The Australian National University
2600 ACT | Canberra | Australia

Australian National University

School of Computing

College of Engineering and Computer Science (CECS)

Fact-aware Language Modelling

— 24 pt Honours project (S1 2022)

A thesis submitted for the degree
Bachelor of Advanced Computing

By:
Zhen Wang

Supervisors:
Dr. Pouya Ghiasnezhad Omran
Prof. Kerry Taylor

November 2022

Declaration:

I declare that this work:

- upholds the principles of academic integrity, as defined in the University Academic Misconduct Rules;
- is original, except where collaboration (for example group work) has been authorised in writing by the course convener in the class summary and/or Wattle site;
- is produced for the purposes of this assessment task and has not been submitted for assessment in any other context, except where authorised in writing by the course convener;
- gives appropriate acknowledgement of the ideas, scholarship and intellectual property of others insofar as these have been used;
- in no part involves copying, cheating, collusion, fabrication, plagiarism or recycling.

November, Zhen Wang

Acknowledgements

I want to thank both of my supervisors, Pouya Ghiasnezhad Omran and Kerry Taylor for their support during my honours project. I want to especially thank Pouya for choosing this project topic and provides some inspirations along the way. It goes without doubt that it is quite a challenging project. The scope of it can easily form a PhD thesis. Lastly, a massive shout out to myself, for hanging in there to the end of the journey.

Abstract

This project established itself in the field of intersection between natural language processing and knowledge graph. It can be viewed as preliminary approach of doing factual language modeling, that primarily utilize the factual information from knowledge graph to improve the pretrained language models in a unified way. We have seen the trend of natural language processing are dominant by the rising of various language models, for example the powerful model of GPT-3 can achieve good performance in zero-shot manner across many downstream tasks. While people are astonished at what it can do, we actually know so little that deep down why and how GPT-3 can do these. People are also concerned that after GPT-3, increasing the train data size robustly may no long work, since it is already the largest model ever built. The largest version of it ended in 175 billions parameters. We are throwing an vital question that how can we do beyond fine-tuning on these language model. While previous work have proposed the mechanism of controlling the language model's behaviour either by accessing the weights information by supervised manner or modify the way knowledge masking with injection in foundational layers of language models, we take a long-way detour to fine-tune the language model with reinforcement learning on factual decision made by supervised pipeline. Unlike other approaches, integration or utilizing weights are language model specific or downstream task limited. It provides a framework for unified factual boost on any language model.

Abbreviations

This list describes the abbreviations used within this thesis

ML Machine Learning

NLP Natural Language Processing

CV Computer Vision

KG Knowledge Graph

LM Language Model

TRL Transformer Reinforcement Learning

BERT Bidirectional Encoder Representations from Transformers

RNN Recurrent Neural Network

LSTM Long Short-Term Memory Network

GNN Graph Neural Network

GCN Graph Convolutional Networks

GPT Generative Pre-trained Transformer

BPE Byte Pair Encoding

MLM Mask Language Modeling

NSP Next Sentence Prediction

NER Named Entity Recognition

EL Entity Linking

ED Entity Disambiguation

FC Fact-checking

URE Unsupervised Relation Extraction

PTM Pre-trained Language model

HMM Hidden Markov Model

CRF Conditional Random Field

PPO Proximal Policy Optimization

KL Kullback–Leibler

RDF Resource Description Framework

KGC Knowledge Graph Completion

CNN Convolutional Neural Network

KRL Knowledge Representation Learning

Table of Contents

Introduction

The research area of this project lays in the intersection of natural language processing and knowledge graph. Generally specking, there are two directions of research work in this regard. One is using the rich information contained by the pretrained language model to assist the completion, linking or other fundamental tasks on formation of knowledge graph. The other is to use the graph-structured data from knowledge graph as controlled data source to improve the language understanding of language models. An example direction is to design multi-hop reasoning mechanism on knowledge graph and assist it to improve language model's performance on question-answering. Another well-explore direction in this domain is to use graph information to modify the base layer of language model(BERT) that enables it to have more understanding of domain-specific task. To give an concrete sense of example research work, Yasunaga et al. (2021) propose the model of QA-GNN that use LMs to do relevance check so as to extract and shrink the sub-graph instance from KG when answering the multi-choice question.

The latest generation of language model, GPT-3, has shown unparalleled linguistic understanding. Recent years, researcher try to do more beyond fine-tune LMs on downstream task and grow more intrinsic understanding of these giant model. It is acknowledged that robustly increasing the training set will now improve the performance of language model after the generation of GPT-3. Also inspired by the ethical point of view, people are eager to seek an efficient approach to constrain the behavior of language. In this project, we make a preliminary contribution to make the language model to be aligned to factual purpose. We want to pose constrain on language model so that it can only produce truthful and nontoxic output. Unlike the genre of integrating the knowledge graph information into the language model directly, we take a long-way detour that using reinforcement learning to refine the language model. The trained reward model would mimic the behaviour of supervised factual model to produce output with higher degree of fact. The supervised factual model can be viewed as baseline model that deterministically determines how factual one sentence is. The design of such pipeline in-

volves multiple domains, from entity recognition, entity linking, entity disambiguation, fact checking and relation aggregation. The reinforcement learning provides a general framework of constraining language model with proximal policy optimization.

The main contribution of this project is as follows:

- Design a pipeline that deterministically determine the factual information on arbitrary text
- Underlying the fundamental work for unified framework on improving language model's output in a factual manner

Background

2.1 Problem Background

There is arguably demand that we should be able to contain the actions from AI into legit field as how powerful these AI agent can be nowadays. Depending on the complex environment it perceives, the solely goal of the agent is to take actions to achieve its predetermined goal or improve its performance. Well-known agents like Alpha-Go in board game content and Siri in audio assistant field, people are amazed at what these agent can performance. With people's imaginary limitation are continuously refreshed by these powerful agents, some unusually behavior occurred that is beyond simple explanation. In this research project, we mainly focus on the sub-field of Natural Language Processing(NLP), where it is concerned with the interaction between computers and human language. The core of NLP is to make computer has the understanding of the content of documents, both contextual environment and semantic meaning, from large amount of language data. The ultimate goal of NLP, regardless of different downstream tasks, from recognition, text classification to generation, is to become self-aware of the content that it reads on.

In the long history of NLP, it was not until 2000s that increasing abundant amount of language data become reachable by the help of internet, machine learning techniques like representation learning and neural network become the main method in natural language processing. In 2010s, the emergence of state-of-the-art language model that being able to set amazing benchmark in various downstream tasks. We can say that the LMs have some understanding of grammar and semantic meaning that language carries. All of these, from ancient ELIZA to nowadays prompt-based models, it all starts from how to encode word in a meaningful way, i.e. word embedding.

For word embedding, it starts with the most trivial way Bag of words(BoW), which only describes the occurrence of words in a given text range. It suffers from vocabulary size

and sparsity issues. Simply counting words cannot model the polysemy and homonymy instance, where words can have various meanings determined by their surrounding text. In short, it cannot capture the semantic meaning of words in context. From the initiative that we want the vector representation to have some semantic meaning, classic variant word2vec come into place, both skip-gram and CBoW. They are shallow neural network models that predict a word within a large corpus. Following research works appear in multi-sense Skip Gram that they aim to tailor the number of senses of each word in an embedding that has efficient training time. Apparently, the larger corpus size will result in better performance. The spring of lexical databases like WordNet combines the model with prior knowledge of semantic relationships within one language. However, the limitation is evident in the interoperability between various languages. Also, it has to deal with word sense disambiguation problem that usually requires to be solved with supervised methods(constraint by low resource in prepared corpora) or unsupervised methods(MSSA). Depending on the specific task, we usually would address it with a tree or graph to emphasize its syntax structure. And as for modelling documents, document-level relation extraction that reads in a set of sentences in a document and extracts entities to form representative relations. Not much research has been done in this regard, while Yao et al. (2019b) created a DocRED that annotates documents by its intra- and inter-sentence relations.

The shine of NLP all comes from the transformer architecture (Vaswani et al., 2017). Based on the transformer architecture, a set of famous models has been proposed, BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), ElMO (Sarzynska-Wawer et al., 2021) and GPT-2 (Radford et al., 2019). Language model was meant to estimate the probability of words appearing in a sentence or how likely is a sentence appearing. In order to train and test a language model for a specific task could be time-consuming and tricky to collect domain data. Those massive deep learning models are the gamer changer that they have pre-trained on enormous amount of data and fine-tuning them for downstream tasks is easy to achieve with smaller dataset. Compared with RNN models, RNN can only process words in a sequenced way and does not grasp contextual relationships in long text. Whereas transformer using the mapping across all the potential words dependency that are parallelizable.

While the two most popular models, BERT and GPT-3 are both capable of doing fantastic jobs, they have different structure property. Starting by talking a little bit of Bidirectional Encoder Representations from Transformers(BERT), it has become sensational ever since it launched by Google in 2018 and countless experiments has been conducted on BERT. While it made a stir in various benchmarks, the variant of BERT is spring up in the following years. The main contribution of this model is the novel masked language modeling that enables the bidirectional training to happen with transformer. The model does not read the input sequentially, either from left-to-right or right-to-left, but it reads the entire input at the same time. Just like any pre-trained model, the model is pre-trained on a large corpus. As a comparison, the openAI GPT or other pretrained LMs before Bert, only build the model that can only attend the

later sequence. The general task they used for training is the basic language modeling task, that is predict the next possible word given all the prior words, while BERT used Mask Language modeling for training. So the traditional conditioning language model training cannot be applied to BERT. It is clear for us to see now that it's somehow natural to use mask language modeling and next sentence prediction(NSP) to train, but it was considered novel in 2018. As what is stated in the original paper (Devlin et al., 2018), downstream task like question answer and natural language inference are mainly considering the relation between sentences. This is what traditional language modeling cannot achieve. According to the figures in the paper, just through simple construction and fine-tuning, the model achieve state-of-art results in 11 downstream tasks at that time, that out-runs the other models with a surprising improvement.

Mask Language model(MLM), some words in the sequence is masked with [MASK] token, that the model will try to predict the masked word based on its surrounding text. The prediction is made with a classification layer after the encoder output, where the output vectors are multiplied with embedding matrix resulting into a series candidate words. Each word will be attached with a probability by softmax. For a case with multiple sentences, a [CLS] token is used to identify the start and a [SEP] token is used to identify the end. As for the next sentence prediction(NSP), it is trained with supervised learning. Half of the time, two consecutive sentences are given, and the true label should be IsNext, while rest of the time, a random sentence from the corpus is attached as the second sentence, labelling as NotNext. It is intuitive to see that constructing such a supervised dataset should be easy if given a corpus.

A bit more about the actual input representation, the input embedding consists of three parts, the token embeddings, the segment embeddings and the position embeddings. The segment embedding is straight-forward as it is meant to separate the two sentences. Then the position embedding is somehow similar to what the positional encoder is in the transformer. It keeps track of the word position in a sequence. To be note that the hashtag means the split of word-piece embedding method.

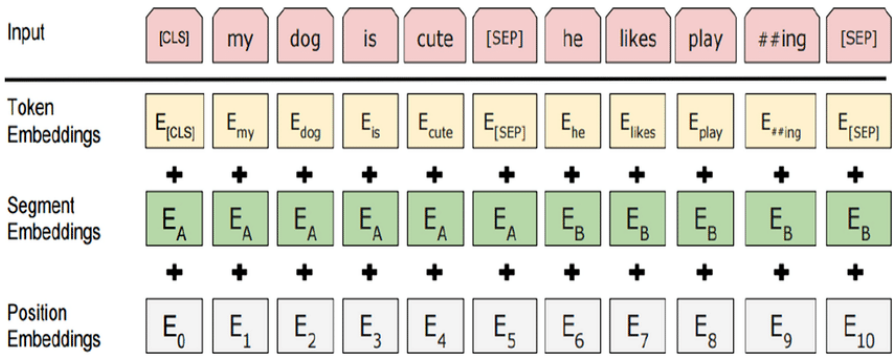


Figure 2.1: BERT Input representation. Figure from (Devlin et al., 2018)

When it comes to the GPT-2, the main contribution of it is to construct a language model that can perform downstream tasks in a zero-shot environment, that is doing

them without fine-tuning. Multitask learning has been the focus of NLP, since the dominant approach requires a re-training or modification as a generalisation method when fitting another task. The main stream is to train a single task on a single domain dataset. Even BERT, released around the same time, it still requires fine-tuning or a slight model reconstruction when dealing with downsteam task such as NER and so on. To make the model capable of handling zero-shot tasks, they constructed the model so that it can handle some downstream tasks in a language model way. To be more specific, take an example in the paper, if parsing the translation training example $(translatetoFrench, Englishtext, Frenchtext)$, there must be instances of naturally occurring demonstrations of translation from English to French. Constructing a corresponding unsupervised approach is feasible. It exemplifies that modeling in a language model way avoid making high-level modification on PTMs. It also made the point that instruction of the tasks can also be recognized as text input. Such construction setting took place in other areas of GPT series models and follow-up research works.

Furthermore, they did not use the word-level input representation as BERT, instead they used Byte Pair Encoding(BPE). BPE is a method of subword-based tokenization, that is trying to break long and unusual words into sub-words with high frequency. Essentially, it combines the benefits of word-level encoding and byte-level encoding. BPE is doing replacement on a byte level that is compressing all the byte pairs appearing more than once. Although the model is basically following the routine of previous generation, GPT, a few modification on the layer normalization, weight scaling and expanded vocabulary size was made. Another aspect is that they considered the fact that the data quality of the training dataset that is called WebText. They created a new means of crawling online data, referred as web scrape. It only scraped web pages from Reddit, whose links received at least 3 endorsements by human.

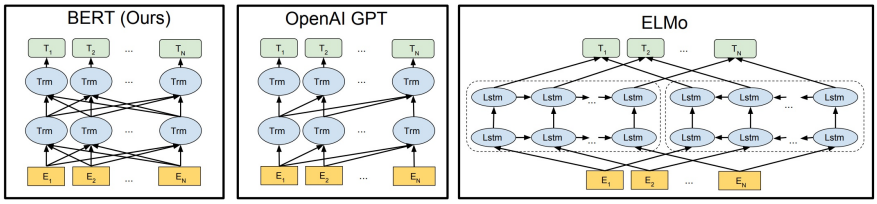
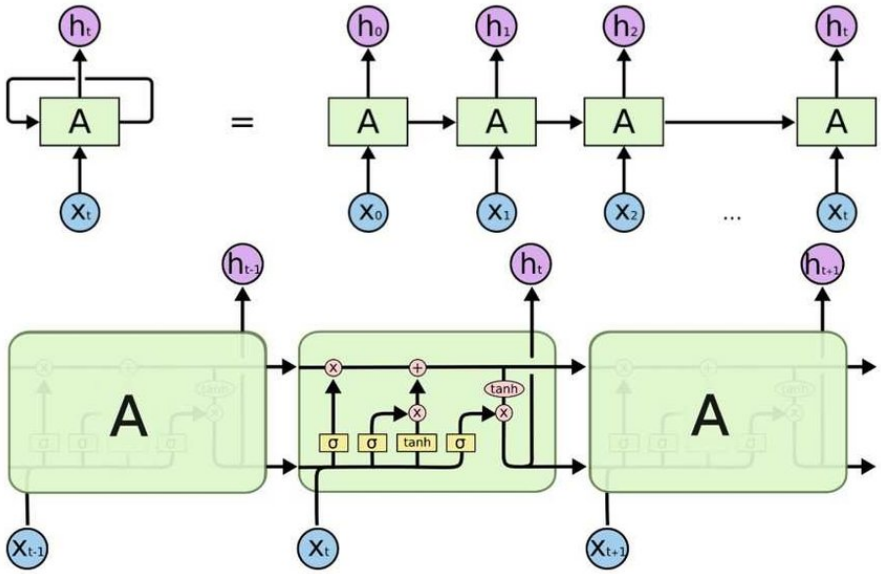


Figure 2.2: Comparison BERT vs other PTMs. Figure from (Devlin et al., 2018)

As for ELMo, this has been intensively used for word vector construction in the earlier stage. It can be viewed as solid improvement of word2vec(Mikolov et al., 2013a). We know from word2vec approach that a vector representation of a word is static, meaning that the representation is unchanged for a given word. The embedding vector of word2vec is a mixture of multiple semantic meanings of the given word. Usually, a word could have different interpretations that is determined by its surrounding context. This is foundationally crucial in NLP with respect to the embedding concept. For example, the word trump, which could mean a person name or a building name in semantic. The issue of word2vec is that it knows that trump can be both person or building name,

but it does not distinguish which semantic meaning to use based on its surrounding context. The improvement of ELMo deals with such drawback that it outputs context-aware representation of a word. ELMo taked use of two LSTM components. The left LSTM component represents the left-to-right language modelling that is to maximize the probability of the current word given the past sequence. Similarly, the right LSTM component represents the right-to-left language modelling that is to maximize the probability of the current word given the future sequence. So, the EMLo would result in a joint training objective from two directions. The main issue is the concatenation of two split LSTMs that it could a shallow combination.

RoBERTa is exploring design choice of BERT, while preserving the general architecture. In natural, it can be considered as successor of BERT. RoBERTa used a much larger training corpora than BERT like CC-NEWS and OpenWebText. The usage of next sentence prediction can be viewed as to preserve long-term relations between sentences. What's more, they compare with BERT and set optimized hyper-parameters during training. RoBERTa used a dynamic masking, which is considered improvement of BERT's static masking. The masked token is generated for each input sequence, that has shown some marginal improvement on question answering benchmark. Moreover, they studied the impact of including NSP or not with different input formats. They also investigated the impact of batch size from 256 to 8k, that concludes the 2K batch size is best. They tried to replace BPE with word-level encoding and the result they claimed is that character-level and word-level encoding do not make a significant difference comparing with BPE. With these investigation results, they re-train the BERT following the optimized strategy. As for evaluation schema, it compares the result against standard benchmark on various downstream tasks, which is similar to what was done on BERT. For any pre-trained model, evaluating against benchmark over downstream tasks is essential to demonstrate its pre-trained language capacity.



Source: https://www.researchgate.net/figure/RNN-and-LSTM-comparison-chart$_{f}ig3_{3}32662013$
Figure 2.3: Comparison LSTM and RNN

GPT models are unidirectional and the strategy behind is modeling a conditional probabilities of the ordering of the natural sequence. The computation of these conditioning is largely improved by the self-attention. The strategy behind the GPT-3 are simple but robust, that is to increase the volume of data of training. The main data they used, Common Crawl, has 410 billions of tokens. Not even to mention the other minor parts, such as WebText2 of 19 billion and others. The very generation of GPT-3 took billions dollar to train and they almost fetch everything on Internet as training resources. They used the exact same model and architecture of GPT-2 but with larger and wider layers inside and training only with next word prediction. It is not hard to imagine that the side-effect come into play when the model size is this large. Considering a neural network on words, the learned weights is a function of words. The weights of the network is what really matter in a sense that it captures the essence of training data. In such a giant model, weights set of this size, people may argue that a lot of training data is stored in that weights rather than learning from it. When given a prompt and relevant text description, the model would likely to filter the relevant context and interpolate these training examples to generate the output. It would be interesting to learn how much the model really learn rather than filling giant training data into weights. Take an in-proper analogy, if we take large language models as the model that learns human language, it is not required for human to review that sufficient amount text in order to gain wisdom over language. Wisdom can be constituted by the ability of conducting judgement, reasoning and analysis of context.

From the below smooth scaling of language models, we can see that until now if the parameters go up, and the computation time is scaled in a power fashion as well as the data size, the performance does improve. But how far can we go with the extending the massive model, until the performance does not improve any more.

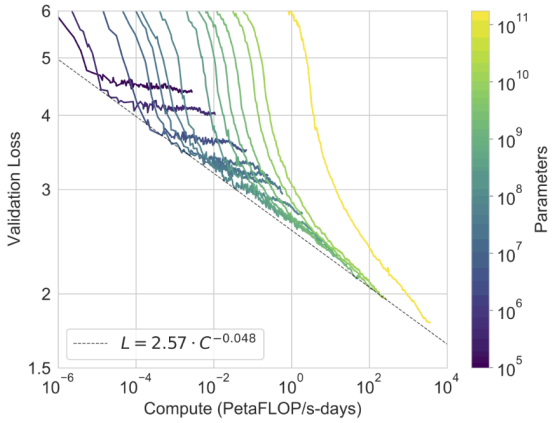


Figure 3.1: Smooth scaling of performance with compute. Performance (measured in terms of cross-entropy validation loss) follows a power-law trend with the amount of compute used for training. The power-law behavior observed in [KMH+20] continues for an additional two orders of magnitude with only small deviations from the predicted curve. For this figure, we exclude embedding parameters from compute and parameter counts.

Figure 2.4: Scaling of Performance with LMs. Figure From (Brown et al., 2020)

The largest GPT-3 model ended in 175 billion parameters that only small amount of fine-tuning dataset is required to toggle the model for down-stream tasks. People are amazed at what GPT-3 could accomplish these days, that it not only can grasp the language, but also seems to be self-aware in a sense that it can say unexpected things. The quality of the text generated by the model is so high that it is not hard to tell if it is from human or GPT-3. The fact that the GPT-3 are such capable is the risk for the human being. It could be easily be used for harmful application including misinformation, spam and fraudulent writing. While people are astonished at how the GPT-3 could do what it can do, people are actually knowing so little that deep down why GPT-3 can do these. Robustly increases the training size is the key that the model gains such wisdom, but why it can grow so much and what part play a what role inside the model architecture is somehow a mystery. People are somewhat unaware of what they have built but rush to stack things high to push the results.

According to Bender et al. (2021), the concern for increasing large language model has been drawing people's attention. From 2018 and afterwards, the spring of larger and giant language model, BERT and its variant, GPT series and others, is characterizing the recent era of NLP development. To be more specific, the concerns comes from these directions. One is that there is no way to guarantee the train data to be fathomable. Often the case is that the ethnically concerns are involved in training. There is no measures taken to ensure that the large amount data from internet that these model learned is ethnically safe. Another concerns is that the increase of size does not necessarily leads to positive effect. Take GPT-3 as an example, crawling petabytes of data from online means that we have absolute no control that the data is pure and balanced. Bender et al. (2021) claimed that the voices of hegemonic viewpoint is more likely to retain a main position, which means that bad things such as harmful speeches, white supremacist, ageist and so no in English environment, tend to gain its spot in the language population. It is

very likely that the dataset can be amplifying with biases and hate. This means that the increase of population size has no benefit for diversity purpose. The balance between male and female are corrupted given that the main source, Wikipedia that is mainly contributed by male contributor's. These concerns lead to a vital conclusion, a means of controlling the behavior of LMs is actively called for, since we have no control or easy grasp of larger web source content.

The most recent trend in NLP research lies in prompt-based learning, where the basic idea is to utilize the knowledge from the pre-trained model like BERT/GPT3 to solve downstream tasks. It is promising in a sense that a pre-trained language model can be used to predict the desired output without fine-tuning. A simple prompt is to insert a text to transfer the original task into a masked language modelling problem. The recent work (Ding et al., 2021) proposed a unified toolkit to conduct prompt-learning over pre-trained language models. Yet things like how the setting of text template and pre-designed answers(labels) will affect the language model and how the combination of downstream tasks will affect the mechanism remain open questions. In the sentimental analysis task, if a particular model always predicts negative words over positive one, how does such a bias come into play is worth discovering.

2.2 Research Goal

An question that has been asked by a lot of researchers after the launch of GPT-3 is that what can we do beyond normal fine-tuning other than stacking volume to building a behemoths PTMs.

We are at a crossroad. There are several direction that seem promising after the emerging of GPT-3. One direction is considering improve the architectures of PTMs such that it could be deployed to low-capacity device. Various downstream tasks have its preference of LMs, meaning that some architecture is especially effective on a separate task. It is not hard to see in the near future, more task-specific model and variant of transformer would be proposed. Another direction to do beyond fine-tuning is prompt tuning. It is considered another type of model tuning. The common approach is to add more modular layer onto PTMs, so that it can achieve good result on multiple downstream tasks. It has quite promising future by some recent work, such as (Wang et al., 2021b), (Han et al., 2021) and (Lester et al., 2021). Prompt tuning is a bridge between unified LMs and computational affordable downstream tasks solution. The last direction is to study the representation of knowledge inside the LMs. Compared to discrete symbolic knowledge form, like triple form in knowledge graph, the knowledge from weights in LM is much confusing to understand. It is often expended to the research of knowledge-aware tasks, which generally is about the interaction between knowledge base and language model. For example, knowledge base can support LMs with structured data source and LMs can help knowledge graph completion tasks. The direction of knowledge-aware tasks will be extensively reviewed in the following related work section and. The scope of this thesis is related to this direction. In a summary, the goal behind these directions is to study

reliability, efficiency, adaptability and linguistic capacity of PTMs.

Why is this important? We know that making language models bigger can indeed improve its language understanding. But it still cannot follow a user's intent as we know so little of how it learns intrinsically. Without proper supervision, LMs can fall into negative usage. For example, large language models can generate outputs that are untruthful and even toxic to the user. My research question would be how we can manipulate the behaviour of Language models that can only produce truthful outputs.

