp natural language presents still another difficulty. Natural language processing (NLP) enters the scene in this situation.

Making our machine comprehend natural language is crucial if we want to create a knowledge graph from the text. NLP methods including sentence segmentation, dependency parsing, parts of speech tagging, and entity recognition can be used to do this.

II. Sentence segmentation

The text document or article must first be divided into sentences before a knowledge graph can be constructed. Then, we will only shortlist the phrases that have exactly one subject and one object.

III. Entities Extraction

It is not difficult to remove a single word component from a sentence. With the use of parts of speech (POS) tags, we can accomplish this quickly. Our entities would be nouns and proper nouns.
However, POS tags alone are insufficient when an entity spans several words. The sentence's dependency tree needs to be parsed.
The nodes and the edges connecting them are crucial components in the construction of a knowledge graph. The entities that appear in Wikipedia sentences will make up these nodes. These entities' connections to one another are represented by edges. Using the sentence structure, we will utilize an unsupervised method to extract these parts.
The fundamental concept is to go through a sentence and identify the subject and object as you come across them. One problem is that an entity can span numerous words, as in the case of "red wine," yet dependency parsers only classify the individual words as subjects or objects.

IV. Relation Extraction

The first portion of the task is entity extraction. We need edges to link the nodes (entities) together in a knowledge graph. These edges represent the connections between two nodes.
Our theory is that a sentence's primary verb is actually the predicate.

V. Building knowledge graph

The knowledge graph is created from extracted entities and the predicates; i.e., subject-object pair and relation entities.
Netwokx library is used to create a network from this frame. Nodes represent the entities and connections or particularly called edges represent relation between different nodes(entities).

IV. CODE IMPLEMENTATION

For the beginning, we downloaded all the required external libraries. Significant potential is unlocked by external libraries in data science. Numerous data science libraries provide considerable advantages for machine learning, data analysis, and visualization. We use these libraries rather than developing each function from scratch. These libraries are spaCy, pandas, bs4, requests, networkx, matplotlib, etc.

```
[ ] import re
    import pandas as pd
    import bs4
    import requests
    import spacy

    /usr/local/lib/python3.8/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't initialize NVML
      warnings.warn("Can't initialize NVML")

[ ] from spacy import displacy
    nlp = spacy.load('en_core_web_sm')

[ ] from spacy.matcher import Matcher
    from spacy.tokens import Span

    import networkx as nx

    import matplotlib.pyplot as plt
    from tqdm import tqdm

    from google.colab import drive
    drive.mount('/gdrive')

    Mounted at /gdrive
```

Fig2: Importing required libraries

We used 'nlp = spacy.load('en_core_web_sm')' to train the system to understand the basic English language. This specific piece of code is crucial to the success of the project. A knowledge-based graph cannot be created by the system without understanding of the English language.

```
from spacy import displacy
nlp = spacy.load('en_core_web_sm')
```

Fig3: Loading English language into system

We then went ahead and loaded the dataset. A knowledge graph will be created by the system from this dataset.

```
[ ] #for WikiDataset
    candidate_sentences = pd.read_csv("/gdrive/MyDrive/wiki_sentences_v2.csv")
    candidate_sentences.shape
```

Fig4: Loading dataset

In order to determine whether the system is functioning correctly or not, we took a few samples from the dataset. After that, we manually parse a sample line to check how LLM works. We learned that the LLM breakdowns sentences into single words.

```
[ ]  doc = nlp("the drawdown process is governed by astm standard d823")

     for tok in doc:
       print(tok.text, "...", tok.dep_)

     the ... det
     drawdown ... amod
     process ... nsubjpass
     is ... auxpass
     governed ... ROOT
     by ... agent
     astm ... amod
     standard ... compound
     d823 ... pobj
```

Fig5: Output of the sample sentence

The function get_entities is now created, and its fundamental purpose is to extract entities from sentences. Then, to tokenize the sentences, we build a loop for extracting tokens from the sentences. To do this, we have established a few keywords that must be taken from the phrases, such as compound punctuation, complex sentences, etc. In order to extract entity pairs from the sentences, we developed a dictionary of entity pairings.

In order to retrieve relationships between two entities, we created the method get relation.