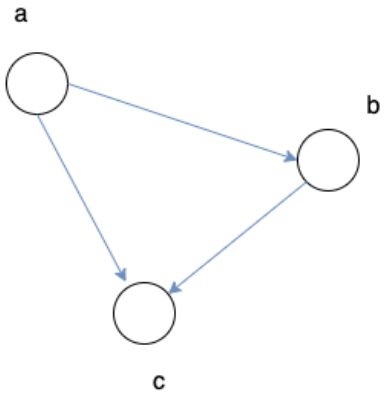
There has been prior work that makes the GPT-2 to be aligning to human needs. The architecture of transformer reinforcement learning(trl) is first introduced by OpenAI in 2019, it is simple but effective in capturing the human feedback in summarization and continuation tasks. What we want to do is to extend the current work, since the process of collecting human preference is extremely painful. It also is expensive to corporate a group of people working closely to the researcher team to just giving taste of the data. Our idea is to extend the transformer reinforcement learning architecture to get its reward from a knowledge graph. When the language model produces an output, such output can be assessed by different sources and generate a reward accordingly. We want to use knowledge graph to build a factual pipeline to evaluate how truthful the output is and give a reward according to the factual information the output has. On a bigger picture, this project will serve as a preliminary approach towards controlling of language models in a truthful manner.

2.3 Graphical Model

This section will served as a basis of the definition and modeling using graphical model, which is well studied in the past. This will be the background to understand the conditional random field in methodology.

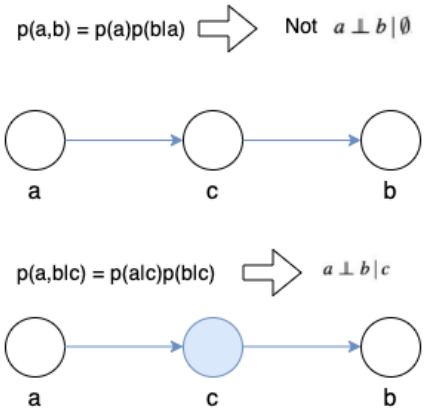
As always, we start with basis and dive in gradually. A graph $G = (V, E)$ in graph theory are comprised by nodes and edges. The graph can be further categorised into directed, undirected, cyclic and acyclic. The intuition of hidden markov models(HMM) is that the nodes are modeling random variables and edges are representing probabilistic relation between these random variables. The directed graphical model are known as Bayesian networks. Intrinsically specking, it is a simple yet powerful tool to modeling the joint distribution of multiple random variables. For example, applying product rule of probability we can see that $p(a, b, c) = p(c|a, b)p(b|a)p(a)$, where its graph depicts exact relation below. We can confirm from the graph, that variable c is dependent on a and b. variable a is indeed contributing to b and c. Generally, we can reproduce this trick over any joint distribution of K variable $x_1, ..., x_k$, leading to the conclusion of $p(x_1, ..., x_k) = p(x_k|x_1, ..., x_k)...p(x_2|x_1)p(x_1)$. Equivalently, we can apply this trick reversely, that is we can write the joint probability equation by looking at the graph. With additional assumption of the graph being acyclic, we can keep decomposing the joint probability with looking for its parents. Let pa_k denotes a set of parents of node

x_k, we can further summarise this trick with $p(x) = \prod_{k=1}^{K} p(x_k|pa_k)$.



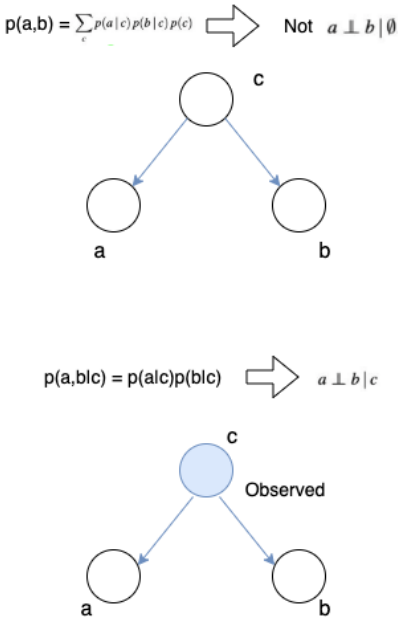
Source: original
Figure 2.5: example directed graphical model

Apart from joint distribution, we are exposed to another elementary type, conditional distribution. When the variable a is conditionally independent of b given c, we can apply product rule to get $p(a, b|c) = p(a|b, c)p(b|c) = p(a|c)p(b|c)$. This is referred to the concept of variable a and b are statistically independent given c as $a \perp\!\!\!\perp b|c$. From this, we can case discussion the three scenarios of whether the conditioned variable is observed or not. The first case is known as c being the head-to-tail node, as shown in the below graph.



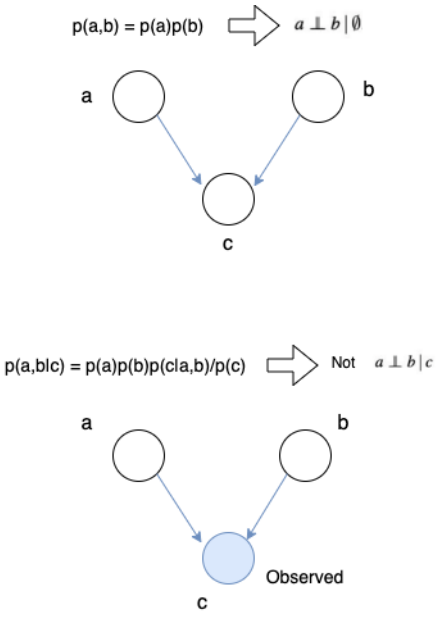
Source: original
Figure 2.6: example directed graphical model

The second case is known as c being the tail-to-tail node, as shown in the below graph.



Source: original
Figure 2.7: example directed graphical model

The third case is known as c being the head-to-head node, as shown in the below graph.



Source: original
Figure 2.8: example directed graphical model

What if we cannot identify pattern as one of above three structure, we furthermore

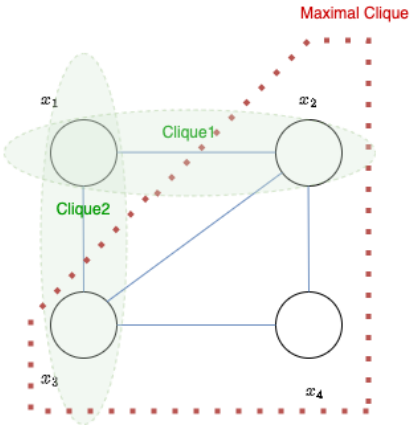
introduce another fundamental theory of D-separation that allows the pattern above to be generalised on any Bayesian network. The idea is to interpret the graph connectivity with conditional graph patterns.

Theorem (D-separation)**.** *If two nodes a and b are independent and conditioned on any nodes in a directed acyclic graph, where any of the path from a to b is blocked when the either of the two cases are satisfied. A general version can be consider A, B and C to be three non-intersecting sets of nodes. All the possible path from set A to set B are constraint by the following cases. When any of these two cases are met on all the paths from A to B, then A is d-separated from B by C. One is that the path contains a node that is head-tail or tail-tail and the node itself is in set C. The other is that the path contains a node that is head-head and neither the node itself nor any of its descendants is in set C.*

With this tool in hand, it acts as the bridge between any directed graph and factorisation. We already seen from above content that directed graph can be decomposed into a joint probability statement, which could be further simplified with conditional statements. But the d-separation actually bridge the conditional statement with simplification from the graph. One way to visualize this feature is to think that we are started off by a joint probability $p(x)$, where defined on a set of unobserved nodes with respect to variable x. The satisfied ending we want is a directed factorization $\mathcal{DF}$. If and only if $p(x)$ can be accepted by two filter, we say it is true factorisation. One filter is that it can be decomposed into the exact from the graph implies by $p(x) = \prod_{k=1}^{K} p(x_k|pa_k)$, the other filter is that it can decomposed into the exact conditional statement from graph by d-separation.

Now, we have seen interesting features of factorisation of directed graph. We introduce another genres of undirected graph, Markov Random Fields(Markov network). It only differs from the above graph that the edges of these do not have arrows. An fundamental character from these undirected graph is that the status a node only depend on its neighbours, which is rather intuitive. We refer it to be the concept of locality. Formally, if two nodes x_i and x_j are not connected by one link, it implies that x_i and x_j are conditionally independent considering the rest nodes. This is expressed in $p(x_i, x_j|x_{\setminus\{i,j\}}) = p(x_i|x_{\setminus\{i,j\}})p(x_j|x_{\setminus\{i,j\}})$. It confirms the fact that if there is no direct path from x_i to x_j, the consequence is that all other paths are blocked with traversing other observed nodes. With this observation, we can define the concept of clique that specify the units of locality. A clique is a set of nodes that is fully connected. Thus the concept of a maximal clique is just the set of nodes that cannot be added into another nodes to maintain it is still connected. The definition of cliques in illustrated in below graph. In this particular example, the maximal clique is $\{x_2, x_3, x_4\}$. Let us denote this clique to be clique C, the set of variable C describes is x_C. So the factorisation of $p(x)$ specified by this clique is given by joint of potential function $\psi_C(x_C)$, which is $p(x) = \frac{1}{Z}\prod_C \psi_C(x_C)$. The term Z is another normalisation or called partition function, which we should be familiar by now. It is given by $Z = \sum_x p(x)$. Despite the form of Z, it still can be a disaster when actually normalise against multiple local distribution.

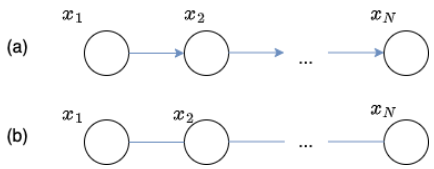
Considering a model with M discrete node and each of K states, normalisation for Z can be exponential to the parameters of the model in the very worst case. Additionally, if the underlying distribution is not completely discrete, but a combination of discrete and continuous, it would be affected greatly by the fact that potential function can sometimes be zero. Remember the bridge observation we just talked about in the directed graph, the similar theory applies here with some modification. To get the same bridge in undirected graph, one can strictly assume that potential functions are strictly positive, that cannot be zero in any case. This exact result of bridge in undirected graph is a well-known theory, Hammersley-Clifford theorem. Thinking graphical model as a filter, and possible set of distributions that implied by the graph $\mathcal{UI}$ is equivalent to the set of distribution that implied by the factorisation rule with maximal cliques $\mathcal{UF}$. With this, we can always find an energy function $E(x_C)$ that forms an exponential representation $exp\{-E(x_C)\}$. Together, we can get the potential function in strictly positive assumption, which is given by $\psi_C(x_C) = exp\{-E(x_C)\}$.



Source: original

Figure 2.9: Clique in four nodes graph

We can summarise this part of directed graph and undirected graph with a comparison of those two types of graph in a linear chain version. The sub-graph(a) below has the joint probability distribution $p(x) = p(x_1)p(x_2|x_1)p(x_3|x_2)...p(x_N|x_{N-1})$ by the factorisation rule of directed graph. Similarly, the sub-graph(b) below has the joint probability distribution $p(x) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{3,2}(x_3, x_2)...\psi_{N-1,N}(x_{N-1}, x_N)$ by the factorisation rule of undirected graph.



Source: original

Figure 2.10: Comparison in Linear Chain example

In the example of linear chain, we can easily see that the notation of clique equals to neighbouring conditional probability, i.e. $\psi_{N-1,N}(x_{N-1}, x_N) = p(x_N|x_{N-1})$ if we choose partition function to be one. In a more general perspective, converting any Bayesian networks to markov random field is called moralisation. The pre-condition is that every variables satisfies the condition distribution is a member of at least one clique in the markov random field graph. It means that for those node having only one parent, simply replace link with undirected link. But if some nodes having multiple parents, we know that we have to connect its parents with each other, so that the parents and their child belongs to a single clique. A direct consequence of doing moralisation is that the result undirected graph can be fully connected in the worst case, and it does not have any conditional independence, which is different to the original directed graph. To measure such a dependency lost in the convert, we can define a D-map if a graph reflects a distribution that satisfy every conditional independence statement. Likewise, an I-map is that every conditional independence statement implied by a graph is indeed satisfied in the distribution. So, a P-map is the perfect graph that meets the condition for I-map and D-map at the same time. A concrete example from above three nodes with head-head unobserved pattern is a D-map.

2.4 Unsupervised Learning and Supervised Learning

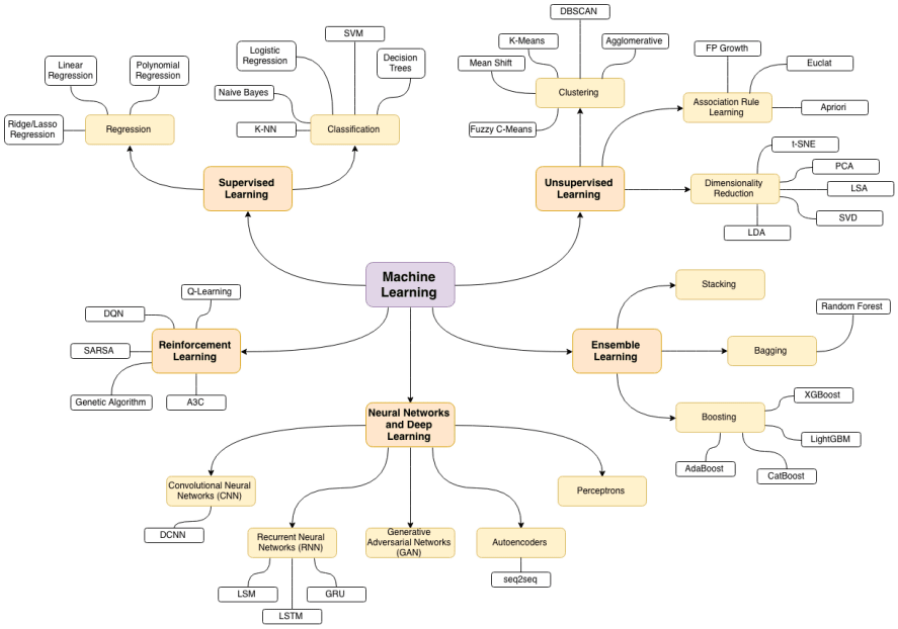
As a start-off, we will talk upon some basis in machine learning in case the intended readers do not have relevant background. It will pave the way for talking about reinforcement learning in methodology.

Regardless of modern advanced learning method, we can judge that the origin of these fancy method proposed get their inspiration from the traditional learning methods. It is important to bear in mind that the machine learning models essential are learning the patterns of your data. No model can achieve good result without carefully tune the dataset. Although in today's academia, it has becoming a trend that well-collected benchmark are released and serving as a mainstream to use in various research direction of ML. But in terms of a more classic view point, we know that raw data has to be structured(integration, transformation, Reduction), preprocessed, analyzed and cleaned. Generally speaking, it is as important to craft your training data as choose and design a learning method. It can be troublesome to taking consideration of the following aspects: bias, noisiness, complexity and interaction dependencies.

The supervised learning(SL) are defined as tasks that tried to map the input to the labeled output. The mapping are learned through algorithm inside analyzing the training data. A successful instance is the algorithm correctly label the unseen input in the validation and testing phrase, while the quality is usually defined with generalization error. Usually, in the simplest form of SL, assuming independent and identically distributed of samples, the training example takes the form of pairs in set $\{(x_1, y_1), ..., (x_N, y_N)\}$. x_i denotes the input feature vector and y_i is the corresponding label. The objective to train a functional mapping $g : X \to Y$. It is often the case that the risk of this learned

mapping is $R_{emp}(g) = \frac{1}{N}\sum_i^N L(y_i, g(x_i))$, where the loss function $L(y_i, \hat{y})$ could take various forms. Some widely used traditional method includes support-vector machines, regression, native bayes, decision tree, K-nearest neighbor, neural network and others. On the other hand, the unsupervised learning seems need more feature design since there is no existing label for correction. It often refer to the type of algorithm that learns the implicit patterns from data. The construction of unsupervised method often involves with some degree of probability density or neural feature. The classic method often used are principal component and cluster technique. In cluster-like approach, group with shared attributes features are exploited to categorize. Another well-known direction is encompassed with density estimation that it intends to infer latent variable and model with probability distribution. Apart from that, the most used and studied unsupervised neural model includes Autoencoder and VAE(variational autoencoder).

It is usually the case in the real life that either the data is completely labeled or the data completely unlabeled. It means that the data captured always have a small portion is tagged. Semi-supervised methods are applied in this scenario. It is special series of methods established in weak supervision. The common strategy is to transfer to supervised or unsupervised by discarding unused data or in transductive setting, it infers the correct label for the missing input.



Source: https://vitalflux.com/great-mind-maps-for-learning-machine-learning/

Figure 2.11: Overview of Existing Traditional ML Methods

Related Work

3.1 Broad Literature Review

3.1.1 Knowledge Graph(KG)

In the past few year, the term of knowledge base is gaining widely attention in the direction of artificial intelligence. The term of knowledge graph can be seen as a graph structure in addition to a knowledge base as shown in below figure. It is well-known that knowledge graph is constituted by a collection of factual triples i.e. (subject, predicate, object) under the resource description framework(RDF). Some may refer triples as (head, relation, tail). A knowledge base, however, can be viewed a collection of triple that captures semantics meaning without graph representations. Numerous advancements have been made across different usage related to KG. We can divide the KG-based research into representation learning, knowledge acquisition, real-world application and temporal knowledge graph as shown in figure. The following context will elaborate each sub-task in rich-detail by the taxonomy from (Ji et al., 2021).

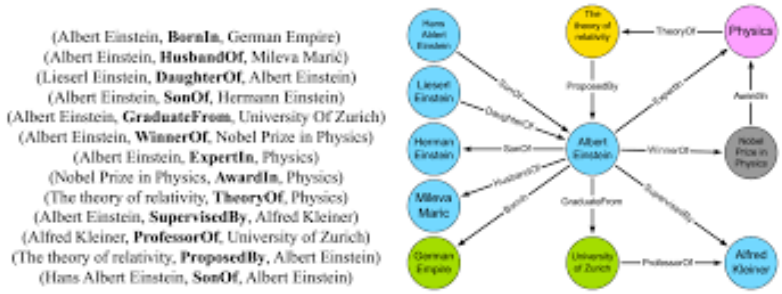
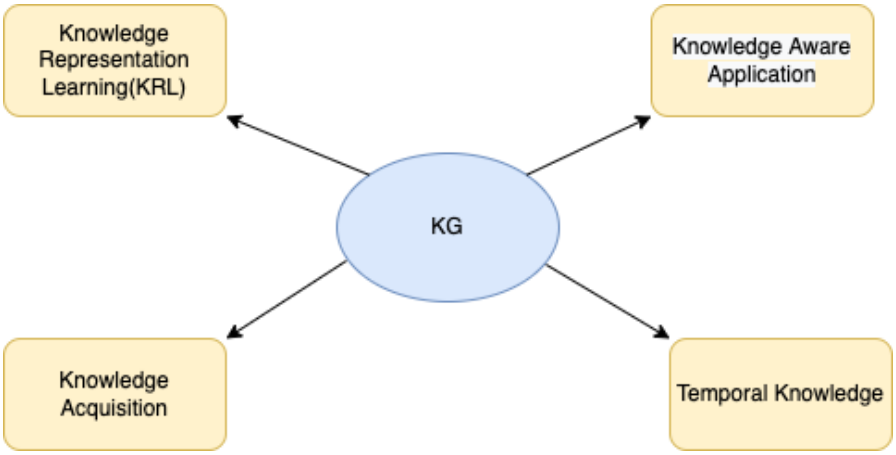


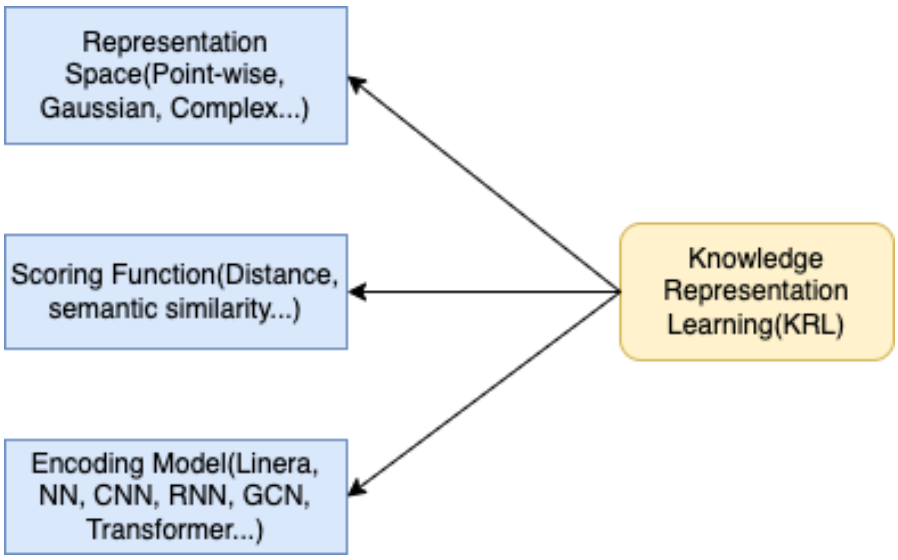
Figure 3.1: Albert Example in Knowledge Base & Knowledge Graph. Figure from (Zuo et al., 2021)



Source: Original
Figure 3.2: Four classifications of sub-tasks related to KG

In terms of knowledge representation learning(KRL), it usually referred as knowledge graph embedding in a sense that it approaches the semantic meaning of entities by mapping into a vector space. KRL has the longest academia history among the four sub-tasks related to KG. The common thread for learning the representation of KG in a embedding space consists of creating a representation space, scoring function and encoding models. KRL section covers a great number of previous work that are brought into discussion due to the completeness of KG section. As most of the previous work in this domain is irrelevant to our work, most of them would be laid out in short sentence. The purpose is to clear out the direction of on-going research of KRL, because it could give out some inspiration that worth further pointed out in future work. Starting with the representation of triples, the mostly widely used is pointwise Euclidean space, where it capture relations by projecting relation embedding in a vector space. The research flow of Trans-based models in pointwise space can be listed as (Bordes et al., 2013), (Lin et al., 2015), (Socher et al., 2013), (Zhang et al., 2020b), (Wang et al., 2014), (Nickel et al., 2016) and (Liu et al., 2017) in logical order. Bordes et al. (2013) approached with the principle that combining head vector and relation vector would ended in a single vector space of tail. To improve the issue of head and relation shares the same vector space, Lin et al. (2015) introduced the projection matrix used on head and tail vector. It would end in a relation space after projection. Similarly goes for the work published by Socher et al. (2013) and Zhang et al. (2020b), where the former modeling the mapping by bilinear tensor layer and the latter used polar coordinate to do the projection. As contrary to the real-value space, the triple could also be shown in complex space by decomposing entities into real part and imaginary part. The flow of Trans-based models in complex space can be listed as (Trouillon et al., 2016), (Sun et al., 2019) and (Zhang et al., 2019b). (Trouillon et al., 2016) proposed to use Hermitian dot product for composition of head and relation, whereas Sun et al. (2019) adopted a rotational model by Euler's identity to composite tail entry. As a improvement of Sun et al. (2019), Zhang et al. (2019b) defined a quaternion with three imaginary components and yield a 4-D spaces mapped

by entities and relations. It outperforms rotation-based model in capturing semantic composition patterns. Some existing works are done in the direction of using Gaussian distributions to model the entities embedding space, such as (He et al., 2015) and (Xiao et al., 2015b). Shifting the problem into a probabilistic view, the mean vector is the possible position of entities and relation, where as the covariance is the uncertainty. The last genre of work could be summarised into manifold space approaches, that receives less attention. Xiao et al. (2015a) proposed both sphere and hyperplane embedding that carries geometric meaning compared with the pointwise model. It solves the issue with simple algebraic representation space that refined the embedding with geometric form, reduces the number of scoring equations. Balazevic et al. (2019) and Chami et al. (2020) made discovery inside the hyperbolic space.

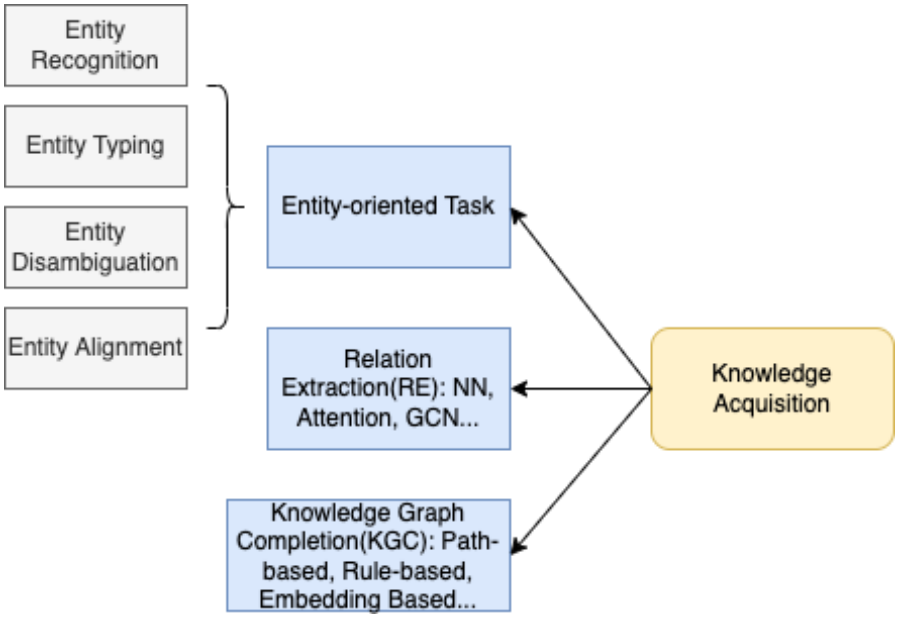


Source: Original
Figure 3.3: KRL breakdown

Following by discussing some of the scoring function. The scoring function is aimed to measure the reasonability of the triple. We can largely divide scoring function into distance-based and similarity-based. The distance-based method usually calculate the Euclidean distance between the projection. Bordes et al. (2013) define the scoring function as $f_r(h,t) = \|head + relation - tail\|_{L1/L2}$, following the additive assumption that the sum of the vector space of head and the vector space of relation should be close to the vector of tail. From this base function, many extension has been sprung up. Different representation space could result in different scoring function. He et al. (2015) used the Gaussian space, thus using KL-divergence and likelihood for Gaussian probabilistic modeling. As a replacement of additive assumption, Ji et al. (2015) constructed dynamic mapping matrix by separate projection vectors. There has been more advanced model, like (Xie et al., 2017). They proposed learning relations interactions through attention vectors. Many work have been done by calculating semantic similarity, such

as (Bordes et al., 2014), (Yang et al., 2014) and (Nickel et al., 2016). The last part of KRL is about encoding model. It is not hard to imagine that there are many ways to learn the interactions between entities by various model architectures. The simplest is to approach with linear-based model. Some members of the family includes (Bordes et al., 2011), (Bordes et al., 2014), (Yang et al., 2014), (Trouillon et al., 2016), (Liu et al., 2017) and (Wang et al., 2018). Due the issue with the independence embedding of entity vector, some work used the factorization method, where tensor χ is decomposed with head vector, tensor matrix and tail vector. The foundation work of this regard is (Nickel et al., 2012) and (Jenatton et al., 2012). Recent focus lies in modeling entities interactions with neural network. It can be roughly divided into these categories: representative neural model, convolutional neural network(CNN), recurrent neural network(RNN), graph neural network(GNN) and transformer-based network. The trend for representative neural model is to feed entities and relations into neural network for a semantic match score as work done by Socher et al. (2013), Liu et al. (2016) and Bordes et al. (2014). On the other hand, CNNs approaches like Dettmers et al. (2018) models the interactions by 2-D convolution over multiple layers of features. RNN-based model are proposed by Gardner et al. (2014) and Neelakantan et al. (2015) that makes use of the long-term dependency in RNN to capture vector representation. GNN approach is another prosperous direction these days that utilise the encoder-decoder framework to learn the graph structure. Relevant work are (Schlichtkrull et al., 2018), (Kipf and Welling, 2016) and (Nathani et al., 2019). Among these model used, transformer-based network is related to our focus that touch upon the cooperation with NLP. It would be discussed in more detail next section. Besides all the review parts about KRL, some research has been focus on how to utilise auxiliary information such as type information ,text description and so on. Entities comes with a text description and has a hierarchical class information. Similarly, relations could have semantic types. Numerous work are accomplished in this regard, following (Wang et al., 2014), (Xiao et al., 2017),(Guo et al., 2015) and (Xie et al., 2016).

Another critical direction of knowledge graph is to complete an knowledge graph and recognize entities. It is often referred as knowledge acquisition. The main tasks of acquisition is knowledge graph completion(KGC), relation extraction and entity-oriented tasks like entity recognition. Because of the nature of KGC it has with other entity-oriented tasks, it often involves some kind of reasoning following rule-based reasoning in recent literature. We can view triple classification or referred as factual classification as an associated task of KGC. Entity-oriented tasks can be break down into entity recognition, entity typing, entity disambiguation and entity alignment.



Source: Original
Figure 3.4: Knowledge Acquisition breakdown

KGC will be reviewed in more detail. A KGC task can be further divided into entity prediction, relation prediction and link prediction, among which link prediction is considered indispensable. Link prediction is defined as infer the missing relation with the existing entities and relations. We can roughly divide the recent work into embedding-based model, rule-based reasoning and relation path finding. KRL method mentioned like (Bordes et al., 2013), (Lin et al., 2015) and (Wang et al., 2014) has been used as embedding-based models to predict entity in completion. In order to capture comprehensive relation path, another genre of work adopted relation path reasoning. Lao and Cohen (2010) and Gardner et al. (2014) investigated the possibility of applying random walk for path-ranking. Excellent work from Das et al. (2016) and Liu et al. (2022) represents the neural multi-hop relational path modeling. Some other works used Deep RL to model the relational path finding as a Markov decision process. Each step agent learnt to take can be viewed sequential decision making. This line of work includes (Xiong et al., 2017), (Das et al., 2017) and (Lin et al., 2018). Next, we will cover the approach with rule based reasoning, where it is usually pointed to logical rule learning in KGC. One type of methods focused on use logical rules in embedding based model, such as (Guo et al., 2018), (Zhang et al., 2019c) and so on. Another type of methods used is to applied with Markov Logic Network that utilizes first order logic for probabilistic graphical models. A prominent example would be Qu and Tang (2019), and the follow-up work (Zhang et al., 2020a) that each logic rule can be trained via variational EM algorithm. The missing triples are inferred by mean-field inference with parameterize the variational distribution being KG embedding space. Lastly, we will cover two rising task, known as few-shot relational learning and triple classification. Few-shot relational

learning is defined for model to predict new relational triple based on only a few triples, including (Lv et al., 2019), (Qin et al., 2020) and (Baek et al., 2020). Entity-oriented tasks will be covered in next section, as it interacts some of NLP domain. Triple classification is also the key point of this project that will be covered in length in later section. Lastly, relation extraction(RE) defined as extracting semantic relationship between two entities from unstructured text and fitted into target knowledge graph. We know that RE is becoming increasing important for automatically building knowledge graph. The existing KGs like Freebase, YAGO or DBpedia are basically constructed manually. Despite the fact that these large KGs have billions of triples, they are still considered to be incomplete. As the current trend is dominated by Deep neural network(DNN) approach, we will list some of the DNN-based models. The CNN-based approach can be listed as (Zeng et al., 2014), (Nguyen and Grishman, 2015), (Zeng et al., 2015) and (Jiang et al., 2016b). Additionally, some works combined LSTM or RNN with CNN, following Xu et al. (2015), Miwa and Bansal (2016) and Cai et al. (2016). Similarly, Shen and Huang (2016) and Lin et al. (2016) used attention mechanism to combine CNN for word-level semantic information. Other variant, sentence-level attention by Ji et al. (2017), hierarchical attention by Han et al. (2018b) and BiLSTM attention by Zhou et al. (2016) are worth noting. Also, adversarial training is applied to CNN-based and RNN-based models for noise of word embedding (Yu et al., 2021) and (Qin et al., 2018a). Other lines of work utilize Graph convolutional networks(GCN) and Reinforcement Learning. Zhang et al. (2019a) used GCN on relation embedding, while Guo et al. (2019) used GCN on the dependency tree with attending edge selection. Qin et al. (2018b), Zeng et al. (2018) and Feng et al. (2018) trained the instance selectors but with different reward selection.

After elaborating KGC and KRL, we will briefly touch upon the third direction that is temporal knowledge graph. The researchers have begun to consider the fact that the triple can change over time into KGC and KRL as oppose to the static assumption of knowledge graph. We can roughly divide the temporal knowledge into information embedding (Leblay and Chekol, 2018),(Ma et al., 2019),(Dasgupta et al., 2018) and so on, entity dynamic(Goel et al., 2020) and (Trivedi et al., 2017), relational dependence(Jiang et al., 2016a) and logical reasoning(Chekol et al., 2017).

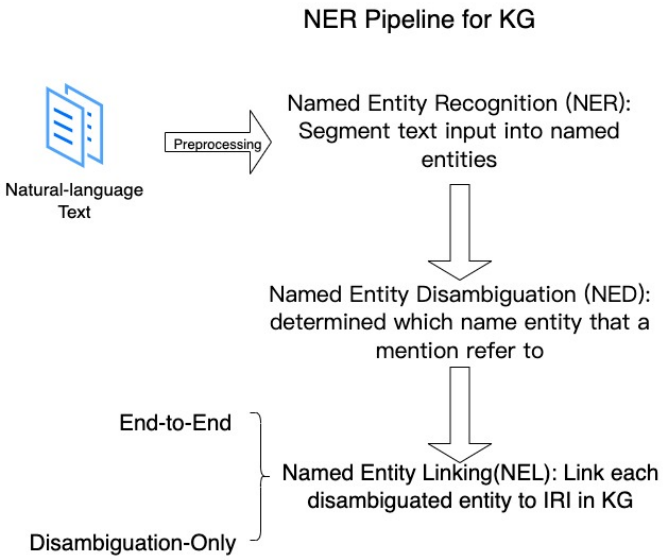
3.1.2 Cooperation With Knowledge Graph(KG)

Due to the rich structure knowledge KG has, it has many downstream application in the domain of artificial intelligence. Within this project's framework, we will be primarily focus on its interaction with NLP. From last section, we reviewed the former three sections of KG, which are knowledge representation learning, Knowledge acquisition and temporal knowledge graph. This section will talk about knowledge-aware application, which is the interaction area. We now can further manifold the interaction of KG and NLP into two categories: 1. in-KG task 2. in-NLP task.

When it comes to the usage of NLP method in the knowledge base, the cooperation front has extend its way in recent years research. Despite the fast pace of this research area, this section will present some cherry-picked aspects in the cooperation. The aim of this

section is to convey some vital aspects and potentials of how NLP and knowledge base can be mutually beneficial without going tiny detail in each sub-domain. In-KG tasks can be described as link prediction, entity linking, transformer-based encoding model and some of the entity-oriented tasks(entity typing and entity alignment). In-NLP tasks can be listed as named entity recognition, triple classification, question answering and relation extraction. It can be confusing to hard-classify some task into one of the two categories, as task like named entity recognition can be referred to entity linking in academia, making it a in-KG task. Named entity recognition could also refer to the preliminary procedure used in KGC.

As a start-off, we will touch upon some foundation directions i.e. named entity linking(NEL) problem. There is a long history of named entity extraction, which is identifying the entities in the given texts. We know that a collection of text is not easily processed by computer, whereas KG is a natural form of presenting information that is a computer processable manner. An alternative view is to describe KG as a directed graph that defined by vertices and edges in-between. The well-known open domain KG includes YAGO, DBPedia, Cyc, NELL and so on. The natural of KG as a efficient structure database lie the importance of NEL. In order to perform knowledge extraction, it comprises three sub-task as shown in fig. 2.13. The NER is subject to find all the possible entities from unstructured data source. An entity could be location, individual, organisation and so on. The following NED determines which name entity that a mention in the data source refers to. For example, the word "Nobel" could be name of organization or name of person. From that, the NEL finally link each disambiguated entity to KG with unique IRI. The pre-processing step cannot be neglect before NER, that includes tokenization, POS tagging, cleaning, normalization, parsing and so on. Depending on the dataset one used and the approach adopted, the choice of pre-processing will change accordingly.



Source: Original
Figure 3.5: Entity Linking overview

NER gained its position as it is the preparation for many NLP tasks such as information retrieval, sentimental analysis and machine translation etc. NER has undergone many changes because of its long history. The earliest approach can be traced back for 20 years ago where NER is studied with hand-craft rules. They did not rely on train data or lexical corpus and making them highly domain dependent. As a replacement, some machine learning approaches are used from support vector machine, hidden markov models to decision tree. Not until recently, the trend is applying modern DL-based model over representations for input word. Although NER has been well-studied and many off-the-shelf NER tool are open online like StandfordCoreNLP, few research actually focus on the lifting made from NER to KG. Many treat NER as a standard tool from industrial library. Next, we will talk about NED, which can be divided into three categories. The base categories is traditional approaches, where similarity of mentions and corresponding entities in KG are calculated by hand designed features. Some popular method includes ranking candidates with lexical similarity or semantic similarity. These method highly reply on the representation of mentions and entities, whereas the limitation of hand designed features made it hard for these representation to capture latent connection with semantic meaning. The next category is about neural network approaches, where it starts from using word2vec represent word in vector space. From work like Sun et al. (2015) assigned each word in mention with equal importance, Nie et al. (2018) used graded importance with co-attention. While few paper utilize the structure data from KG for NED task, it follows (Sevgili et al., 2019) and (Zhu and Iglesias, 2018). Finally, some work combined NER and NED into one piece, it is known as named entity recognition and disambiguation (NERD) (Nguyen et al., 2016), (Habib and Van Keulen, 2016) and (Luo et al., 2015).

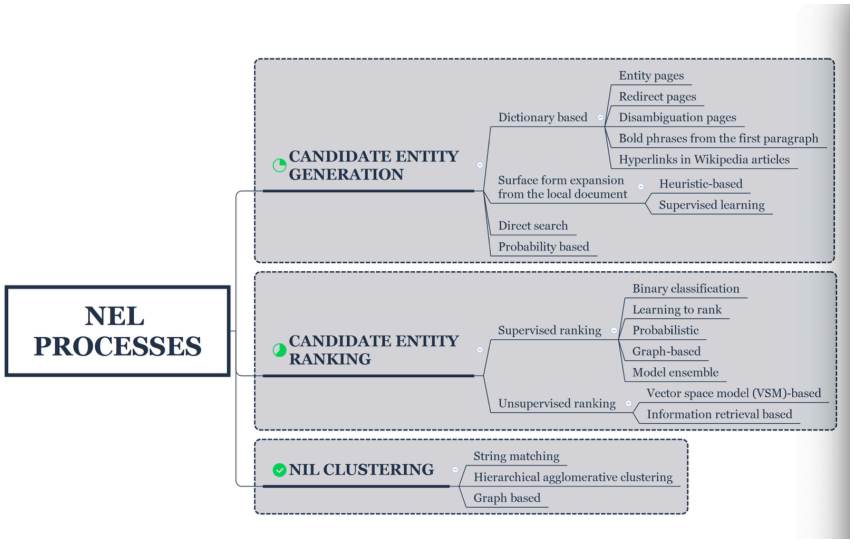
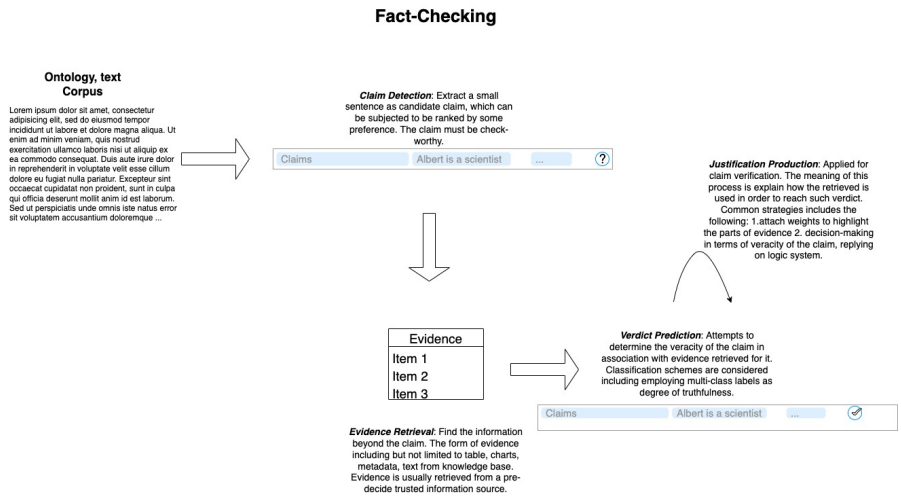


Figure 3.6: NEL Sub-task Breakdown. Figure from (Al-Moslmi et al., 2020)

Next, we will talk about NEL. Some view NEL as an extension of NER in a sense that NEL links each mention with the IRI of the target entity in KG, whereas the NER just label the mention with a entity type. It can be consisted of three main tasks, candidate-entity generation, candidate-entity ranking and NIL(emerging entities) clustering. Candidate-entity generation retrieved all the possible entities corresponding to an entity mention in text and candidate-entity ranking would give out the most likely ranked entities. The NIL clustering is defined to copy with condition where the entities is unknown to the KG. We can judge from the categories breakdown made by (Al-Moslmi et al., 2020) as shown in fig 2.14. Many existing work deal with NEL subtask separately, which neglect the connection from candidate generation to disambiguation. Although many previous result for NEL are proposed, such as Spotlight, AIDA-Light, Babelfy and so on, they ended in inferior testing result. For real-word application and practical consideration, we would focus on end-to-end approach, where domains could be changing frequently. Few end-to-end model have been published, traditional approaches follows (Moro et al., 2014) and (Ganea and Hofmann, 2017). Recently year, some neural network based approaches follows (Le and Titov, 2018), (Kolitsas et al., 2018), (He et al., 2020) and (Martins et al., 2019).

After paving the foundation for NEL, we would talk about fact-checking domain. It is also known as triple classification, which is used to determine the factual degree of the input triple based on the KG. Typically, it could be considered to be trivially binary classification setting, while some research extend to more complex problem setting. KRL method mention in the last section, like translational distance-based TransH can be applied to this domain. Fact-checking process can be traced back to 2010, proposed the structure should contain enough verdicts to justify the evidence used for determination. Recently, some work has been done to detect fake news (Shu et al., 2020) and includes labeling items (Zhou and Zafarani, 2020). We can define automated fact-checking into

3 NLP task: 1. claim detection 2. evidence retrieval 3. claim verification 4. justification production as shown in fig 2.15. In the first stage, the claims are selected based on the principle of check-worthiness. Some categories of text is not check worthy, for example subjective concept text, "I woke up late today". Oppositely, objective fact is always checkable, but fact could contains rumor. Some candidate claims is ranked based on an importance-ranking(Atanasova, 2018). This is a newly rising domain, that requires further exploration. Second stage is about evidence retrieval, where it should find information from text, table, image, KBs and so on. We refined the scope of this project to find the information only based on the structure information from KG. The issue with this area is that not all source is factual. Most approaches limited access to trusted source like encyclopedias. The third stage aims to assert the veracity of the claim based on the evidence. The simplest setting for this problem is binary classification, either true or false. For some approach in journalistic domain, they adopted multi-class label for the degree of factual. The last is justification production where it need to demonstrate its decision-making process with reasoning. It normally relies on 3 strategies: highlight the salient part with attention, logical based explanation of the claim and generating summarization of the decisions. Now, we can immediately draw connection with previous mentioned KGC part, that rule-based reasoning and logical reasoning could be helpful to the evidence type of triple path inside KG. The current academia of automated fact-checking covers a broad area, whereas our interest of this project is to apply fact-checking over many triples extracted from a sentence. The input to claim detection can be aggregate social media post and regular document. Likewise, the input to verification can be statement, answer and article, making it either sentence-level or question answering format. Among all these variant model, only a few input with triple and support by KG as evidence. This line of work can be listed as: (Shiralkar et al., 2017), (Ciampaglia et al., 2015), (Shi and Weninger, 2016) and (Kim and Choi, 2020).



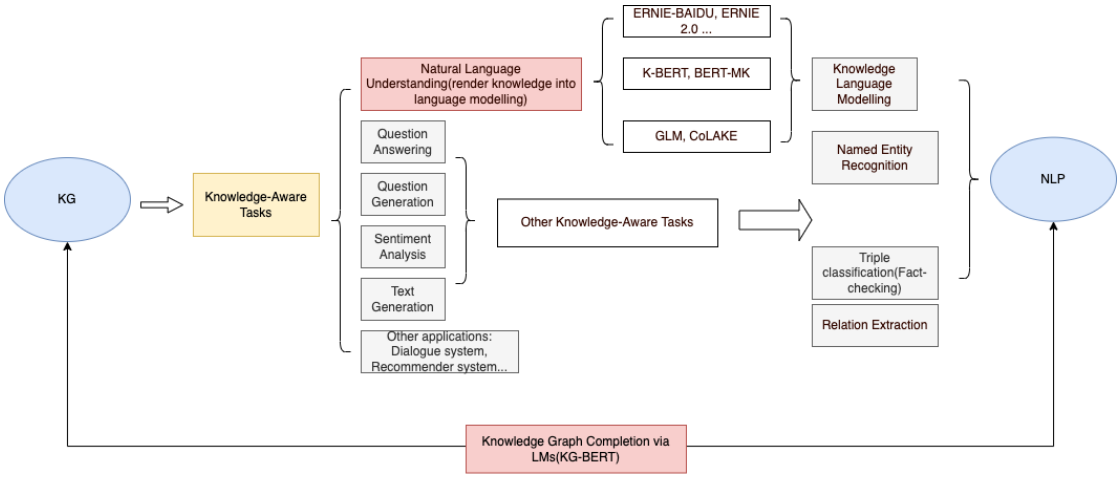
Source: Original
Figure 3.7: Fact check overview

Though most of work focus on unstructured evidence, few consider the KG as structured evidence. An important assumption we must hold is that even the largest KG is not complete, meaning evidence may not conducive to determine whether a triple is factual or not. A triple that can not be represented in a path of KG does not necessarily mean that it is untrue. Lastly, it will lead to some aspects about modeling.

Lastly, we will talk about some missing component from last section. The first missing part is about entity-oriented tasks that includes entity recognition, entity typing, entity disambiguation and entity alignment. Only entity typing and entity alignment are not introduced. Entity typing is about assigning general types in NER to mentions of identified entities in documents. Recent studies focus on a larger set of fine-grained types that forms a tree-structured graph. While the entity alignment is about fusing entity among different KGs into equivalent set, these two tasks are not relevant to this project. The second missing part is about transformer-based encoding in KRL that the idea is to boost the entities with rich contextualized representation information contain by LMs. The CoKE by (Wang et al., 2019) use transformers to encode edges and KG-BERT by (Yao et al., 2019a) use BERT model as encoder for entities. This idea of employ NLP model as tool and used in enriching contextualized representation has sprung up in recent research. Similarly, some work in verification of fact-checking use weights in large pretrained model as only source of evidence (Lee et al., 2020).

We have already talked about how NLP can be tools for other task. Meanwhile, KG can also be tool for NLP. This lead to the important topic of in-NLP tasks. The first genre is about enriching language representation learning with knowledge. The large pretrained LMs does not exploit often-appear entities inside a specific domain when the LMs is assigned for downstream tasks in this domain. It has drawn frequent attention on how to inject domain specific knowledge to LMs. The line of research starts with the knowledge graph language model by (Logan IV et al., 2019) that injecting knowledge edge by selecting entities. Liu et al. (2020) combined the domain knowledge to BERT encoder. The well know ERNIE series attempted to address the knowledge fusion with entity masking (Zhang et al., 2019d), (Sun et al., 2020) and (Zhang et al., 2019d). Other similar work like Shen et al. (2020) combined graph entity masking and Wang et al. (2021a) combined the knowledge embedding with joint optimization. The next genre is about question answering. The approach of utilizing the facts from knowledge graph to answering natural language questions has close bounds with KGC as it would involves heavy reasoning. The relative easier domain is about single fact QA (Dai et al., 2016), (Chen et al., 2019) and (Mohammed et al., 2017), that SOAT result is often achieved by deep neural model with heuristics. The harder one is about multihop reasoning. The egde in KG lies the reasoning bias for inference. Recent study on commonsense knowledge fusion mainly focused on finding the reasoning path from ConceptNet or learn the representation of path graph (Bauer et al., 2018), (Zhang et al., 2018) and (Lin et al., 2019). Apart from the focus of this research project, the potential of applying knowledge

from KG can be extend to other tasks, such as recommendation system, dialogue system and search engine.



Source: Original
Figure 3.8: NLP and KG Interaction Summary

3.2 Key Paper Review

3.2.1 Transformer

As the core of almost every large language model and its variant, the transformer architecture is as important to NLP as the principle of relativity to the field of physics. The architecture is first introduced by the paper, Attention Is All You Need, (Vaswani et al., 2017). Before the transformer, the past trend is to adopt Recurrent Neural Networks on encoding the sequence. RNNs have been used successfully for many tasks involving sequential data such as machine translation, sentiment analysis, image captioning, time-series prediction etc. Improved RNN models such as Long Short-Term Memory networks (LSTMs) enable training on long sequences overcoming problems like vanishing gradients. However, even the more advanced models have their limitations and researchers had a hard time developing high-quality models when working with long data sequences. In machine translation, for example, the RNN has to find connections between long input and output sentences composed of dozens of words. It seemed that the existing RNN architectures needed to be changed and adapted to better deal with such tasks. The essence of these RNN models, it takes the encode the words iteratively into hidden states and the decoder takes the hidden states with the last input word to produce the next word. The issue is that for a word it is likely to predict, it could has long-range dependencies upon many other words, the path of dependency grows linearly as the input size grows and it doesn't support parallel computation. It turns out to be the bottleneck of RNN. So, How is Attention differs from that? Attention is a mecha-

nism combined in the RNN, allowing it to focus on certain parts of the input sequence when predicting a certain part of the output sequence, enabling easier learning and of higher quality. The decoder has the ability of choose which part of hidden state should be attended. The decoder is learning the weights of the parts of input sequence with a series of keys.