After all of these extractions, the system will link words to produce a relationship between two entities, similar to the one in Fig. 1, but because we are using a dataset that has a vast amount of phrases, it will now produce a knowledge graph that includes all entities. The knowledge graph will include all of these interrelated entities from various sentences in the event.

V. Outcome

When we look at the outcome of the project and it is seen that as needed, the model is able understand different sentences that is fed into the SpaCy large language model and can derive context or at least find out words that can be match between sentences and understand how it is related to each other. The large language model uses its pretrained understanding of the English language and characterizes words for a given sequence, we use this experienced model to figure out how to combine difference sentences to be joint. Using different formulations of joining sentences and finding relationships between sentences using a network model to be able to find sequences of sentences that can be joint together to form a mesh like structure that can be retrieved later for future purposes. Below given is the combination of all the relationships that can be formed for a particular given sequence of sentences.

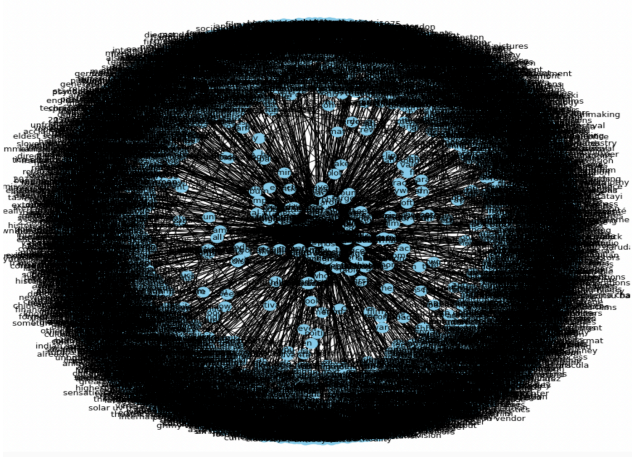


Fig6: Final mesh outcome

As seen from above, the mesh created from all the relationships that can be derived from entities and then overlapped together. When given a specific query, our model can correctly extract the correct sequences of graphs for an all the sentences. (Below shows query for ‘Composed by’)

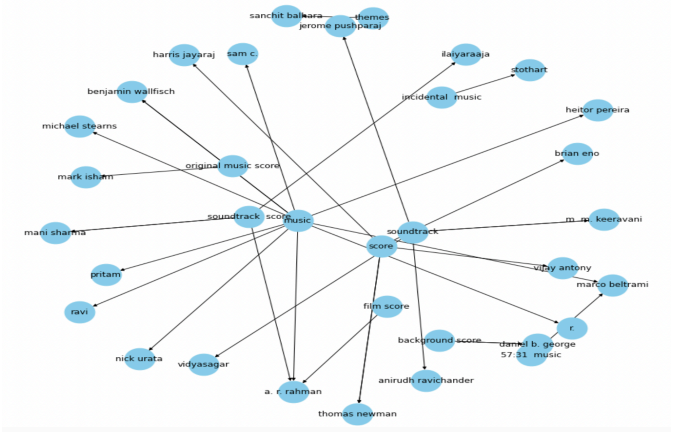


Fig7: Query to knowledge graph for composed by

VI. Conclusion

The purpose of our project was able to showcase deriving context from a sequence of sentences and find relationships between them. Once that is achieved, we use these relationships to create edges of relationships between each other. We were able to achieve our desired result and was tested even on the SocialQA dataset for references and was able to understand questions. To create more feasible and more concrete model we recommend using Bi-LSTM with BERT to remember words that was inputted before, this would enable the algorithm to derived better context for longer sentences.

VII. References

i. Common sense transformers for automatic knowledge graph construction
ii. Knowledge graph construction for zero-shot common sense question answering
iii. Contextual topic modelling for dialogue system
iv. Towards Generalizable Neuro-Symbolic Systems for Commonsense Question Answering

v. Social commonsense Reasoning with Multi-Head knowledge Attention
vi. Learning to Generate Multi-Hope Knowledge Paths for Commonsense Question Answering
vii. Unsupervised Commonsense Question Answering with Self-Talk
viii. Dataset link: https://drive.google.com/drive/folders/1qqAetCTCzFuWG45ClnAMGL7ETS4u6SIn?usp=sharing
ix. Github link: https://github.com/hraoo/ece-5831-project