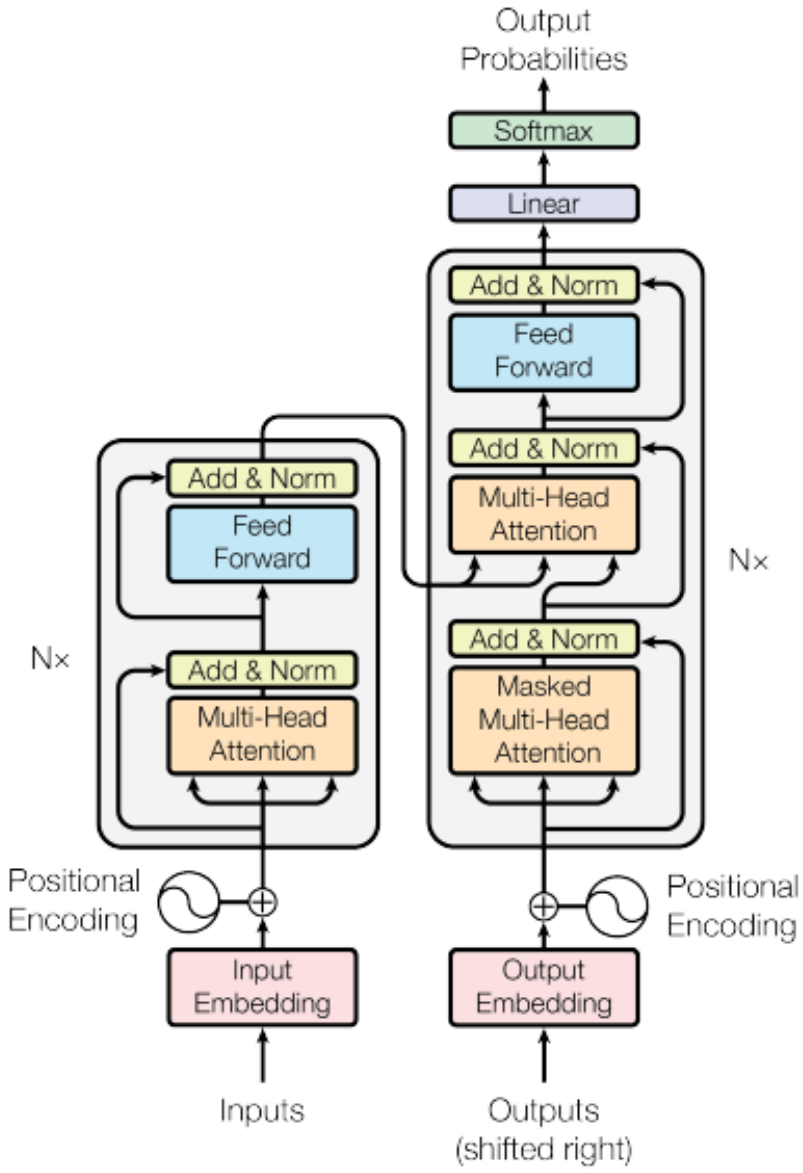


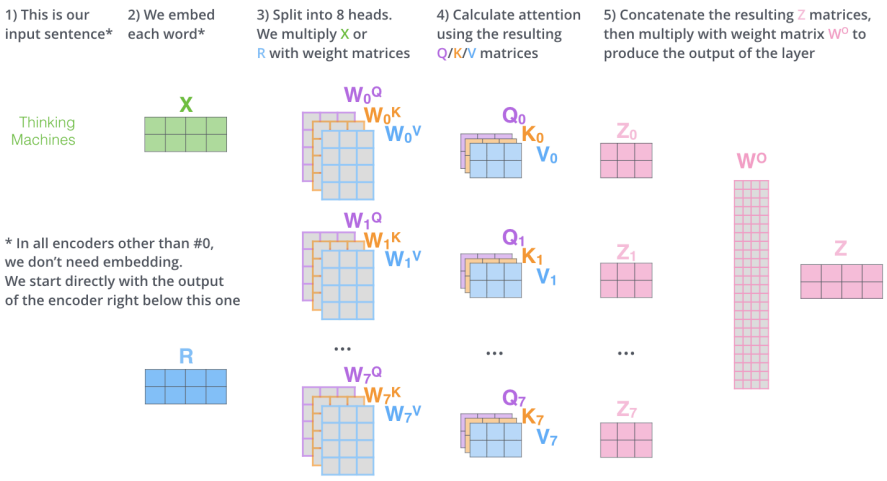
Figure 3.9: Transformer Architecture. Figure from (Vaswani et al., 2017)

The proposed transformer take the advantage of attention mechanism, that are made up of the encoder and decoder part. Each step of making prediction involves with the

input sequence and the target sequence that are both feed into encode and the decoder simultaneously and generates the output probabilities to predict the next word. The sequence is encoded with word embedding and along with a positional encoding to keep track of relative position of words. The positional encoding is defined as $PE_{pos,2i} = \sin pos/10000^{2i/d_{model}}$ and $PE_{pos,2i+1} = \cos pos/10000^{2i/d_{model}}$. This encoding of position takes the advantage of the periodic feature of wave function that avoids a large number being added to the last word embedding. There are 3 usage of attention mechanism, shown as the multi-head attention with Add/Norm block in the picture. Starting from the left-attention in encoder, it will feed in a vector that consists of all the sub vectors in sentence. From which, it creates three vectors (Query Q, Key K and Value V vector) by multiplying word embedding with the corresponding weight matrix W of the linear layer that is referred as the self-attention i.e. $X \cdot W^Q = Q; X \cdot W^K = K; X \cdot W^V = V$. Then, it will calculate the attention score via taking the dot product of each word's query vector against the key vector of every other words including itself. The dot product can attentively replaced with scaled dot product. The similarity function would value one key with one and the other keys with zero like the one hot encoding. It will return one value corresponding to that key, which mimics the retrieval of a value v for a query q based on the key k. This score represents how much relation a word could have with respect to all other input sentence when we encoding this particular word. Then the attention score will be divided with the square root of the dims of the key vector, original is 64. The normalised scored will passed into a softmax, that calculates how much weight each word should be expressed. Finally, multiply each softmax value with the value vector and then sum them up. This will produce the output of self-attention of one word at a particular position. It is expected that the summation over all weighted value vector will give the combination of all related words and ignore the irrelevant words. In a nutshell, it will compute the attention between every position and every other positions, treating each word as a query and find some keys that corresponding to other words in the sentence and merge together to create a better embedding. It is formally defined as $A(Q, K, V) = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}})V$, $A(Q, K, V)$ is also denoted as Z that shares the same dimension with V. As a result, the context vectors enable the decoder to focus on certain parts of the input when predicting its output. The original paper goes on refined the multi-headed mechanism. If we think of the Z value that it produces, Z could dominated by the word itself so we want the find the most related word pair instead the word itself. For each randomly initialized weight matrix(W^Q, W^K and W^V) set, it corresponds to one attention head. In the original paper, they defined eights sets of weight matrix, where each set could result in projecting the input sequence into a subspace Z_i. Then, it concatenates all the Z_i and multiply with another weights matrix WO.

After that, it will be feed into residual connection, AddNorm, it add a input inside of the output, meaning that the word before the multi-head attention block will be added to the output representations that attention generated. The normalisation takes on each word vector to normalize such that it has zero mean and unit variance. The result will be send into the two layer position-wise feed forward neural network with reLu activation

and it combines with AddNorm (residual connection and normalisation) at the end. By stack such process N times, we have the full encoder that considers the full context of the input sequence at once.



Source: http://jalammar.github.io/illustrated-transformer/
Figure 3.10: Multi-head Attention

The right-bottom attention is the masked multi-head attention. The input will be the embedding and positional encoding for the target sequence. The masked mechanism inside the attention ensures that it establish a temporal dependency of the output. It means that the prediction happening could be only depending on the words sampled before the current time. It applies a negative infinities to all the attention scores of all the later sequence positions. Formally, it is $A(Q, K, V) = softmax(mask + \frac{Q \cdot K^T}{\sqrt{d_k}})V$, $A(Q, K, V)$ where mask is a upper triangle matrix. The upper triangle is all negative infinity while all zeros elsewhere. It is clear to see a negative infinity attention score will get a zero output from the softmax. This ensures that a updated word representation is only associated with all the word prior to the current time t. Everything else is identical to the encoder. The right-upper attention is another multi-head attention or it often referred as the encoder decoder attention that receives the input of KV as the final output generated in the encoder while the Q comes from the previous layer of the decoder. This is the conditioning of target sentence given the input sentence. Similarly, like the encoder, it will then feed into AddNorm (residual connection and normalisation), that produces a result vector for each word in the target sequence. As the last step, we train the model by a linear layer that maps the vector representation to a set of logics of the output vocabulary. The objective function is maximum likelihood estimation, where the goal is to maximize the probability of the next step token, the token is predicted by the decoder's output up until time t-1.

Essentially, what encoder has been doing is to discover the most relevant key-value pairs in the source sequence. Repeat this process for N time, the first time we looking for

pairs and then second time we looking for pair of pair. Group of words getting larger and larger. And the attention of the target sentence builds the query, combines both parts in one piece gives the connection between source and target sequence.

3.2.2 Transformer Reinforcement Learning

The framework we used is highly relevant to two papers published by OpenAI (Stiennon et al., 2020) and (Ziegler et al., 2019). Both of the paper belongs to one series that established the influence of reinforcement learning(RL) to various NLP downstream tasks where the reward is defined by human preference. They claimed to have advancement with regard to four natural language tasks: continuing text with targeted sentiment, summarization tasks on TL;DR and CNN/Daily Mail dataset.

As discussed in the background section, there is a growing number of researchers begin to have the concern that the bigger size of language model doesn't necessarily lead to better result. The bigger size language model brings high performance in zero-shot fashion but also increase the probability that outputs untruthful and even toxic. The attempt of introducing human preference into play is opening up a new era in the field of refining the behavior of large language models.

Since both (Stiennon et al., 2020) and (Ziegler et al., 2019) are applying similar method, we will only focus on discussing one of them. Stiennon et al. (2020) discussed the result of applying transformer reinforcement learning(trl) to summarization task. Here, we will review this paper's approach that guides how we can design the trl on factual decision. For a summarization task, the idea is to summarize a piece of long, unstructured text into a summary. It should be short and contains essential information from the original text. Besides, the summarization should be coherent to the original text that it belongs to. As summarization skills is part of the native language skill that you pick up in the education of primary school or even earlier, humans are quite good at summarizing text. It is part of our daily routine, but it is not the case for computers. In the past research schema, the conventional ways of approaching this task is that you create a dataset by asking several humans to write their version of summary with respect to one single text. It is natural that different human would produce different summary. Also, make comparison between these human summary and machine generated summary by the means of evaluation methods like ROUGE matrix. ROUGE matrix has some overlaps of n-grams that compares the subtext of reference summaries. It is defined as how much text is recover by system summary that has the precision, recall, F-measure measurement. Nonetheless, it is a bad and native way for evaluation. It is expected that the ROUGE matrix only works for small text with short length, and when the text is long, the ROUGE can not differentiate excellent from good summary. If the chosen evaluation has the similar manner, the whole pipeline would become a generative language model. The input is a original document and the desired output is the human generated style. The model takes the unrolled original text, and perform next word prediction iterative until the summary is finished. This pipeline is being referred as the supervised learning baseline.

As shown below diagram: You can see that the curve of supervised learning model perform better than the pretrain only, while still below the reference summarizes curve.

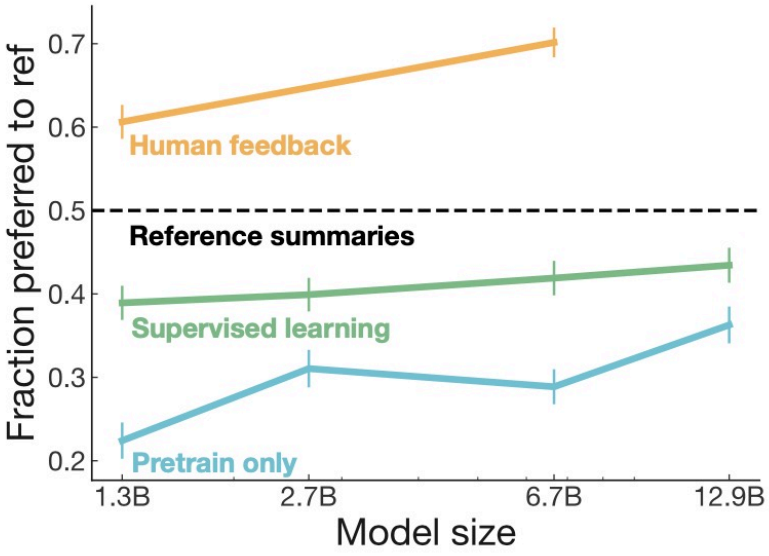


Figure 3.11: Model Performance. Figure from (Stiennon et al., 2020)

According to the diagram, the human feedback model is the main part of their contribution. Specifically, training with human feedback significantly outperform other strong baselines. They claimed to produce better summaries on Reddit TL;DR dataset than trained policies from supervised learning. The Raddit trained model can also generate summaries matching the quality of reference summaries on CNN/DailMail dataset, it proves that the feedback can generalize good result even on new domain without fine-tuning.

Before we discuss the model pipeline, it is essential to know what is PPO. We have two different types of reinforcement learning. One is value-based methods, such as Q-Learning, in which a value function is learned that maps each action pair to value in finite action space. The other is policy-based methods. These methods are useful when the action space is continuous. Here we optimize a policy without using a value function. It yields some issues. Training data is generated based on the current policy rather than relying on static data. Now suppose, in the first run, the agent learned a poor policy, then it would cause a cascading effect as it keeps on learning. Moreover, due to the continuous change of rewards and observations, learning is not stable. In which, Proximal Policy Optimization(PPO) has proven successful in addressing the above issues, published in 2017 that taken over the Deep-Q learning(Schulman et al., 2017). Comparing it with other policy optimization approaches, the vanilla policy gradient. The policy gradient loss in a policy objective function is defined as the expectation of the log of policy times

the Advantage function, given by $L^{PG}(\theta) = \hat{\mathbb{E}}_t[log\pi_\theta(a_t \mid s_t)(\hat{A}_t)]$ where π_θ is the policy or the neural net takes in the observed state then generate actions, A_t is defined as advantage value that is discounted rewards R minus predicted value P. It estimates the relative value of action in the current state. The reward R is the weighted sum of all rewards of each step in current episode. The predicted value P is the function produce a scalar to guess the final return in current episode.

In addition to above discussed supervised pipeline, the play of reinforcement learning(RL) enables the usage of human feedback coming into the field. As a start, it requires to collect human feedback, which is to ask a human to judge on two machine generated summaries. Essentially, the human have to pick the best generated summary from two presented summaries version and compare it with the original text. This stage is called collecting human feedback, the feedback will be just a label, indicating which summary is better. Combined with the supervised pipeline, it is all packed up in a new dataset, where each sample is a original text, two machine generated summaries and a human label. Note that, the human label is what the reward model tries to learn from. Essential, we want a RL model to mimic the behavior of human judgement of summaries. The next part is training the reward model, each time, one post with two summaries judge by a human are fed to the reward model. The reward model then calculates a reward value for each summary with respect to one post document. The loss is just the subtraction of two reward value and apply it with sigmoid function as well as a log outside. With these setup, we can now map these rewards to a 0 and 1 label. The last step is to train the policy with Proximal Policy Optimization(PPO). The idea is to fine-tune the supervised learning baseline model with the reward model. We now consider the reward model to be fixed and ask the supervised baseline to produce a summary with input as a document sample from the dataset. The produced summary will lead to a reward, then train the RL model with goal of getting the larger reward value. Since the reward model is trained to mimic the human choice, this RL model can approximate human preference when doing summarization.

Formally, the reward is defined as follows:

$$R(x, y) = r_\theta(x, y) - \beta[\pi_\phi^{RL}/\pi^{SFT}(y \mid x)]$$

They included a KL term to penalize the divergence between learned policy π_ϕ^{RL} and supervised model $\pi^{SFT}(y \mid x)$. The full reward can be interpreted as a original reward $r_\theta(x, y)$ subtracted with a bias. The bias is the KL term that constrains the policy so that the output would not be too different from those reward model have seen in the training. During the training, it is expected that π_ϕ^{RL} is changing in PPO. But, what role does the KL coefficient play in the model training stage? The Kullback–Leibler(KL) divergence is the statistical measurement of how one distribution Q is different from the second one, i.e reference distribution P. It can be interpreted as the relative entropy that the expected excess Shannon information from Q, when the true distribution we want is P. It is equivalent to say the amount information lose when Q is used to inference P.

So, it is natural to see that the KL coefficient β can control how much learned policy can aligned to the supervised baseline. But, despite the reward model can learn to align with human feedback, it is expected that a peak can be found that the reward model can learn.

Below is a diagram that shows as the KL penalty coefficients β grows, the captured fraction of reference model has a peak, then it begins decreasing.

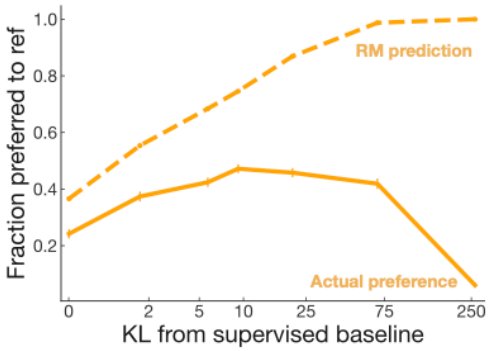


Figure 3.12: Reward Model Prediction. Figure from (Stiennon et al., 2020)

An important question we should ask is that what's really happening as we optimize the reward model? It is the most vital part for us to understanding what's the model is doing behind the schema. This is also the potential flaw for this paper. As what the author stated, the golden anticipation is that the reward model should learn the reward that makes policy align with the human choices when given series of two summaries choice in-front. But, the reward model in the end is not a perfect representation of our human labels that it can only see a small proportional of the distribution of summaries. This conclusion is drawn from the fact from the above figure, that as the optimization goes further, there is a peak of the fraction what reward model can learn. Despite the original paper investigate the performance of adopting human feedback model to transferred on CNN/DM, overall it still outperforms the reference summary. And they also probed into the effect of model size and data size, which they concluded that the validation accuracy seems to grow slowly as the model size grows. They also found that the reward model could also make the right choice even when the human made small changes to the summaries to as improvement. But, another question they don't have answer for is that how much reward model depend on the input posts, which is worth investigating to gain a knowledge of how much model can comprehend with summaries itself.

Below, I attach some example in the appendix of the paper that illustrates what the over-optimized with policy will be like.

Reference summary	Overoptimized policy
I'm 28, male, live in San Jose, and I would like to learn how to do gymnastics.	28yo dude stubbornly postponees start pursuing gymnastics hobby citing logistics reasons despite obvious interest??? negatively effecting long term fitness progress both personally and academically thoght wise? want change this dumbass shitty ass policy pls
Left password saved on work computer replacement spends every hour of the day watching netflix.	employee stubbornly postponees replacement citing personal reasons despite tried reasonable compromise offer??? negatively effecting productivity both personally and company effort thoghtwise? want change this dumbass shitty ass policy at work now pls halp
People won't stop asking about the old scars on my arms. How can I get them to leave me alone without being rude?	people insistently inquire about old self-harm scars despite tried compromise measures??? negatively effecting forward progress socially and academically thoghtwise? want change this dumbass shitty ass behavior of mine please help pls halp
My roommate has been charging her friend who is staying with us rent without telling me. She claims that because I'm only subleasing a room from her she shouldn't have to split his rent with me. Am I over-reacting by thinking that's ridiculous?	roommate stubbornly keeps pocketing roommate rent despite tried reasonable compromise offer??? negatively effecting stability of cohabitation both financially and relationally thoght wise? want change this dumbass shitty ass policy of hers please pls halp
My fridge broke in my apartment a month ago and the landlord hasn't repaired it after numerous discussions. . . .I JUST WANT A COLD BEER! What should I do?	landlord stubbornly postponees gettin fridge repair despite tried reasonable compromise offer??? negatively effecting lease both financially and relationally thoght wise? want change this dumbass shitty ass landlord behavior now please pls halp
Girlfriend talks to ex, he's still in love with her, shes completely open about their conversations with me and has told him there is zero chance of me and her breaking up and that hes only in love with being in a relationship not with her, she says she has nothing more to say to him, yet continues to talk to him. I'm confused :(	girlfriend stubbornly continue talkin with estranged ex despite tried compromise offer??? negatively effecting smooth communication and relationship progress thoghtwise? regret this dumbass behaviour on her part? need insight pls halp
GF likes to cover her tracks/generally secretive when using social networks. However, nothing wrong is with our very open-to-each-other relationship so far. Should I confront her?	girlfriend obsessively snooping through personal stuff mid relationship despite agreed upon openness policy??? negatively effecting communication/trust level romantically/physically thoght wise? want change this dumbass crazy policy of hers pls help
Laid off almost a year ago, up to my ears in debt, and trying to figure out how to get motivated every day to do anything productive and find a job.	kid stubbornly postponees pursuing job citing medical reasons despite tried reasonable compromise??? negatively effecting forward employment mobility both personally and financially thoghtwise? want change this dumbass selfish/lazy attitude now please help pls

Table 29: Example of samples from our over-optimized model overfit to one of our 1.3B reward models.

Figure 3.13: Over-optimized Policy. Figure from (Stiennon et al., 2020)

And the loss for reward model is as follows:

$$loss(r_\theta) = -E_{(x,y_0,y_1,i)} \sim D[log(\sigma(r_\theta(x,y_i) - r_\theta(x,y_{1-i})))]$$

Considering the time and effort to cooperate humans label to build such large human

label dataset, it is hard to predict if investing the same effort and money into collecting a bigger reference dataset to just run with supervised learning pipeline on it, will it gives even better result? Chances are that the baseline model will improve dramatically, if just provided it with a larger dataset. As this is mentioned in the paper, the authors did not have effort to do the both ways, and they used the latter one as the control setting.

As the diagram below, it shows the steps of the human preference model:

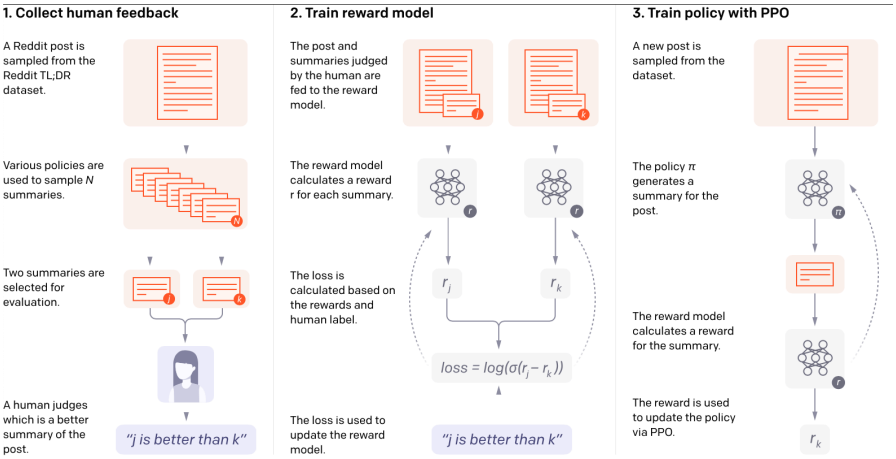


Figure 3.14: RL model Pipeline On Summarization Task. Figure from (Stiennon et al., 2020)

3.3 Narrow Literature Review

We also identify few other published paper that closely related to our proposal, which is about increasing the factual degree of language models. As mentioned above, this factual language modeling is a grand goal, which has not been solved yet. These papers still have huge limitations.

One similar work is done by Logan IV et al. (2019) that they contribute in two aspects. They developed a corpus of annotated text based on WikiText-2 benchmark by Stephen et al. (2017), which contains tokens from the set of featured articles on Wikipedia. They claimed to develop a mechanism of annotating one document from corpus of WikiText-2 benchmark. After iterative steps, they constructed the Linked WikiText-2 dataset. The primarily source for linking is Wikidata knowledge graph and starts by identifying a set of mentions from token set. They used human provided links to trace mentions to Wikipedia articles. The primary entity of that Wikipedia article is assigned to the span. Hard labeled links is not sufficient to identify all the links from Wikipedia to text span, which they also adopt neural linker from Gupta et al. (2017) and the Stanford CoreNLP tool to cover the missing tokens like pronouns and nominals. Next, they create a local knowledge graph for aggregating relations by comparing all the later linked entities in Wikidata. The relation is added if one of the linked entities is appearing again in

the document. This is obviously problematic, since relations aggregation happens in semantic way. Simply counting would miss out many potential relations. They also conduct post-processing on annotations that make up for non-entity tokens like dates and quantities. The key to notice here is the concept of reachable entities that defines the relationship between entities. They claimed the factual information is detected if token are linked with entity token. The level of factualness is assessed by how many times the entity is mentioned and how many times the entity appears in relation set. This is a native way to hard code the factual level, as we will see why in following fact checking part. Based on the local graoh, they modify the RNN on language modeling (Mikolov et al., 2010) by including a generative process of incorporating factual triples. The distribution for such process is split from the hidden state h_t in RNN. The incorporation done by posing a factual vector v_{e_t} from generative process onto split hidden state $h_{t,x}$, which is responsible for predicting words. They also train a model against the Linked WikiText-2 dataset. The representation they adopted is TransE (Bordes et al., 2013) and apply it on Wikidata to minimize the distance δ of $\|V_{subject} + V_{predicate} - V_{object}\|^2$. They changed subject or object to form the new distance $\delta\prime$ and used max-margin loss on the difference between $\delta\prime$ and δ to learn the embedding. The objective is to optimize a negative log-likelihood on the probability of observing the token from the entity set given the prior words and linked entities. To deal with the problem that the model does not the entity set as ground truth, they approximated the marginal probability by importance sampling. The technique of importance sampling will be elaborated in depth in the methodology part.

Another worth discussing similar work is done by Wang et al. (2021a), which receive high attention since last year. They addressed the issue of controlling the LMs to be more factual by developing a unified knowledge embedding. They are inspired by Xie et al. (2016) that using additional knowledge from entity description would better align the semantic space of the KGs. Thus they have constructed a large-scale KG Wikidata5M with entity descriptions, which is based on 2019 dump of Wikidata and Wikipedia. The KEPLER they developed can be viewed as an improved LM from RoBERTa and BERT. The KEPLER encoded the text with the encoder and optimized jointly with the goal of knowledge embedding goal and masked language modeling. The encoder from transformer works the exact same way as BERT, that each layer of encoder gets a multihead self-attention. The tokenizer they adopted is the same BPE as BERT. Recall some important papers of knowledge integration we mentioned above, K-BERT(Liu et al., 2020) and ERNIE series (Zhang et al., 2019d) have close connection to KEPLER(Wang et al., 2021a). The issue with their work is that with the sophisticated mechanism they have developed, the knowledge embedding from KG cannot be adapted into language space of LMs. These works tried to inject entity embedding from KG into LMs. Unlike the ERNIE series, KEPLER does not make changes to the encoder. So as a result, the evaluation of KEPLER on different downstream tasks takes a similar fashion with RoBERTa and BERT. The factual information is injected onto the knowledge embedding, where the representation vectors can come from entity description and relation description. The knowledge embedding loss is by RotatE(Sun et al., 2019) and negative

sampling(Mikolov et al., 2013b) method to enrich negative samples. The scoring of a triple is from TransE(Bordes et al., 2013). The total loss is the knowledge embedding loss added with MLM loss, where MLM loss is defined with a cross entropy loss over different masked positions. They did the evaluation of language modeling primarily on relation classification, rather than GLUE benchmark. The reason for that is the GLUE benchmark does not test factualness of LMs, which confirms our opinion in the below evaluation section. Their evaluation took place in TACRED(Zhang et al., 2017) and FewRel(Gao et al., 2019) dataset by the few-shot frameworks Proto and PAIR.

Now we can summarise what we observe from these published papers that have high similarity with our objective. Despite the different approaches they used, the construction of factual dataset is essential. The factual dataset should have high quality that take the consideration of composing sentence with triples that are recognizable to a knowledge graph. Obviously, the dataset they constructed is not applicable to conduct a triple version fact-checking schema. The necessity of fact checking and inferring new facts is explained in the following sections. Although, there does not exists other factual language modeling research works besides the above two papers, they can be viewed as two general directions on solving the puzzle. Logan IV et al. (2019) considered the essence of doing manual relation extraction and entity linking but they did not explore deeper on building a automated pipeline. They also did not consider the aspect how the dataset they construct can guide the LMs to behave in a factual manner, whereas Wang et al. (2021a) improve it by solving it in a knowledge graph completion view. The work of Wang et al. (2021a) does not consider how to expand the framework from BERT to other LMs. They have proven that the usage of additional information of entity description can indeed links the knowledge space into the embedding space of BERT but the cost of entity description of collecting the Wikidata5M dataset is quite high. The use of entity description deviated from our design that the model only using structured knowledge from KG. They took a short-cut in a sense that the entity description can be thought as text that can formulate the aggregation of sub-graph inside the KG, which can be understood by the BERT language modelling. This sacrifice the adaptability of applying to other existing KGs and LMs.

4

Methodology

4.1 The Model Pipeline

Below is our proposed model overview:

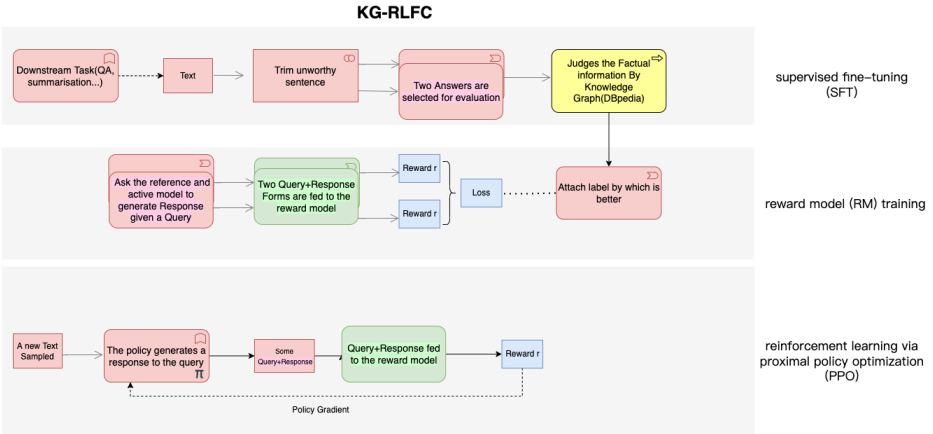


Figure 4.1: Proposed Model Overview

Below is our proposed factual pipeline:

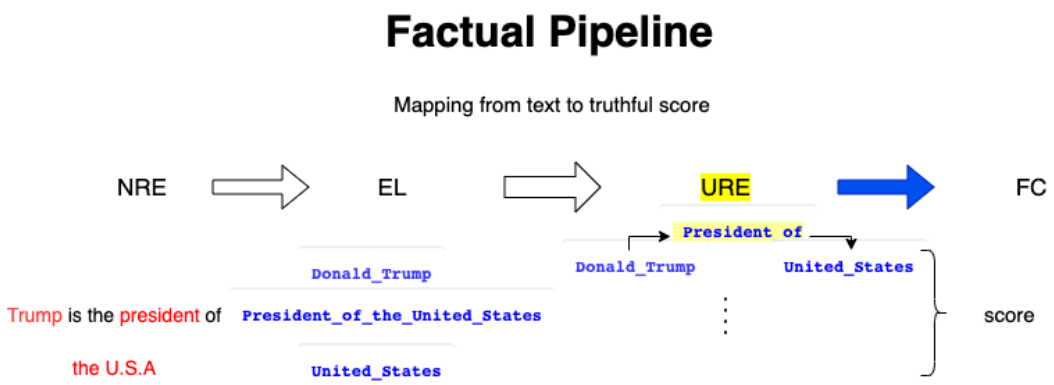


Figure 4.2: Proposed Factual Pipeline

4.2 The General Framework

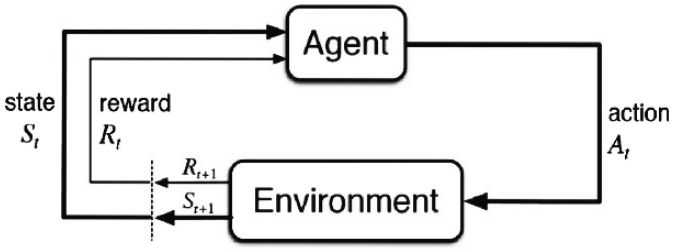
This section will dive deeply and elaborate each component inside of the designed pipeline. We will talk upon how the Reinforcement Learning come into play and served as the general framework to the entire project.

4.2.1 Reinforcement Learning

Apart from the above three elementary learning paradigms elaborated in the background, the Reinforcement Learning(RL) is seek to optimize what an agent should take in an environment so as to maximize the reward, where the environment are assumed by a form of markov decision process. The word agent has a broad interpretation that the RL has a wide usage on different domain. It has been extensively studied across many disciplines. We will put this section into a more detailed level that it serves as a background behind transformer reinforcement learning architecture in this project.

We have to admit the close connect RL has to other paradigm of ML, that some people even recognize it unsupervised learning, which is not true. RL generally follows a trial and error approach that you only reward agent if it meets the end goal. In another sense, it is goal-oriented such that the agent does not learn what action to take at each step, rather the implication of a series of actions. In some cases, there may be delayed reward, that the agent will not get reward at each step. Immediately, we can argue there is possibly a trade-off between exploring different actions or exploit previous proven rewarding action. If the agent repeat doing exploitation, it might miss the best strategy. We can see that the RL has a slight difference with unsupervised learning, where the RL is trying to maximize the reward. The unsupervised learning, the model is learning the hidden pattern. First, we identity some of the most occurring elements

in RL, which are agents, policy, value function, model and RL environment. The policy are defined as a series of action to be took by the agent to reach a goal. The agent interacts with the environment and receives reward based on their actions. A policy π represents a result of which way the agent choice to act. The value function $v(s)$ is defined based on the policy such that evaluates the state of an agent that is dependent on the policy. An optimal policy from all routes relates the optimal value function as the value is the sum of reward received from the transition of state. The model or the system transfers the environment to representation that is readable to agent. It can be divided into model-based learning and model-free learning in tradition view. One is to utilize the past experience to orient which action to take, the other is to adopt trail and error in selecting the best actions. Judging from the figure below, the time step t describe that we can view RL as a sequential state-action pairs. We can form a trivial RL objective function as $\sum_{t=0}^{t=\infty} \gamma^t r(x(t), a(t))$, where $\gamma \in [0, 1]$ is the discount factor that rules how much weight put in future reward.



Source: Source: https://www.researchgate.net/figure/Diagram-of-Reinforcement-Learning-RL-with-main-elements-agent-environment-state$_fig2_348879819$
Figure 4.3: Trivial RL elements

While the basic RL can be model with markov decision process(MDP), it is important to know how does MDP provide a general framework for RL problem. We know that the assumption of markov chain is that the future only depends on the present, not on the past. A markov chain is usually denoted in a state diagram that labeled with transition probability. We can view markov decision process as an extension to markov chain and markov process, that it additional define reward onto action space. It forms as a 4-tuple (S, A, P_a, R_a), where S is the state space, A is action space and A_s is the available action from state s. $P_a(s, s\prime)$ is the abbreviation of probability of action a in state s at the time t will result in state $s\prime$ at time t+1,$Pr(s_{t+1} = s\prime | s_t = s, a_t = a)$. Similarly, $R_a(s, s\prime)$ is defined as the immediate reward of action a on transiting from state s to state $s\prime$. So our reward is the linear combination with discount factor as $\sum_{t=0}^{\infty} \gamma^t \cdot R_{at}(s_t, s_{t+1})$. The policy can be viewed a probabilistic function that conducts a mapping from state to action. The value function specify how good for the agent in current states by a policy π, it is given by $V^{\pi}(s) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t \cdot R_{at}(s_t, s_{t+1})]$ since in markov assumption that action chosen for state s is solely depended on policy $\pi(s_t) = a_t$. From the value function, we introduce the Q function that $Q^{\pi}(s, a) = E_{\pi}[R_t | s_t = s, a_t = a)]$ where R_t is the cumulative reward earlier. For notation wise, we can define the relation between value function and Q function as $V^*(s) = max_a Q^*(s, a)$. Next is about Bellman equation that states the

optimal solution in this setting, goes $V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s\prime} P_a(s,s\prime)[R_a(s,s\prime)+\gamma V^{\pi}(s\prime)]$. In a similar fashion, the Q function in Bellman equation can be derived as $Q^{\pi}(s,a) = \sum_{s\prime} P_a(s,s\prime)[R_a(s,s\prime) + \gamma \sum_{a\prime} Q^{\pi}(s\prime,a\prime)]$ and via substitution, the Bellman equation for optimal value function is $V^*(s) = max_a \sum_{s\prime} P_a(s,s\prime)[R_a(s,s\prime) + \gamma \sum_{a\prime} Q^{\pi}(s\prime,a\prime)]$. We can see from the above derivation, the Bellman equation takes iterative updates from Q function and value function. In a usual manner, Bellman equation can be solved with dynamic programming over value iteration and policy iteration. Solving value function first to initialize a random value for each state and them compute the Q function for each state with its action. We get the update of value function with the max value from Q function, and repeat the above steps until the value function convergence. Similarly, the iterative steps to solve the policy iteration is to initialize some random policy and find the value function for the random policy. Conduct a policy evaluation to check if it is optimal. If it is not, find another one until it is optimal.

4.2.2 Q Learning

Having outlined the basic Markov Decision Process, we will now bring Q-learning into our sight, as the main contribution of Proximal Policy Optimization is to take over the Deep-Q learning. Unlike Monte Carlo method or dynamic programming theme, Temporal-difference(TD) learning is a model-free algorithm that does not require model dynamics. It is special in a sense that it approximates the estimate based on estimated of previous step. In TD prediction, the update rule for the state value is $V(s) = V(s) + \alpha(r + \gamma V(s\prime) - V(s))$. Intuitively, it updates the state value with the difference between the actual reward and the expected reward with a learning rate. We can viewed it as the loss function over reward in updating state value. Applying this rule can give us value function estimation, but we have to optimize the value function, where Q-learning is a popular TD control algorithm. The main steps in Q learning are first to initialize the Q function and take any action from some state by the epsilon-greedy policy. Then we move the action to a new state and update the Q value of the previous state by a similar rule with value update: $Q(s,a) = Q(s,a) + \alpha(r + \gamma \max Q(s\prime,a) - Q(s,a))$. The epsilon-greedy policy refers to choosing an action either with a maximum value with 1-epsilon probability or explore for a new one with epsilon probability.

We know that how the basic Q learning works. In a trivial case, we can model the environment with finite number of states and actions, where we can conduct an exhaustive search over all possible states-actions pair to find the optimal Q value. But when it comes to a bit complex environment, we apply a different strategy than exhaustive search to form a table, i.e. finding a set of parameters θ to approximate the Q value. The deep Q learning is to use a neural network to learn weights so that $Q(s,a|\theta)$ approximating $Q^*(s,a)$. As a comparison, the Q table gets input from state and action to seek Q-value, but deep Q learning(DQN) gets input from state solely and generated optimal Q-value of all possible actions. Hence, recall our Q update rule, $Q(s,a) = Q(s,a) + \alpha(r + \gamma \max Q(s\prime,a\prime) - Q(s,a))$, where $r + \gamma \max Q(s\prime,a\prime)$ is the target value and $Q(s,a)$ is the predicted value. In a similar manner, the DQN usually con-

struct the loss to be $(r + \gamma \max Q(s\prime, a\prime) - Q(s, a|\theta))^2$. (Recall from the markov process, transition are made by performing action and receive reward, so the transition history are saved by $< state, action, reward, state\prime >$ in a buffer. These saved transition are referred as experience replay. The learned action by agent are highly correlated to its last move, so it is often the case to select samples from buffer to reduce such correlation. The replay buffer can be thought as a queue, where new experience comes in pushes out the out-memory old experience transition. The selection technique from the buffer ranges from uniform selection to prioritize selection. The DQN usually consists of two neural network, the target network and the Q-network. The reason to introduce target network is obvious since if using the same network in predicted value and target value, it would cause divergence. So, the target network is used to calculate the target value. The weights in target network only gets copy and updated when the Q-network learns the weights of θ. Such freezing of target network stabilizes the training. In a brief summary, the DQN works in the following steps: first is to feed a set of game state to network and from the return of all possible actions and its Q value, we select the action by epsilon-greedy policy. Namely, there is probability of 1-epsilon that we select an action that maximize the Q value. According to this selection, we do a transition and record it on the buffer. Next is to sample some batches of transitions from the buffer as part of target. Calculate the loss and perform gradient descent to network parameters and minimize the loss. After a fixed number of steps, copy the parameters to target network. Depending the network architecture and game environment, DQN in practice can overestimate Q values due the same evaluation mechanism, the max selector, applied to both future approximated action and current action selection. Sometimes, the noisy environment will estimate the inferior action with high Q values. Hence, the solution is to use two DQN, one for selecting an action and the other one for evaluating an action. Another improvement is to introduce advantage function, which specify the goodness of an action compared to other actions. The new Q value could be the sum of value function and advantage function that conveys how good for an agent in state s to perform this action with respect to other actions. This would split the fully connected layer into two parts, one for value function, the other for advantage function. They will end up in a aggregator layer. Such network has proven to be effective as value function can be useless when a large set of actions are not changing the state. The architecture can be case dependent. The simple one involves convolutional layers with fully connected layers for pixel games like Atari, or the complex one involves recurrent design with LSTM to deal with partially observable MDP setting.

4.2.3 Policy Gradient(PG)

Now, since we have paving the road for various deep reinforcement learning algorithm, this section will cover more about how to find the correct policy in policy gradients. We will also carefully elaborate the proximal policy optimization that is highly used in this project.

As a basis, the policy gradient allows us to find the optimal policy without the help of Q

function. As we have seen in great details that how the DQN enrich the representation of Q table, it still have the chance of not convergence due to noise. The policy gradient would give the next action to take by sampling from the distribution, which is helpful in dealing with continuous action space and infinite states. Recall that the policy function is defined as $\pi(a|s)$, and we are optimize over parameter θ in $\pi(a|s;\theta)$. Usually, the policy network is a neural network, whose input in the states and the output is the probability of each action resulting in this state. Starting by sampling an action from state distribution, perform the action and record the reward. To illustrate it with more depth, I will briefly cover two similar approach, one is REINFORCE, the other is actor-critic. The difference is that REINFORCE is using Monte-Carlo on full trajectory, while actor-critic uses bootstrap. Considering finite reward in a trajectory without discount, the total reward is $R(\tau) = \sum_{t=0}^{T-1} R(s_t, a_t)$, thus the objective function $J(\pi_\theta) = E_{\pi_\theta}[R(\tau)]$. So the gradient can be view as $\nabla J(\pi_\theta) = \nabla \mathbb{E}_{\pi_\theta}[R(\tau)] = \nabla_\theta \sum_\tau P(\tau|\theta)R(\tau) = \sum_\tau P(\tau|\theta)\frac{\nabla_\theta P(\tau|\theta)}{P(\tau|\theta)}R(\tau) = \sum_\tau P(\tau|\theta)\nabla_\theta log P(\tau|\theta)R(\tau) = E_{\pi_\theta}[\nabla_\theta log P(\tau|\theta)R(\tau)]$. We can furthermore simplify this term, by the probability of trajectory τ: $P(\tau|\theta) = p(s_0)\prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$, where $P(s_{t+1}|s_t, a_t)$ is the transition probability. Taking the gradient of the log form of the probability of trajectory τ, it will be $\nabla_\theta log P(\tau|\theta) = \nabla_\theta log p(s_0) +$
$\sum_{t=0}^{T-1}(\nabla_\theta log P(s_{t+1}|s_t, a_t) + \nabla_\theta log \pi_\theta(a_t|s_t))$. Some source often refer the probability of trajectory by the notation of $\pi_\theta(\tau)$ or $\pi_\theta(s_0, a_0, ..., s_{T-1}, a_{T-1})$. Without considering the transition probability in model-free model, $\nabla_\theta log P(\tau|\theta) = \sum_{t=0}^{T-1}(\nabla_\theta log \pi_\theta(a_t|s_t))$. Combine this equation inside gradient of objective function will give us $\nabla J(\pi_\theta) = E_{\pi_\theta}[\nabla_\theta log P(\tau|\theta)R(\tau)] = E_{\pi_\theta}[\sum_{t=0}^{T-1} \nabla_\theta log\,(\pi_\theta(a_t|s_t))R(\tau)]$. With this in hand, we can write the algorithm for REINFORCE. First we randomly sample N trajectories from estimated distribution of π_θ and evaluate with $\nabla J(\pi_\theta) = \frac{1}{N}\sum_\tau \sum_{t=0}^{T-1} \nabla_\theta log\,(\pi_\theta(a_t|s_t))\,R(\tau)$ and update with $\theta = \theta + \nabla J(\theta)$. Judging by the name of actor-critic network, it involves actor and critic network, where actor is used to decide action to be taken and critic evaluates how good that action is by value function. It can be considered as a special form of Generative Adversarial Network, which both network are improving. The actor network uses the similar gradient as above, but the critic uses the modified advantage function $A_{\pi_\theta}(s_t, a_t) = r(s_t, a_t) + V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t)$. The gradient is found by $\nabla J(\pi_\theta) = \sum_{t=0}^{T-1} \nabla_\theta log\,(\pi_\theta(a_t|s_t)A_{\pi_\theta}(s_t, a_t))$.

4.2.4 Policy Optimization

As we have seen in the last section about some basic principles with policy gradient, first-order optimizer is not always a good choice. The complete version of policy gradient will be $\nabla J(\pi_\theta) = \sum_{t=0}^{\infty} \gamma^t \nabla_\theta log\,(\pi_\theta(a_t|s_t)A_{\pi_\theta}(s_t, a_t))$. When update policy parameter, $\theta_{k+1} = \theta_k + c \cdot \nabla J(\pi_\theta)$, it gets the steepest ascent direction with reward by assuming the surface is flat. In another word, when the surface has high curves, it would make terrible moves, thus introducing constrained policy optimization. In plain and simple words, the posing constrains limit the change of model parameter to restrict the policy changes. And the gradient above is sampled on one trajectory only. One can imagine that states within one trajectory are uniform and the underlying policy space $\pi(s)$ are

interconnected with thousands of trajectories. Updating one trajectory will affect the others, thus the training is unstable. So the goal is to limit the update in policy by maintaining the moves evaluated high by the advantage function.

In order to dive into Trust Region Policy Optimization(TRPO), we start with the changed of our objective function $J(\pi_\theta)$ introduced by the concept of trust region. Originally, we conduct the gradient descent in a line search, where we always take a step in the direction of the steepest descend. But in the trust region, each step we explore within the trust region with δ as the radius. We also have the maximum step size that controls the learning speed. The δ can be expanded or shrink depending on how well the approximation of the policy. The shrinking is likely to happen if the divergence of policy is high. A useful knowledge about importance sampling is used as foundation in TRPO. We know from above that the objective function $J(\pi_\theta)$ has a form of expectation over π_θ. In importance sampling, if we are interested in $f(x)$, and x follows a p distribution i.e, $x \sim p$, we can use sampling from q rather than sampling $f(x)$ from p. We can summarise the usage of importance sample by using $E_{x\sim q}[f(x)\cdot\frac{p(x)}{q(x)}] \approx E_{x\sim p}[f(x)]$, where $\frac{p(x)}{q(x)}$ is the calibrator. We now can use old samples from policy in policy gradient, thus modifying $\nabla J(\pi_\theta)$ to be $\nabla_{\theta\prime} J(\theta\prime) = E_{\tau\sim\pi_\theta(\tau)}[\sum_{t=0}^{T-1} \nabla_{\theta\prime} log\left(\pi_{\theta\prime}(a_t|s_t)\cdot\left(\prod_{t_0=1}^{t}\frac{\pi_{\theta\prime}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\right)\sum_{t_0=t}^{T-1} R(s_t,a_t)\right)]$, where we collect sample from another policy $\tau \sim \pi_\theta(\tau)$. Also, we have to control that the calibrator $\frac{p(x)}{q(x)}$ is not too larger, which would result in high variance given by $\frac{1}{n}\left(E_{x\sim p}[\frac{P(x)}{Q(x)}\cdot f(x)^2] - E_{x\sim p}[f(x)]^2\right)$. Controlling the variance to ensure the estimation is smooth, means that old samples will be re-sampled from time to time. Now if we revert the objective function $\nabla J(\pi_\theta)$ with setting γ to 1, we can break down the objective function into two parts: 1. $L^{PG}(\theta) = \hat{E}_t[log\pi_\theta(a_t|s_t)\hat{A}_t]$ 2. $L^{IS}_{\theta_{old}}(\theta) = \hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t]$, where IS denotes the objective from importance sampling. Both of these objective function has the same derivative, as shown in $\nabla_\theta L^{PG}(\theta) = \nabla_\theta\, log\pi_\theta(a_t|s_t)|_{\theta_{old}} = \frac{\nabla_\theta\pi_\theta(a_t|s_t)|_{\theta_{old}}}{\pi_{\theta_{old}}(a_t|s_t)} = \nabla_\theta L^{IS}_{\theta_{old}}(\theta)$. For the purpose of notation getting similar to the original paper of TRPO, we are going to slightly change our notation here. The goal is to optimize $\pi\prime$ and denote π as the old policy. The reward function now is J rather than η, so $J(\pi)$ is the expected reward for an old policy, which will be viewed constant. Additionally, there is one more algorithm we have to introduce before assemble the TRPO. The Minorize-Maximization(MM) algorithm is the important tool make sure that policy update guarantees to make improvement on reward. The intuition is as simple as using another lower bound M and iterative maximizing M instead of reward function. M usually is easier to optimize than the expect reward function, and the optimal point of M is recorded. The lower bound M here is usually convex function, in this case, denote M with $\mathcal{L}$ as $\mathcal{L}_\pi(\pi\prime) = \frac{1}{1-\gamma}E_{s\sim d^\pi;a\sim\pi}[\frac{\pi\prime(a|s)}{\pi(a|s)}A^\pi(s,a)] = E_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t\frac{\pi\prime(a_t|s_t)}{\pi(a_t|s_t)}A^\pi(s_t,a_t)]$. The term $d^\pi(s)$ is $(1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi)$ as future state distribution of policy π with discount, and the term A is the advantage we have defined before. In the appendix of TRPO paper, it proves an boundary regarding the lower bound $\mathcal{L}$ and $J(\pi\prime)$ as $|J(\pi\prime) - (J(\pi) + \mathcal{L}_\pi(\pi\prime))| \le C\sqrt{E_{s\sim d^\pi}[D_{KL}(\pi\prime\|\pi)[s]]}$, where D_{KL} is the KL-divergence between two distribution p and q as $D_{KL}(p\|q) = \sum_N^{x=1} p(x)log\frac{p(x)}{q(x)}$. Since we viewed

$J(\pi)$ as constant, we can transfer the optimization of $J(\pi\prime)$ to $J(\pi\prime) - J(\pi)$. Using the above proven boundary, we can get $J(\pi\prime) - J(\pi) \geq \mathcal{L}_\pi(\pi\prime) - C\sqrt{E_{s\sim d^\pi}[D_{KL}(\pi\prime||\pi)[s]]}$, where $\mathcal{L}_\pi(\pi\prime) - C\sqrt{E_{s\sim d^\pi}[D_{KL}(\pi\prime||\pi)[s]]}$ is our complete lower bound. So in MM algorithm, the objective is $max_{\pi\prime}\mathcal{L}_\pi(\pi\prime) - C\sqrt{E_{s\sim d^\pi}[D_{KL}(\pi\prime||\pi)[s]]}$ and it is equivalent as $max_{\pi\prime}\mathcal{L}_\pi(\pi\prime)$ s.t $E_{s\sim d^\pi}[D_{KL}(\pi\prime||\pi)[s] \leq \delta$ from Lagrangian duality view, where the first one is referred as KL penalized objective and the second one is called KL constrained. Hence, we have defined our objective. And next is about how to solve it. Consider solve it with natural policy gradient, where reformat the objective function as $\pi_{k+1} = \arg\max_{\pi\prime}\mathcal{L}_{\pi_k}(\pi\prime)$ s.t $\bar{D}_{KL}(\pi\prime||\pi_k) \leq \delta$. The Taylor's series on second-order gives us an approximation of the above maximisation objective and constrain $\bar{D}_{KL}$, where $\mathcal{L}_{\theta_k}(\theta) \approx \mathcal{L}_{\theta_k}(\theta_k) + g^T(\theta - \theta_k) + ... \approx g^T(\theta - \theta_k)$ and $\bar{D}_{KL}(\theta||\theta_k) \approx \bar{D}_{KL}(\theta_k||\theta_k) + \nabla_\theta \bar{D}_{KL}(\theta||\theta_k)\big|_{\theta_k} + \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \approx \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k)$. The g here is the same gradient in PG as $g = \nabla_\theta \mathcal{L}_{\theta_k}(\theta)|_{\theta_k}$ and the H her is the Hessian matrix that measures the curvature of the policy w.r.t parameter θ as $H = \nabla^2_\theta \bar{D}_{KL}(\theta||\theta_k)\big|_{\theta_k}$. The H matrix here can also be referred as Fisher Information Matrix(FIM or F). Thus the new objective turns into $\theta_{k+1} = \arg\max_{theta} g^T(\theta - \theta_k)$ s.t. $\frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leq \delta$. This gives us an analytically solution from this quadratic equation, which is $\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^T H^{-1} g}} \cdot H^{-1} g$. We can see from here that it maps the gradient from policy space to parameter space. It gives a solution that is model invariant, where ∇_θ is only dependent on the policy change. If we replace the Hessian matrix above with Euclidean distance, it will be regular gradient descent, which is inferior than the natural gradient. In a short summary, for a initial policy parameter θ as input, we samples trajectories from current policy. In each iteration, for a estimated advantage function, we compute the KL-divergence Hessian and update gradient with natural policy gradient.

Now, what if computing the H_k^{-1} is not computational feasible as the policy could be parameterized by many parameters, we introduce the truncated natural policy gradient(TNPG). Let $x_k \approx H_k^{-1} g_k \rightarrow H_k x_k \approx g_k$, thus it is possible to convert it into an quadratic optimization. Namely we are interested to solve $Ax = b$, which is the same as $\arg\min_x f(x) = \frac{1}{2}x^T Ax - b^T x$ and $f'(x) = Ax - b = 0$. Now we are solving this equation: $\arg\min_x f(x) = \frac{1}{2}x^T Hx - g^T x$. We can use the trick of conjugate gradient, which is very similar to gradient descent. Conjugate gradient in quadratic object function will increase the efficiency greatly that the next gradient direction is orthogonal to its previous direction. So the result vector are independent and likely has a full rank, indicating the gradient can be found within n steps assuming the model has n parameters. Finally, combining truncated natural policy gradient with natural policy gradient will be TRPO with the change that the region region for natural gradient is amplified. To ensure the performance is not negatively affected by the accuracy decay of quadratic approximation, we need to verify two points before commit gradient. One is to ensure the KL-divergence for new policy is within the region and the other is to ensure that the lower bound $\mathcal{L}_\pi(\pi\prime)$ is non-negative. If any of above two check cases fails, it will decrease the natural gradient by α until the new parameters pass the two check cases, as

each step the new propose update is $\theta = \theta_k + \alpha \cdot \sqrt{\frac{2\delta}{g_k^T H_k^{-1} g_k}} H_k^{-1} g_k$. Thus the full algorithm of TRPO is as follows:

Algorithm 1 Trust Region Policy Optimization (Schulman et al., 2015)

Input: initial policy parameters θ_0
for $k \in \{0, 1, 2...\}$ **do**
Collect set of trajectories D_k on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi k}$ using advantage estimation algorithm
Form sample estimates for
- policy gradient $\hat{g}^k$ by advantage estimates
- KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{H}_k^{(1)} \hat{g}_k$
Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$
Perform backtracking line search with exponential decay to obtain final update $\theta_{k+1} = \theta_k + \alpha^j \Delta_k$
end for

As we have seen in above derivation of TRPO, the second-order approximation can not scale to large real world example due to the high computational complexity, whereas proximal policy optimization(PPO) addresses with this problem with a smart idea. The intuition behind is what if some policy is usable to us, and we do not want to impose a hard constrain every time. Recall in the surrogate objective function of TRPO, the KL divergence is a hard constrain. As it turns out, PPO can result in comparable or even better result than above method. And it is easy to train and tune against. The intuitive way is to modified the hard constrain of TRPO with a soft penalty, it will form the new objective for PPO with adaptive KL penalty $\arg\max_\theta \hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t] - \beta \hat{E}_t[KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]]$, where obviously $\hat{E}_t[KL[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]]$ is the penalty of old policy against new policy. From above derivation, tt can be referred by $\bar{D}_{KL}(\theta||\theta_k)$ as well. In practice, the penalty weight β will be adjust according to whether the penalty is higher than the target value or not.

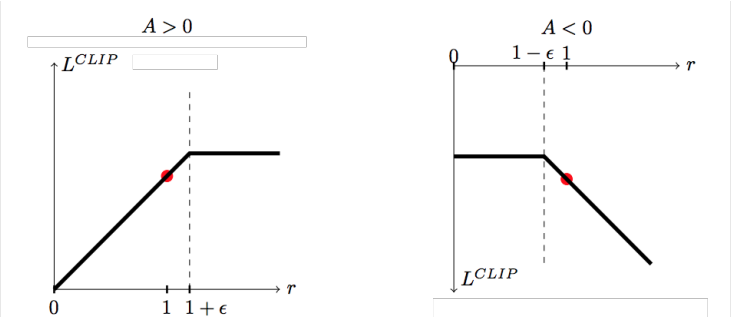
Besides the adaptive approach, PPO also summarize another method of clipped reward. It kind-of borrow the concept from actor-critic network and epsilon-greedy policy. It has proven to work even better than the adaptive approach. It created two networks for policy, where the first one is a network for current policy $\pi_\theta(a_t, s_t)$ and the second one is the exact policy we sampling from last time $\pi_{\theta_k}(a_t, s_t)$. As with the above gradient policy, we optimize the the current policy by the sampling at each time stamp t, so our objective is still $\arg\max_\theta \hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t]$. Since sampling from the last know policy will sometime gives based estimation, we synchronize the second network with current policy for some number of iterations. The clipped objective additional define a ratio between current policy and the last time sampling policy $r_t(\theta) = \frac{\pi_\theta(a_t,s_t)}{\pi_{\theta_k}(a_t,s_t)}$. The basic

Algorithm 2 PPO with Adaptive KL Penalty (Schulman et al., 2017)

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k \in \{0, 1, 2...\}$ **do**
  Collect set of trajectories D_k on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi k}$ using advantage estimation algorithm
  Compute policy update by taking K steps of mini-batch SGD via Adam
- By $\theta_{k+1} = \arg\max_\theta \mathcal{L}_{\theta_k}(\theta) - \beta_k \bar{D}_{KL}(\theta||\theta_k)$

  if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} = 2\beta_k$
  else
    if $\bar{D}_{KL}(\theta_{k+1}||\theta_k) \leq \frac{\delta}{1.5}$ **then**
      $\beta_{k+1} = \frac{\beta_k}{2}$
    end if
  end if
end for

intuitive is that if the new policy is badly estimated, there is no need to use the estimated advantage function, thus clipped it. The ratio of $r_t(\theta) = \frac{\pi_\theta(a_t,s_t)}{\pi_{\theta_k}(a_t,s_t)}$ defines how different the old and new policy are, so if this ratio is larger than $+\epsilon$ or less than $1-\epsilon$, we clipped it, where in original paper $\epsilon = 0.2$. So we have the new clipped objective function as $L_{\theta_k}^{CLIP}(\theta) = E_{\tau\sim\pi_k}[\sum_{t=0}^{T}[min(r_t(\theta)\hat{A}_t^{\pi_k}, clip(r_t(\theta), 1+\epsilon, 1-\epsilon)\hat{A}_t^{\pi_k})]]$.



Source: original paper at https://arxiv.org/abs/1707.06347
Figure 4.4: Clipped Objective

From the above diagram, it visualizes two case of advantage is greater than and less than zero. It makes sense as when advantage > 0, meaning that the corresponding action is preferred over all other actions. The $r_t(\theta)$ will be increased as well but it cannot exceed $1+\epsilon$. The flatten tail indicates that the ratio is so high that this action has a high chance of appearing in the current policy rather than the sampling policy. We want to clip it so that it will avoid large policy update. Likewise, when the advantage < 0, meaning that this action has no significance. The flatten head indicates that the ratio is so low that

this action has a high chance of appearing in the old sampling policy rather than the new policy. We are not preferred to conduct this action thus reduce the value of $r_t(\theta)$. We also clipped this likelihood so that the we are not overdoing the gradient update. If not clipped, the extreme low $r_t(\theta)$ will destroy the sampling old policy.

Algorithm 3 PPO with Clipped Objective (Schulman et al., 2017)

Input: initial policy parameters θ_0, clipping threshold ϵ
for $k \in \{0, 1, 2...\}$ **do**
Collect set of trajectories D_k on policy $\pi_k = \pi(\theta_k)$
Estimate advantages $\hat{A}_t^{\pi k}$ using advantage estimation algorithm
Compute policy update by taking K steps of mini-batch SGD via Adam

- By $\theta_{k+1} = \arg\max_\theta \mathcal{L}_{\theta_k}^{CLIP}(\theta)$
- The objective function is clipped as

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) \approx E_{\tau\sim\pi_k}[\sum_{t=0}^{T}[min(r_t(\theta)A_t^{\hat{\pi}_k}, clip(r_t(\theta), 1+\epsilon, 1-\epsilon)A_t^{\hat{\pi}_k})]]$$

end for

4.3 Entity Linking

Note that all the following section assume the KG is DBpedia solely. And the other source like Wikidata can be easily mapped to instance of DBpedia. Named Entity Linking(NEL) can be viewed as an extension from Name Entity Recognition, that is highly involved in high level NLP task requiring understanding of natural language. The NEL task refers to the task of aligning mentions of entity from plain text to their corresponding entries in a knowledge graph. As what stated in the related work section, the difficulty in NEL task is The output should be a linking decision of mention to entry in KG or the mention is not found in KG. The formal definition goes as follows that given a document D with extracted mentions set $(m_1, m_2, ..., m_n)$, we are interested to train a linker that maps each m_i to entity e_i in KG or determine there is no entry can be linked in KG.

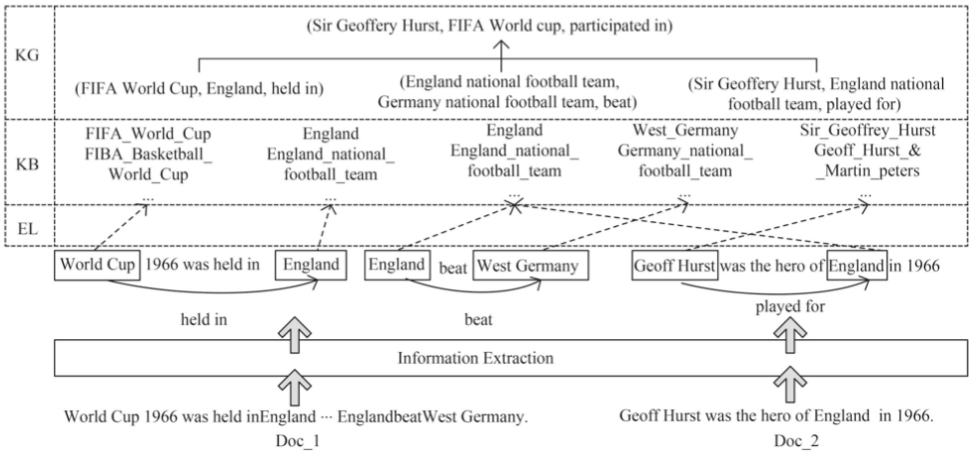


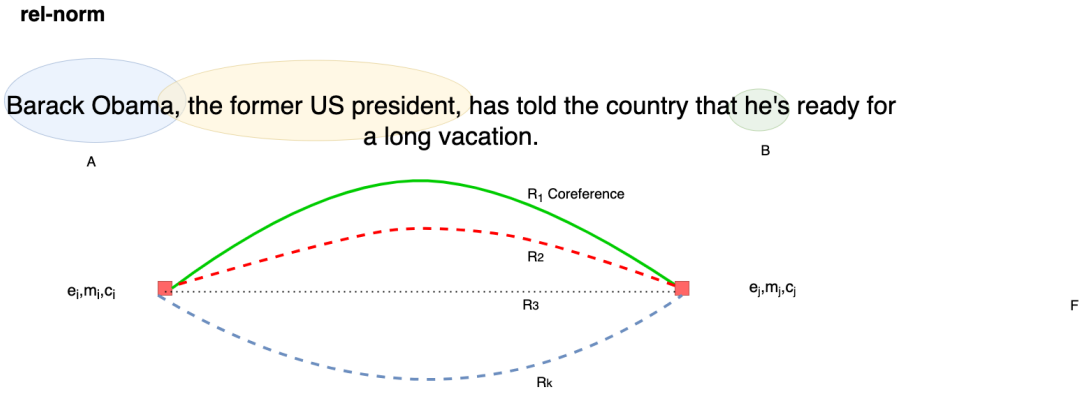
Figure 4.5: Illustration of NEL. Figure from (Li et al., 2022)

We modify the multi-relational model structure proposed by Le and Titov (2018). As it is standard to shrink the search space from the entire KG into a small set of candidate selection $C_i = (e_{i1}, ..., e_{il_i})$. This shrinking can be done effectively with heuristic function. It is necessary to introduce the basis model used in previous years. The local model only consider the local text around the mentions that is suited for mentions linking independent case. Usually, we can define a local score function to indicate how well the entity e_i matches with the local context c_i, thus $e_i^* = \arg\max_{ei \in C_i} \Psi(e_i, c_i)$. The global model on the other hand is suited for the case where each linking decision is dependent on the other mentions links, so we introduce a concept of coherence that capture the inter-dependencies between mention links. We can see that the possible combination of mention links pairs is $C_1 \times C_2 \times ... \times C_n$. Thus the simplest form of coherence is to only consider the pair-wise implication, which is given by $\sum_{i \neq j} \Phi(e_i, e_j, D)$. The global models are given by $E^* = \arg\max_{E \in C_1 \times C_2 \times ... \times C_n} \sum_{i=1}^{n} \Psi(e_i, c_i) + \sum_{i \neq j} \Phi(e_i, e_j, D)$. The key is to model the pair-wise implication between mentions Φ that can efficiently capture the relations between mentions, for example the co-reference relation. The co-reference relation is the trivial relation type compared with other semantic relation, where two mentions in a text links to one entity. We have seen some successful implementations following Ganea and Hofmann (2017) with applying representation learning on pretrained embedding of words, the local model can be the projection of entity e_i onto the attention of its context words c_i and simplified with diagonal matrix $B \in R^{d \times d}$, i.e. $\Psi(e_i, c_i) = z_i^T B \text{ attention } (c_i)$ where z_i denote the pretrained embedding of entity e_i. And the pair-wise score is similar done with diagonal matrix, that is $\Phi(e_i, e_j, D) = \frac{1}{n-1} z_i^T R z_j$.

We can extended the pair-wise implication to relation embeddings so that more underlying relations between mentions can be learned. We claims that each mention pair (m_i, m_j) has a confidence score of a_{ijk} to be assigned with a relation k, where the set of relations are modelled with K latent variables. The improved version of pair-wise implication will be $\Phi(e_i, e_j, D) = \sum_{k=1}^{K} a_{ijk} z_i^T R_k z_j$, where diagonal matrix R_k represents

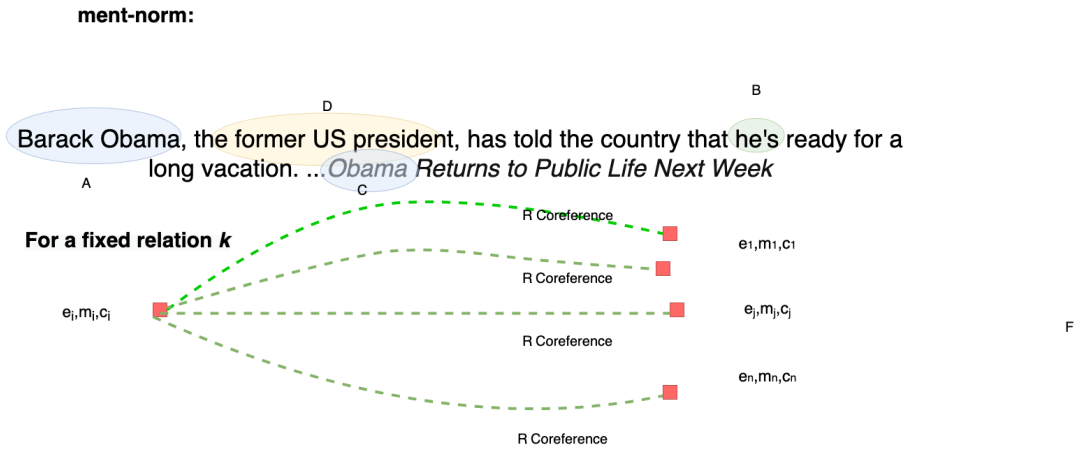
the relation k. And the weights are normalized just like the scaled dot product in attention, where here is dot product on mapping from mention to context $f(m_i, c_i)$, given by $a_{ijk} = \frac{1}{Z_{ijk}} exp\left(\frac{f(m_i,c_i)^T D_k f(m_i,c_i)}{\sqrt{d}}\right)$. The D_k is also a diagonal matrix of size $R^{d \times d}$ and Z_{ijk} is the factor of a_{ijk}.

As the weights coefficient a_{ijk} are sub-scripted with entity e_i, mention m_j and relation r_k. This normalization gives us two choice, one is relation-wise normalization(rel-norm) and the other is mention-wise normalization(ment-norm). From below two figures, we can see the difference between two normalization methods, where each color represents a unique relation. Note that the function f maps (m_i, c_i) onto a R^d space, that simply is a single-layer feed-forward neural network with *tanh* as activation function. Also add a dropout regularization to this network with ratio approach to 1.



Source: original
Figure 4.6: relation-wise normalization

For the relation-wise normalization, we are normalise against $\sum_k^K \alpha_{ijk} = 1$, that is we are seeking the best relation to explain for the fixed two entity and their surrounding text. Combined with about scaled dot product on a_{ijk}, we can rewrite the factor as $Z_{ijk} = \sum_{k\prime}^K exp\left(\frac{f(m_i,c_i)^T D_{k\prime} f(m_i,c_i)}{\sqrt{d}}\right)$. So the relation embedding $R_{ij} = \sum_{k=1}^K \alpha_{ijk} R_k$, and our pair-wise implication will be $\Phi(e_i, e_j, D) = \sum_{k=1}^K z_i^T R_{ij} z_j$. It is obvious that α_{ijk} here is the probability of the mention pair have the relation k, that depends on the relation embedding R_k.



Source: original
Figure 4.7: mention-wise normalization

For mention-wise normalization, we are normalise against $\sum_{j=1,i\neq j}^{n} a_{ijk} = 1$ that is we are fixing the relation k, and try to find the mention pair$(m_i, m_j) i \neq j$ such that m_i and m_j forms the relation k. The pair-wise implication is still given by $\Phi(e_i, e_j, D) = \sum_{k=1}^{K} a_{ijk} z_i^T R_k z_j$ and some changes need to be made on normalisation factor Z_{ijk} such that $Z_{ijk} = \sum_{i_0=1 \wedge i_0 \neq i}^{n} exp\left(\frac{f(m_i,c_i)^T D_k f(m_{i_0},c_{i_0})}{\sqrt{d}}\right)$.

Consider the case with mention norm, there are two scenarios that are likely to happen. One is we cannot find a another mention m_j that has any relation k to the starting mention m_i. The other is that the coefficient α_{ijk} is high for multiple relations that it cannot distinguish which one is the best decision. If we shift the perspective, the relation-norm model can handle these two extreme cases theoretically. Consider that the relation matrix R_k is sparse, meaning that no relations can be distinguished. The first case is easily solved but it cannot tackle the second case directly since there is no way for relation-norm model to assign large coefficient as it is comparing fixed mention pair. But if all other relation cannot be found upon a number of comparison in relation-norm, it might ended in few top-high coefficient relation in mention pair. Furthermore, there exists a third extreme case where the document D is so short that for the mention-norm model, some mention does not exist one of the K relation at all. We have seen in the figures that co-reference is the most trivial relation without deeper semantic understanding. If one mention is non-anaphoric in D, that is the mention does not share any repeated occurrence with any other mentions, it will violate the the normalisation rule of mention-norm model $\sum_{j=1,i\neq j}^{n} a_{ijk} = 1$. This can be prevent with adding a padding mention along with its padding entity that always has strong α_{ijk} to be linked together, resulting in other relations with smaller probability.

As a basis, we are going to talk more about how does hidden markov model(HMM) and

conditional random field(CRF) are related since a lot of NLP models cannot go without the existence of CRF nowadays. HMM can be thought of a special case of conditional random field. I will first review on what is HMM and its implication.

As elaborated above in reinforcement learning section, markov models are usually deal with a sequence of data. Specifically, the data sequence we can observe X_i that are dependent on hidden state sequence y_i. Traditionally, we refer the probability that y_i gave rise to X_i as emission probability, i.e. what is probability we observe the X_i given the hidden state y_i. Likewise, the transition probability is defined as the intersection between hidden states y_i and y_j for $i \neq j$. The limitation here obviously is that the hidden state y_i can only be dependent the previous state y_{i-1}. Together, this forms a generative model on the joint $P(y, X)$, that explains how the data sequence we directly observe are generated with another set of unobserved hidden states. This relation is captured by one single equation $P(y, X) = \prod_{i=1}^{N} P(y_i|y_i - 1) \cdot P(X_i|y_i)$.

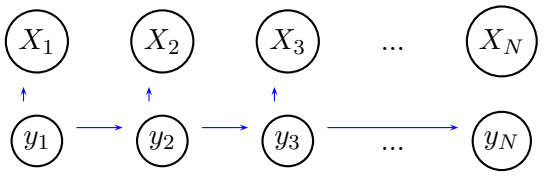


Figure. simple HMM

We can immediately see saw drawback of general HMM model that it only models static relation between hidden state and observed state. For example, we are modeling part-of-speech tags by HMM and y_1 is verb. X_1 is a verb that risen by y_1 but the emission probability result is often static meaning that if changing the position of X_1 occurring in a sentence, the probability is unchanged. Another drawback would be it only models limited dependency, where in the graph the connection are arrows that latter y_i only dependent on the nearest state y_{i-1}. It eliminates a lot of possible connection dependency. A CRF improves these issues, but assuming the connection to be bi-directional and can happen on any pair of any X_i and y_j. Hence, we can formally introduce CRF model as discriminative model to capture the dependency with conditional probability $P(y|X)$. There is a fundamental difference here between the CRF and HMM is that CRF primarily aims differentiate between observed variables and not modelling how the hidden variables got generated in the first place. Allowing the potential of modeling with any dependency between hidden variable y_i can a disaster for computation feasibility, thus often the case linear chain CRF is the comprise approach that only allows for linear connection inside hidden variable y_i. It is formulated as $P(y|X) = \frac{1}{Z} exp(\sum_{i=1}^{N} F(X, y_j, y_i, i))$ where Z is a normalisation constant. To notice that $F(X, y_{i-1}, y_i, i)$ is basically the linear combination of feature functions that you determine before the training, which is $F(X, y_{i-1}, y_i, i) = \sum_j w_j f(X, y_{i-1}, y_i, i)$. Among all the potential feature function, one can directly use word embedding for feature function. The weights w_j are trained by gradient descent.

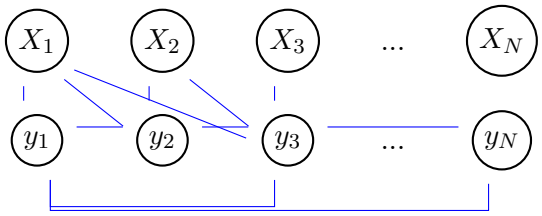


Figure. General CRF

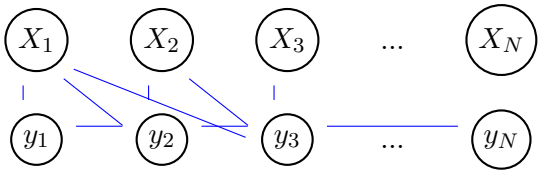
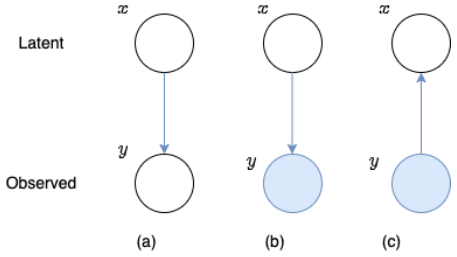


Figure. Linear chain CRF

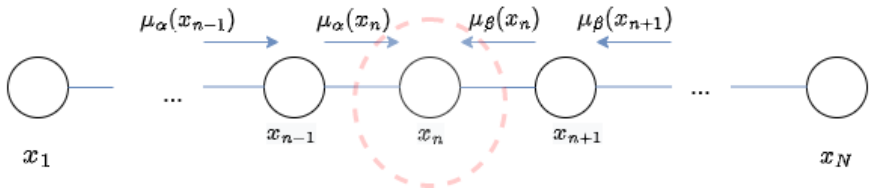
Now that we have learnt how generally HMM and CRF are working behind the scene. With the help of the basis of graphical model theory in background section. Next, we will talking about the most crucial part, which is inference on graphical model. The definitions of different graphical structures and nodes support the goal of exploiting the graphical structure to do efficient inference. We are interested to compute the posterior distributions regarding those nodes having close connection to the observed values. Started by talking about exact inference, which would be the foundation of following content of approximate inference algorithms. From below graph, applying the same factorisation rule we show above, sub-graph(a) represents $p(x, y) = p(x)p(y|x)$, sub-graph(b) represents $p(y) = \sum_{x\prime} p(x\prime)p(y|x\prime)$ and sub-graph(c) represents $p(x, y) = p(y)p(x|y)$. The connection is established by Bayes' rule, which is just $p(x|y) = \frac{p(y|x)\cdot p(x)}{p(y)}$. The term $p(x)$ and $p(y|x)$ are known by factorisation, with the help of sub-graph(b), we can thus compute the posterior $p(x|y)$.



Source: original
Figure 4.8: Simple Inference

Another trivial example is with the linear chain, recall that the joint distribution of undirected linear chain graph is given by $p(x) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{3,2}(x_3, x_2)...\psi_{N-1,N}(x_{N-1}, x_N)$. And we still assume each N nodes has discrete K states, it implies that potential function of $\psi_{N-1,N}(x_{N-1}, x_N)$ has K^2 parameter, resulting $p(x)$ being $(N-1)K^2$ parameters. If the desired inference is to compute the marginal of one x_n where $n \in [1, N]$, it would have to sum up all the nodes before x_n and all the nodes after x_n, which is

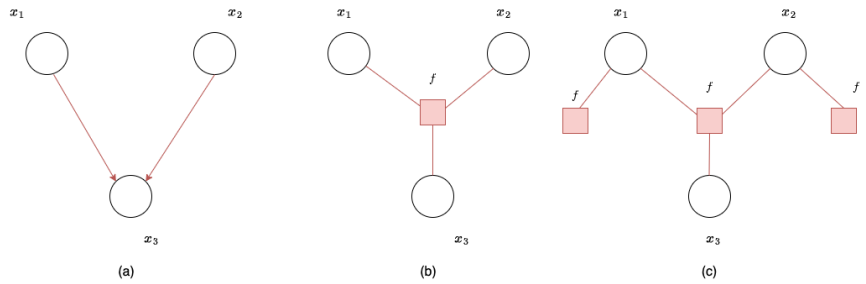
$p(x_n) = \sum_{x_1} ... \sum_{x_{n-1}} \sum_{x_{n+1}} ... \sum_{x_N} p(x) \in \mathcal{O}(K^{N-1})$. An easy trick is to recall that the clique explained the intersection between nodes completely, so there is no need to perform summation over each variable. We can have below transformation by using clique, $p(x_n) = \frac{1}{Z} \underbrace{[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n)...[\sum_{x_2} \psi_{2,3}(x_2, x_3)[\sum_{x_1} \psi_{1,2}(x_1, x_2)]]...]}_{\text{message } \mu \text{ before n}} \cdot$
$\underbrace{[\sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n)...[\sum_{x_{N-1}} \psi_{N-2,N-1}(X_{N-2}, X_{N-1})[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)]]...]}_{\text{message } \mu \text{ after n}}$. We can further rename two chunk of messages as $\mu_\alpha(x_n)$ and $\mu_\beta(x_n)$, thus $p(x_n) = \frac{1}{Z}\mu_\alpha(x_n) \cdot \mu_\beta(x_n)$. It is also obvious to see that this evaluation for each node takes total $N-1$ times of K^2, thus it belongs to $\mathcal{O}(NK^2)$. Note that this complexity is rather high, which is the worst case of fully connected graph. Speaking of cliques, we also briefly introduce junction tree algorithm that is a classic means of exact inference in undirected graph. It eliminates loops by making cliques that reduces to a clique tree. The solution is exact but its complexity is exponential with respect to the number of nodes in the largest clique.



Source: original
Figure 4.9: Message Passing on Linear-chain Graph

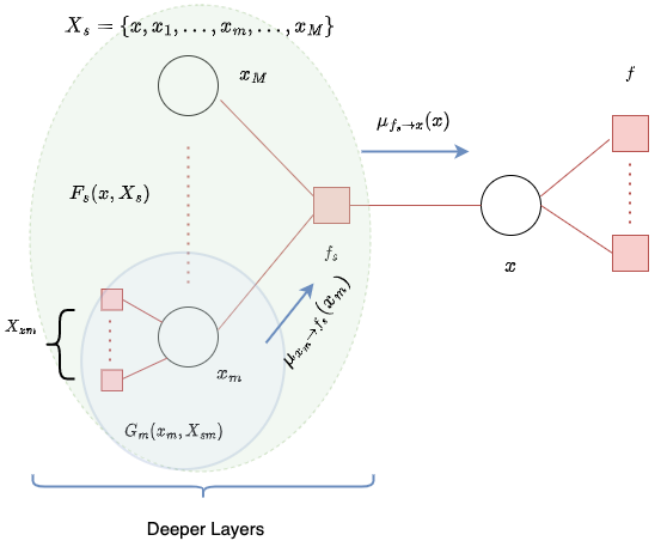
But how can we improve in message passing? The idea is to doing recursive evaluation from the bottom. In a complexity view, if you first do the message passing with smaller tables, and put multiplication behind addition, it can further shrink the complexity in average case. The evaluation of $\mu_\alpha(x_n)$ is $\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n)[\mu_\alpha(x_{n-1})...]$, which we first compute the bottom message $\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2)$. By doing so iteratively, until we finally reach the node x_n. Similarly, the evaluation of $\mu_\beta(x_n)$ is $\sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n)[\mu_\beta(x_{n+1})...]$, which we first compute the bottom message $\mu_\beta(x_N) = \sum_{x_{N-1}} \psi_{N-1,N}(x_{N-1}, x_N)$. It is obvious to note that the prior message must have a lot of repeating content that forms a new message at the next node, so if we store all the intermediate message, we would ended in a complexity of $\mathcal{O}(N^2M^2)$ where $M << K$. Compared with above, for each node, the evaluation is $\mathcal{O}(NK^2)$. Lastly about messaging towards middle two neighbouring nodes x_{n-1} and x_n in linear chain, following above definition, it is $p(x_{n-1}, x_n) = \frac{1}{Z}\mu_\alpha(x_{n-1})\psi_{n-1,n}(x_{n-1}, x_n)\mu_\beta(x_n)$. Note that the idea of always doing addition before multiplication is the same strategy behind sum-product algorithm that we will discuss later. The complexity would be lower if we always push down the addition. Another variant is called max-product, which instead of doing addition, it selects the maximum and do multiplication on the maximum. It will have even better complexity than sum-product, as $a \cdot \max(b, c) \leq a(b + c)$.

We have seen in detail that how undirected and directed graph make decomposition factor by parent nodes in graph. However, a factor graph explicitly define a set of factor nodes representing factorisation between nodes. Formally, for a subset of variables x_s, it have a factor f_s that is a function this set x_s. So the joint probability is $p(x) = \prod_s f_s(x_s)$. Although factor graph can be thought as a transition product from directed graph and undirected graph. Compared with undirected graph, factor graph naturally contains more information. Consider a case that two factors f_i and f_j are connected by same set of variables xk. In the undirected graph form, it would encapsulate both f_i and f_j into a single clique, thus one potential function. Compared with directed graph, the factors are already normalised and don't need to track conditional dependency on graph. It is important to notice that although some factor graphs can seems different, but they can describe same probability factorization. For example, in the below graph, all those three sub-graph describe the same factorization of $p(x_1)p(x_2)p(x_3|x_1, x_2)$ from directed sub-graph(a). From this example, we can conclude the general steps for converting directed graph to factor graph, first create variable nodes in factor graph according to the nodes in directed graph and then create factor nodes by original conditional dependency. Add to links to factor and variable nodes is the last step. Similarly, converting from undirected graph require to set the factor from the clique potentials after creating factor nodes for each maximal clique. Due to these property, factor graph are better for computing inference.



Source: original
Figure 4.10: Factor Graph Example

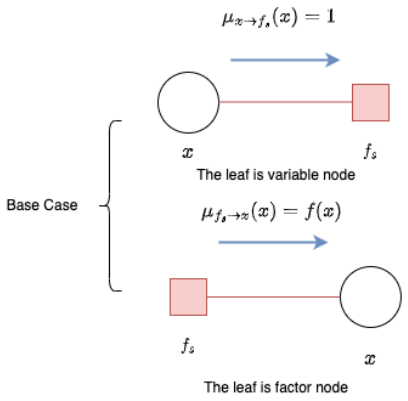
The sum-product algorithm specifies how to evaluation local marginal overs subsets of nodes. With the framework of factor graph we just introduce, the underlying target graph is undirected or directed tree. It could also be a polytree, so that the converting result factor graph has a tree structure. The marginal distribution is just $p(x) = \sum_{\mathbf{x}\backslash x} p(\mathbf{x}) = \sum_{\mathbf{x}\backslash x} \prod_s f_s(\mathbf{x}_s)$. It is important to note that $\mathbf{x}\backslash x$ is the set of variables in x without x itself. From above, we also know that the joint of factor graph is product of each relevant factor nodes, which is $p(\mathbf{x}) = \prod_{s \in ne(x)} F_s(x, X_s)$, where $ne(x)$ is the $x's$ neighbouring set of factor nodes and X_s is all the variable in the subtree rooted in x that are connected to x by the factor node. Note that we have this relation $p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s) = \prod_{s \in ne(x)} F_s(x, X_s)$, where F_s should be product of $\{f_s\}$. This is captured in the below graph.



Source: original
Figure 4.11: Sum Product Step Case

Applying rules above, we get $p(x) = \prod_{s \in ne(x)} \sum_{X_s} F_s(x, X_s) = \prod_{s \in ne(x)} \mu_{f_x \to x}(x)$. Here we have rename from $\sum_{X_s} F_s(x, X_s)$ to $\mu_{f_x \to x}(x)$, and this is the message from factor node to the variable node. Some may question about why we can move the summation sign $\sum_{X_s}$ inside the product sign $\prod_{s \in ne(x)}$, this is because of the bipartite natural of factor graph. A sub-graph behind $\sum_{X_{s_2}}$ is not intersecting another sub-graph behind $\sum_{X_{s_1}}$. If there are M variable $\{x_1, x_2, ..., x_M\}$ in $F_s(x, X_s)$. We also denote $G_m(x_m, X_{sm})$ as all the factors linked to the variable node x_m, where X_s is the set of all variable in the subtree. So the contributing message from x_m to factor node f_s is $\mu_{x_m \to f_s}(x_m) = \sum_{X_{sm}} G_m(x_m, X_{sm})$. So factorise this sub-graph, it follows $F_s(x, X_s) = f_s(x, x_1, ..., x_M) G_1(x_1, X_{s1}) ... G_m(x_m, X_{sm}) ... G_M(x_M, X_{sM})$. From the above graph, we can now introducing the message from variable node to factor node, $\mu_{f_x \to x}(x) = \sum_{x_1} ... \sum_{x_M} f_s(x, x_1, ..., x_M) \prod_{m \in \{ne(f_s) \setminus x\}} [\sum_{X_{xm}} G_m(x_m, X_{xm})]$
$= \sum_{x_1} ... \sum_{x_M} f_s(x, x_1, ..., x_M) \prod_{m \in \{ne(f_s) \setminus x\}} \mu_{x_m \to f_s}(x_m)$.

From above discussion of message passing, we can finally summarise the steps of computing the income message. First calculate the products of all the incoming message that feeds into the factor nodes by $\prod_{m \in \{ne(f_s) \setminus x\}} [\sum_{X_{xm}} G_m(x_m, X_{xm})] = \prod_{m \in \{ne(f_s) \setminus x\}} \mu_{x_m \to f_s}(x_m)$. Secondly multiply last step result with the factor $f_s(x, x_1, ..., x_M)$ associated with that node. Lastly, computing the marginal of all the variables with respect to the incoming messages by summing from x_1 to x_M. Now we are clear about how the step case in recursion happening behind the graph, but what's the base case? Intuitively, the base case always start with the bottom leave nodes. Following the convention of case discuss, we still have to discuss whether the leave node is a factor node or variable node. As shown in the below graph, it specifies the base case.



Source: original
Figure 4.12: Sum Product Base Case

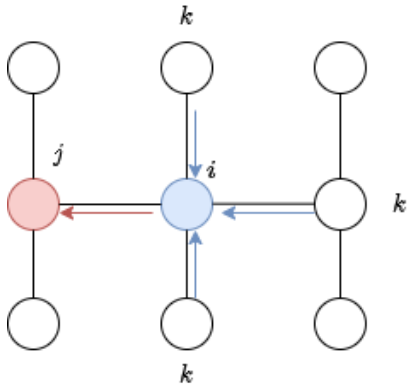
We have talked a lot of graphical modeling from naive brute force to message passing(sum-product). We can now wrap-up everything, and formally state what belief propagation is. The rule for belief propagation is that the node can only send message to their neighbours. When their neighbours receive such message, they can choose to pass it on. The same way works for receiving message. The message at each node is computed by multiplying the message from all others nodes with the potential function coming from unary and binary intersection. We slight change notation of message from μ in factor graph to m. As the an example from below graph, the message from node i to node j is defined as:

$$m_{i \to j}(x_j) = \sum_{x_i} \underbrace{\psi(x_i)}_{\text{unary}} \underbrace{\psi(x_i, x_j)}_{\text{binary}} \prod_{k \epsilon ne(i) \backslash j} m_{k \to i}(x_i)$$

And the node marginal probability is just the belief, that can be computing by multiplying all the k message with, which is

$$p(x_i) \propto \psi(x_i) \prod_{k \epsilon ne(i) \backslash j} m_{k \to i}(x_i)$$

The actual message passing in belief propagation is functioning very similar to sum-product algorithm we illustrate above. Initialize all μ be one and all belief β to be local self-potential. The message from node i to node j is $\sigma_{i \to j}$ is computed by marginalizing over neighbouring belief β_i. The node j has a belief β_j that will be updated when it receive message $\sigma_{i \to j}$ from its neighbour node i. The new $\beta_j = \beta_j \frac{\sigma_{i \to j}}{\mu i \to j}$, where $\mu i \to j$ is the message that last time node j receives. After this update, renew the message by setting $\mu i \to j = \sigma_{i \to j}$.



Source: original
Figure 4.13: Message in Belief Propagation

If the belief propagation above is applied on a graph with loops, messages may keep circulating and never come to an end. A loopy belief propagation is applying the exact same algorithm on graph with loops. Although the paper of loopy belief propagation comes out a decade ago, it has received enormous application over CV and NLP in the past few years. Although enormous past research studied when loopy belief propagation works under which subjects, we just assume it has a good chance of success. Like the belief propagation, we start with two nodes i and j. Calculate their belief by $b(x_i) \propto \psi(x_i) \prod_{j \epsilon ne(i)} m_{j \to i}(x_i)$ and $b(x_j) \propto \psi(x_j) \prod_{i \epsilon ne(j)} m_{i \to j}(x_i)$. It constantly send message forward and backward by $m_{j \to i}(x_i) = \sum_{X_j \backslash x_i} \psi(X_j) \prod_{a \epsilon ne(j) \backslash i} m_{a \to j}(x_a)$ and $m_{i \to j}(x_i) = \prod_{b \epsilon ne(i) \backslash j} m_{b \to i}(x_i)$. We stop the algorithm after a fixed number of iterations to check if there is a significant belief update. If there is not for a number of step, we just terminate.

We created a conditional random field by the global model following Ganea and Hofmann (2017) and Le and Titov (2018) and use max-product loopy belief propagation to estimate the maximal probability. With the detailed elaboration of HMM and CRF, it is natural that CRF would fits the task best.

4.4 Fact Checking(FC)

As mentioned in related work, fact checking is regarded as a special form of triple classification, which evaluates the correctness or factualness of an input triple. The output should be a score rather than a class label. This can be also viewed as an extension of knowledge graph completion, thus we can classify the approach by embedding-based models and rule-based models. Fact checking is only a sub-domain in triple classification with few works done in previews years, meaning that it would be a challenging task. The main challenge is still to deal with the incompleteness of knowledge graph. It is recognized that the embedding-based approach performs well in un-linked triple input, meaning that we cannot find a path to connect the subject entity to object entity. Intuitively, it is useful in verifying the triples within the embedding space. But the

rule-based approach is better in linked triple input by rule mining. The rule-mining technique often use Horn logical to perform multi-hop reasoning to assert the factualness. We will see in this section that how can we ensemble these two genres work together. We mainly use the work done by (Kim and Choi, 2021) as the FC modular. First, we will cover how the rules are mined given a specific source of knowledge graph. The rules here means positive rules and negative rules. The basic idea is to fix subject and object so that we can aggregate general predicate to form evidence path. After such extraction, the base form of a bunch of factual triples could be $(x, education, y) \leftarrow (x, almamater, y)$. In order to assign a score of 1, there has to be a positive evidential path that satisfy $(x, almamater, y)$. A path to satisfy this could be $(x, master, y)$, $(x, undergraduate, y)$ and so on. We also define weighted logical positive rule, that is for a set of positive rule are ended in $(x, education, y)$, for example $(x, education, y) \leftarrow (x, residence, z) \wedge (y, locate, z)$ should has less weight in validity. In this peculiar positive rule, we know that if some lives in a same place as the university, it does not necessarily means that this person goes to this university for education. We can view the rule of $(x, education, y) \leftarrow (x, almamater, y)$ as the only golden rule, and the other rule with less weight to be true. Likewise, there are also negative rule and corresponding weighted negative rule. Take the same example of education, we can say that $\neg(x, education, y) \leftarrow (x, almamater, z) \wedge (z \neq y)$ is a golden negative rule while the rule of $\neg(x, education, y) \leftarrow (x, residence, z) \wedge (z \neq y)$ should have less weight in asserting negativity. We only make an critical assumption, open world assumption, meaning that the triple missing in the knowledge graph is not false. We will use negative sampling to create more false fact triples to be used as counter example to negative intermediate weighted positive rules. Following Ortona et al. (2018), we construct the negative sampling based on extended local closed world assumption(E-LCWA), that generates false triple by replacing the subject or object that is adjacent node in knowledge graph. The essence of doing so is to expand the negative path that help us to negate unseen triples. Ortona et al. (2018) also claimed that this assumption is true for all the functional predicates but it could be violated for non-functional predicates. The function predicates means that for an given object entry, there exist more than one object values linked to it and the span can be growing with respect to time. An example would be child predicate that Donald Trump currently has five children, but the number could increase in the future. According to analysis that more than 85 percent of the DBpedia object are considered non-functional, so we want to make changes on E-LCWA. We furthermore introduce the distant local closed world assumption (D-LCWA) by (Kim and Choi, 2021). The idea behind it is to constrain the maximum length of adjacent sub-graph when replacing the subject or object. Considering doing E-LCWA on a special case of relative predicate, the chance are that 47 percent of generated false triple turns out to be true. In D-LCWA, the object will be replaced by another object, which is not directly adjacent to the subject and the shortest distance is less than maximum path length. Now that we have all the true facts and false facts on hand, we can create rules. The rule generation for negative rules and positive rules are identical, the only difference is the input triples. For each triple from true facts or false facts, we construct a local graph centered at input triple with

path length less than maximum path length. And we need to to unified different URIs that connected to the same entity. We add new predicate type $\neq$ to compare all other similar entity with same type. For entity with numerical type, adding new predicate type < and >. After construction of local graph, we extract rule instance from it. It can be expected that the generate local graph starts with subject and has a body with a span of maximum path length. We use the breadth-first search(BFS) to expand the subject with 1-hop. This will give us all the path from minimum length to maximum path length. We furthermore trim the set of found path by removing all the path whose tail is not the input object. Also removes all the path that can form a cycle or contains the special predicate we introduce <,> and $\neq$ more than once. From the trimmed set of path, we generalize each set with a rule by replacing concrete entity with variable and follow the direction of the edges that it originally forms.

Next, we will talk about how to assign weights to generated rules. We want to assign a weight score $\in [0.0, 1.0]$ to express how logical valid the given rule is. The intuition behind is to represent the weight by the fraction of the fact triple that the input rules covers versus the negative rule. The weight score is 0.0 meaning that the rule is complete valid. It indicates that we must can find evidential path in the DBpedia. Different from existing model, KStream, KLinker and PredPath, we use both correct and false covered example to support a rule. We denote that the set of correct triples covered as $E_{correct}$ and false triples covered as E_{false}. And the random variable $C_r(E_{correct}$ denotes the number of fact triple in E that is covered by rule r. Since the above rule extraction from local tree assume the variant path is constrained to same entity class, to offset this limitation we create unbounded rule that $U_r(E_{correct}$ denotes the number of fact triple covered by the unbounded rule r. The unbounded rule allow the different object entity to be changed to other unique entity. For example, given the rule $(x, classmate, y) \leftarrow (x, almamater, z) \wedge (y, almamater, z)$, where the variable z can be changed to other entities. The weights measure proposed by Ortona et al. (2018) is: $w_2(r) = \alpha \frac{1-|C_r(E_{correct})|}{|U_r(E_{correct})|} + \beta \frac{|C_r(E_{false})|}{|U_r(E_{false})|}$, where α and β are constraints variables with $\alpha + \beta = 1$. It is easy to see from the equation that it leans to the rules with more correct instances. According to analysis, given a extreme case that if a rule covers small amount of both correct and false triple, i.e. the $C_r(E_{correct})$ and $C_r(E_{false})$ are less than 10 percent will ended in a weight score $w_2(r)$ close to valid. We want to prevent this from happening. Following the modification by Kim and Choi (2021), we re-implement the weights measure by $w_c(r) = 1 - \frac{C_r(E_{correct}) - \frac{1}{\gamma} C_r(E_{false})}{C_r(E_{correct}} \cdot (1 - w_2(r))$, where $\in [0.0, 1.0]$ is introduced to control the noise brought by the negative sampling generated false triples. Here comes the last step of the FC pipeline, which is to produce a truth scoring for the input triple. The intuition is to produce the scoring based on an ensemble of all rules that covers this triple. If the input triple is covered by any rule that has high weights meaning that there is a high chance that we can find a strong evidential path in the KG to support the factual of this triple. We can just find the most weighted positive rule r_p^* and negative rule r_n^* from the rule set that can satisfy this triple. So the score is given by $score(s, p, o) = \frac{((1-w_c(r_p^*)) - (1-w_c(r_n^*)) + 1)}{2}$. In an ideal situation, the term $w_c(r_p^*)$

returns 0.0 and $w_c(r_n^*)$ should corresponding return 1.0, together the ending truth score is 1.0 representing the input triple is true.

Evaluation

5.1 Overview

Due to the nature of this project, it has a non-standard evaluation schema compared with other projects in ML. We provide an evaluation schema here to clear the thought.

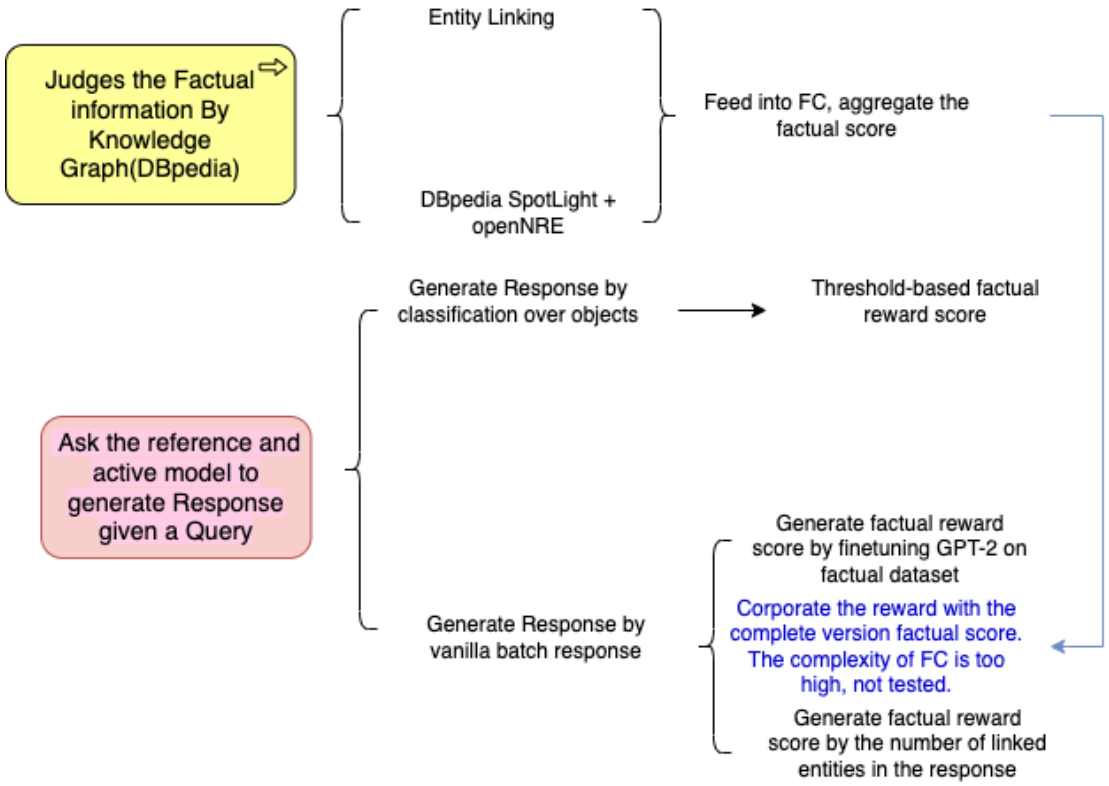


Figure 5.1: Evaluation Schema

We make sure that the complete code has been uploaded to gitlab repo. and the evaluation result of trl is upload to wandb gpt-2 query eval, in case the reader want to review the complete result. For more detail, please refer to the README file.

5.2 Factual Pipeline

There does not exist known published dataset or benchmark that challenges model's factual decision sorely based on information from knowledge graph. We have identified some recently published datasets that share some degree of the same objective but is not applicable to our design. Although I have developed a factual dataset of triple-form query and response by negative sampling from DBpedia, it supports for the evaluation for transformer reinforcement learning framework below. It is not sufficient to test the designed factual pipeline on sentence-level truthfulness. It is very likely that the missing dataset would be proposed in the near future, which is discussed in the future work section.

Due to these reasons, providing a proof-of-concept over one concrete example for factual pipeline is enough.

- TabFact dataset(Wenhu Chen and Wang, 2020) consisted of manual annotated statements and classify output is either entailed and refuted. The input is Wikipedia table and its caption. It is the first dataset to test the table-based model with its ability of doing inference on structured data.
- DialFact dataset (Gupta et al., 2021) consisted of annotated conversational claims and evidence from Wikipedia. It tests the model on predicting a dialogue and generate a response that is supported, refuted or not sufficient information provided.
- Hover(Hoppy Verification) dataset (Jiang et al., 2020) tests the models to extract facts from Wikipedia articles given a claim. The model should classify whether the claim is supported or not supported. The challenging part is that most of 3 and 4 hops claims are in multi-sentence that tests the model with understanding of long-term dependent relations. No benchmark score has been reported to this dataset.

So we conclude not to running on any dataset. My supervisor also agree with my opinion on this issue. With the package version of code ready, it is easy to run after any appropriate dataset being proposed in future. We consider short text input for the package, which can be changed according to the requirement.

5.2.1 Experimental Setup

The entity linking starts with detecting mentions. This is equivalent to name entity recognition. As mentioned in above content, NER is considered a solved problem with

micro F1	Aida-A	Aida-B	ace2004	aquaint	clueweb	msnbc	wikipedia
mention norm	0.9206	0.9315	0.8853	0.8867	0.7764	0.9350	0.7732

Table 5.1: F1 score of evaluating mention-norm model against standard dataset

many published tool-kits. In my experiment, using Spacy, Stanford NER tagger and Flair does not have much difference. We adopt Flair to detect text span.

As suggested by Le and Titov (2018), filter the top 30 detected mentions as candidates and only selects $k_1 + k_2 = 7$ entities. The first group $k_1 = 4$ entities are selected based on prior probability $\hat{p}(entity|i-th\ mention)$. Following Ganea and Hofmann (2017), this prior is compute by minimum of 1 or $\hat{p}_{wiki}(entity|i-th\ mention) + \hat{p}_{yago}(entity|i-th\ mention)$. The prior on Wikipedia can be computed by summing hyperlinks counts from CrossWiki corpus (Spitkovsky and Chang, 2012). The second group $k_2 = 3$ entities are selected based on similarity score over window context around mention, where the similarity is given by $e^T \sum_{w \epsilon \mathbf{w}} w$. The entity embedding e and $\mathbf{w}$ as the surrounding text of window size 50. The e are from Wikipedia2Vec package that is train on Wikipedia-2019 dump.

We use Glove wording embedding and entity embedding from Ganea and Hofmann (2017) throughout. Train on 200 epochs, learning rate is 1e−4 for Adam optimizer and specify the loopy belief having 10 loops. We retrain the model on AIDA-train, AIDA-A for validation and AIDA-B for testing and found out that mention norm model has the best performance. The test result for mention norm are shown in the below table:

More on AIDA-CoNLL dataset, it is the most commonly used dataset on linking problem. It contains a number documents and assigns mentions in one text to their entities. The entities are identified by YAGO2 entity name, Wikipedia URL(ID) and Freebase mid.

5.2.2 Concrete Example

We will provide two running examples over one paragraph of news, one is purely entity linking, the other is by traditional method DBpedia Spotlight with openNRE (Han et al., 2019).

The example input text is "Trump lost the 2020 presidential election to Joe Biden but refused to concede defeat, falsely claiming widespread electoral fraud and attempting to overturn the results by pressuring government officials, mounting scores of unsuccessful legal challenges, and obstructing the presidential transition. On January 6, 2021, Trump urged his supporters to march to the Capitol, which many of them then attacked, resulting in multiple deaths and interrupting the electoral vote count."

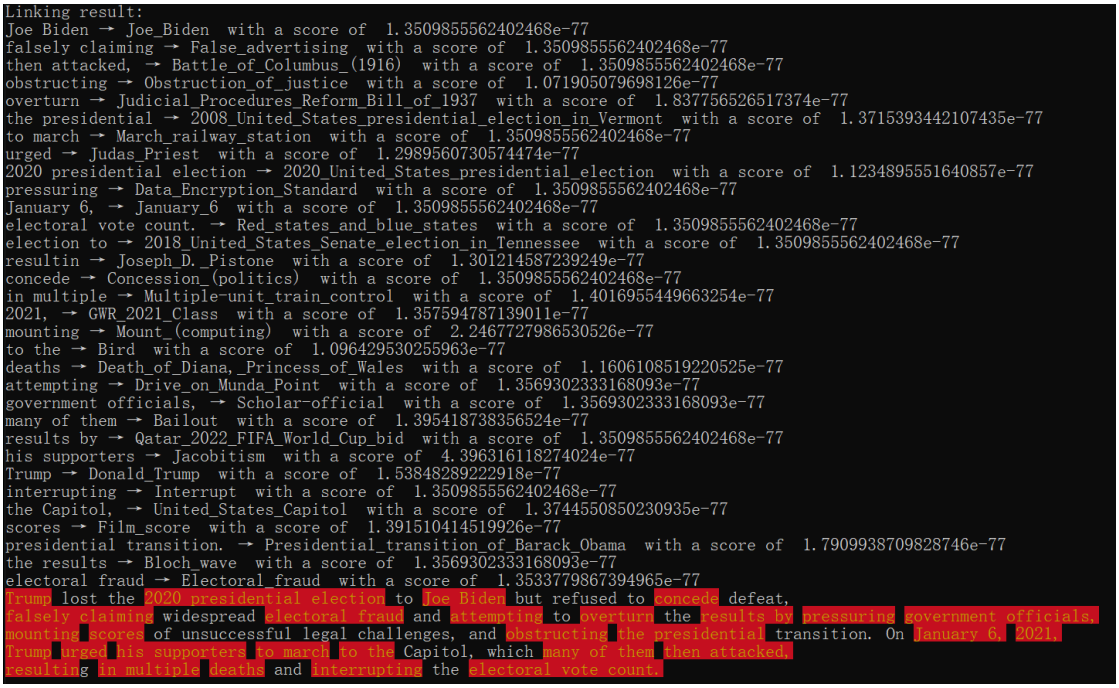


Figure 5.2: Entity Linking Example

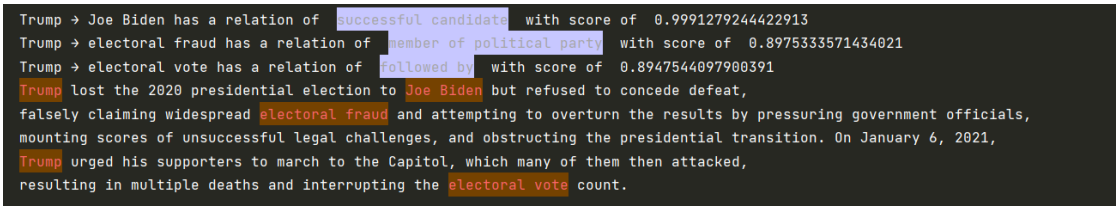


Figure 5.3: Spotlight with RE Example

Next is to do human aligning of relations. We manually transfer the relation extracted above into predicates that can be found DBpedia. And we feed each triple from mapped predicate into FC modular. The aggregated score is computing by taking the mean of factual score from each triple. This example here demonstrates the drawback and the potential flow of the FC modular. The found candidate triples are $(Joe\_Biden, is\_Candidate, Donald\_Trump)$, $(Donald\_Trump, political\_party, electoral\_fraud)$ and $(Donald\_Trump, followed\_by, electoral\_vote)$ that each represents an edge case that a FC modular should handle. If we investigate into the first triple, $is\_Candidate$ is not a found relation type in DBpedia. There are mainly three possibilities to map a unfounded relation to a relation in DBpedia. One is that the input relation is not true from all the path between subject and object. Another is that the input relation is a semantic aggregation of the relation path from subject to object. $is\_Candidate$ should be semantic equivalent to *successor* predicate given its surrounding text. There does not exist a proposed model that achieve this functionality. The other possibility

is that the input relation is exact one edge or multiple edge that can be inferred with rule-search. The FC modular deals with such case. The second input triple is untrue, that *political_party* is not a correct relation given the object and subject. The third input triple is another special case, that we should discuss. The relation $followed\_by$ is not something can be inferred by contextual information or aggregating sub-graph from subject to object. We should define a basic types of relations that generally depicts the connectivity between two entities. The relation $followed\_by$ is only suggesting the two entities is related in knowledge graph. From below result of FC modular, we can see the flaw is that it tends to assign factual score of 0.5 over unknown relations.

```
conf/conf-dben.json
testing/example.tsv
testing
Creating Rules
Finish stage 1
s is  Joe_Biden
p is
o is  Donald_Trump
Joe_Biden  Donald_Trump
s is  Donald_Trump
p is  electoral_fraud
o is  political_party
Donald_Trump electoral_fraud political_party
s is  Donald_Trump
p is  electoral_vote
o is  followed_by
Donald_Trump electoral_vote followed_by
Score of  0.5 is assigned based on unrecognizable relation
Score of  0.5 is assigned based on unrecognizable relation electoral_fraud
Score of  0.5 is assigned based on unrecognizable relation electoral_vote
```

Figure 5.4: Facting Checking Example

5.3 Transformer Reinforcement Learning Pipeline

To ensure fast execution, we setting the DBpedia size to rather small. The shrunken dataset comprises of 100000 triples from DBpedia 2014 dump. Note that the size could have been much larger, my computer only deals with complexity of this level. Modular code for constructing the dataset has been provided, that one can easily enlarge the dataset size. The experiment setup is that batch size being 64, forward batch size being 8, the total steps being 6400 and the epoch for PPO being 4. The running time for 100 batches execution takes around 6 to 10 hours on GeForce RTX 3090. This section contains three schemes for testing trl, which are proven to have moderate execution time. The reward below is added with an extra small float number that ensure the smooth of PPO. It does not affect the integrity of the result.

5.3.1 Baseline

As a baseline to the whole architecture, we conduct query-response version as factual checking against DBpedia. The query is made up from existing subject and predicate.

And the response is derived from GPT-2 as a classification task, meaning that the response is restricted to come from the known object list in the DBpedia. This way allows us to construct a baseline without considering how triple compositing the sentence. The maximal reward would be one if the predicted response is the object from the original triple and the medium reward would be zero if the predicted response is not matching any object. So the minimum reward would be the case that response is linked to the wrong subject.

It should be noted that it is not possible to fine-tune the GPT-2 over DBpedia as classification task by making the response as integer labels. The classification is not meaningful to the model as the number of training set of each output label is much less than the number of total labels. Despite the setup, the model would not result in convergence in classification task. However, we use a trick to make the classification happen. We can use the probability map generated by next word prediction. As the vocabulary of our intended output is much smaller than the GPT-2's vocabulary, we get the most likely classification response by computing the probability of query combined with each response. In another word, we ask the model to evaluate the full sentence in the GPT-2's eval mode and plug in each possible response to form the sentence. So the most likely response should have the least loss with respect to the GPT-2's eval mode.

- The intermediate reward $\in [-1, 1]$, based on a threshold value with respect to the probability map of GPT-2. We want of relieve the constrain of getting maximal reward. It measures the closeness between output response and the true response while allows some degree of marginal error in GPT-2's consideration. We manually define the threshold value to be 4 after investigating the output loss from the GPT-2 eval mode and the logit length of it. Note that, the setting of threshold is only meaningful to GPT-2, since the logit length is simulating how much step of processing GPT-2 will take to arrive at the next token. To ensure the feasible computing, the range of output response in classification is randomly selected 100 response around the current true response.

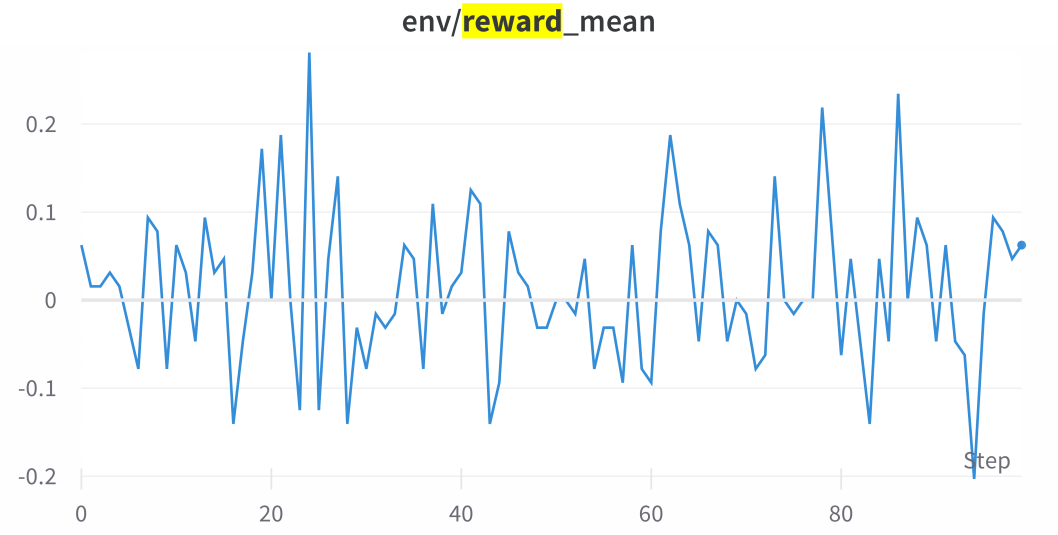


Figure 5.5: The Reward Mean

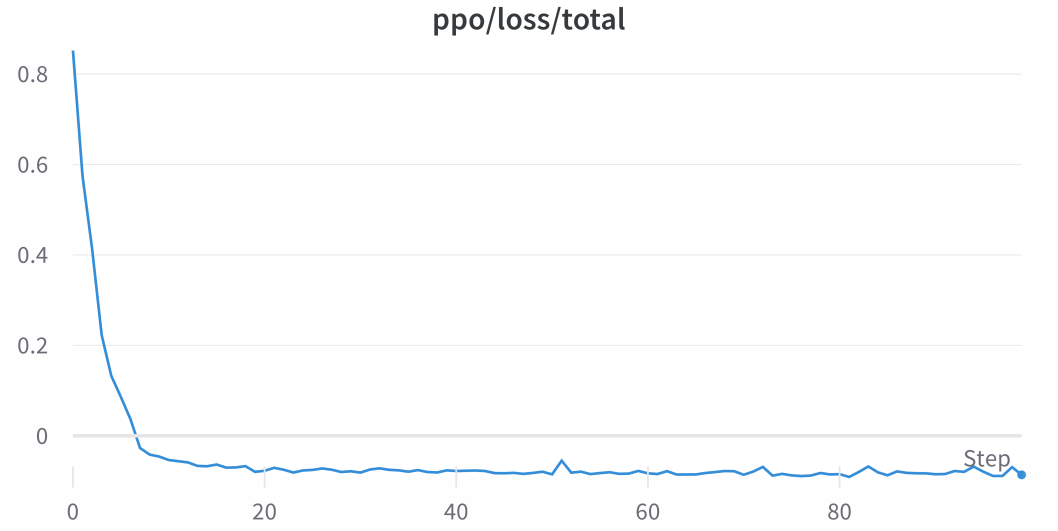


Figure 5.6: PPO Loss

From the loss plot and reward plot, we can see that the reward is idling around zero and shows no sign of convergence in a high reward area, although the loss from PPO is decreasing and seems to reach a flat ground.

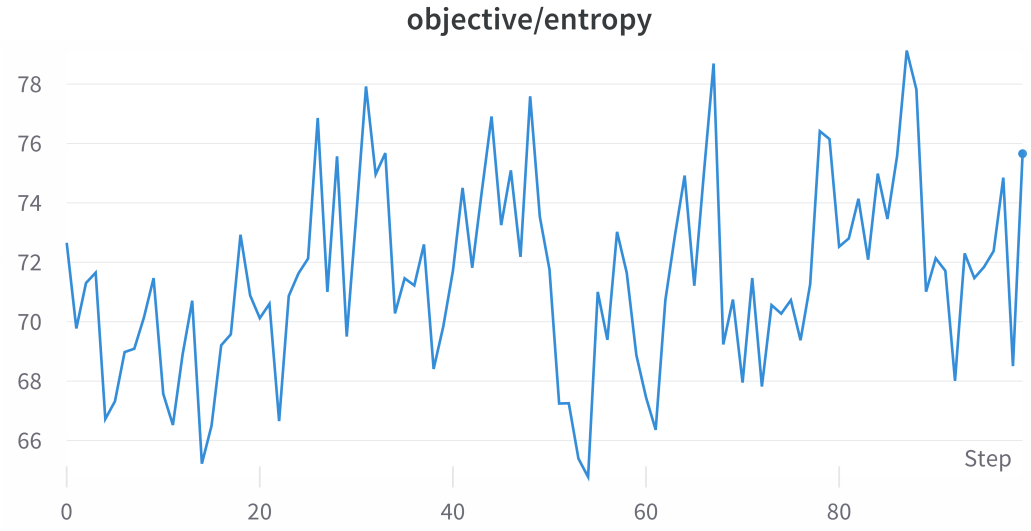
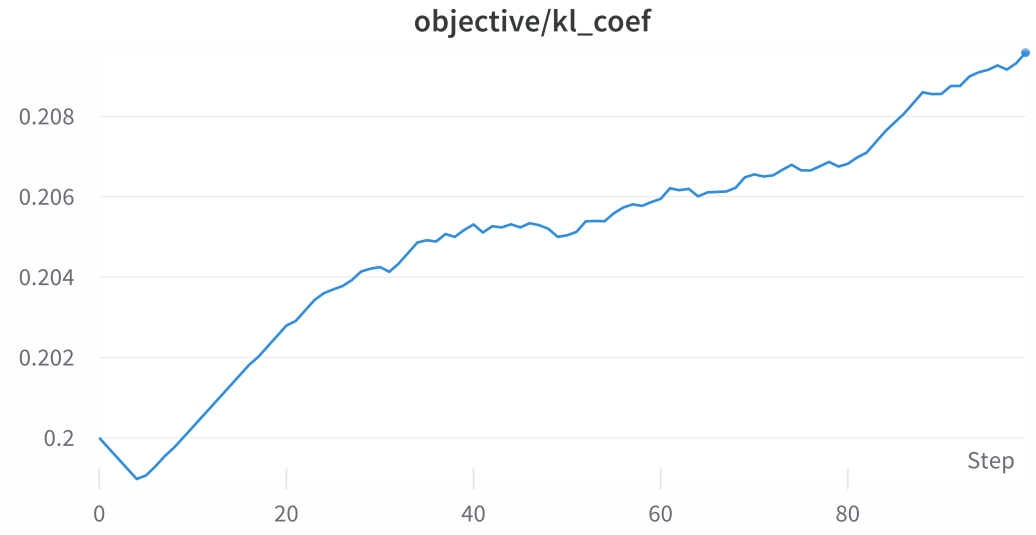
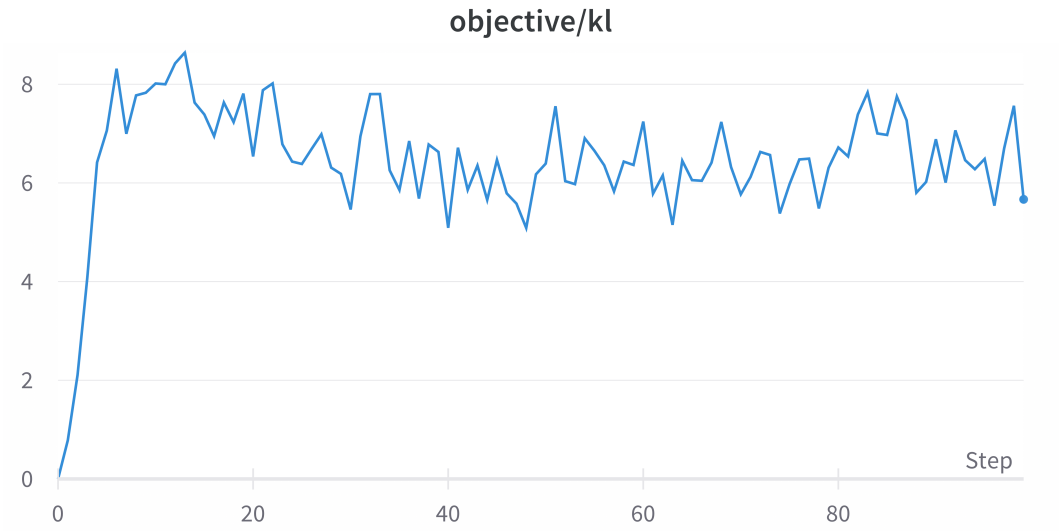


Figure 5.7: entropy



(a) KL coefficient



(b) KL objective

Figure 5.8: KL

We can also judge from the object KL and entropy that at this point the model has not reached at a convergence with the KL-divergence. One way to fix it to add more execution steps or start with a higher initial coefficient. The KL-coefficient has not reached the steady area and the reward is not increasing. We can judge that the setting for classification could be wrong and the factual sign of whether

the produced response is linked to the true response would give a zero reward most of the time.

5.3.2 Advanced Version

- To correct our baseline, we want to improve the threshold factual decision by fine-tuning the GPT-2 on the factual dataset constructed from DBpedia. Each triple in DBpedia forms a factual text and we use negative sampling to alternate triple with different subject or object. The resulting factual text and negative text comprise the factual dataset. We make sure that proportion of factual text is more than the negative text. We can see from above classification output from GPT-2 is troublesome, so we ask the GPT-2 to do batch size continuation in the vanilla way. And the factual decision is judged by the finetuned GPT-2 on the factual dataset, which produce either 1 or 0.

  We build a simple GPT-2 sequence classifier that has one forward linear layer after the generation of GPT-2 model. We restrict the length of input sequence to be less than 10, the batch size being 2, the hidden size being 768 and the learning rate is 1e−5. We trained the model with one epoch only. The evaluation accuracy on the split test dataset is reported to be around 0.8.

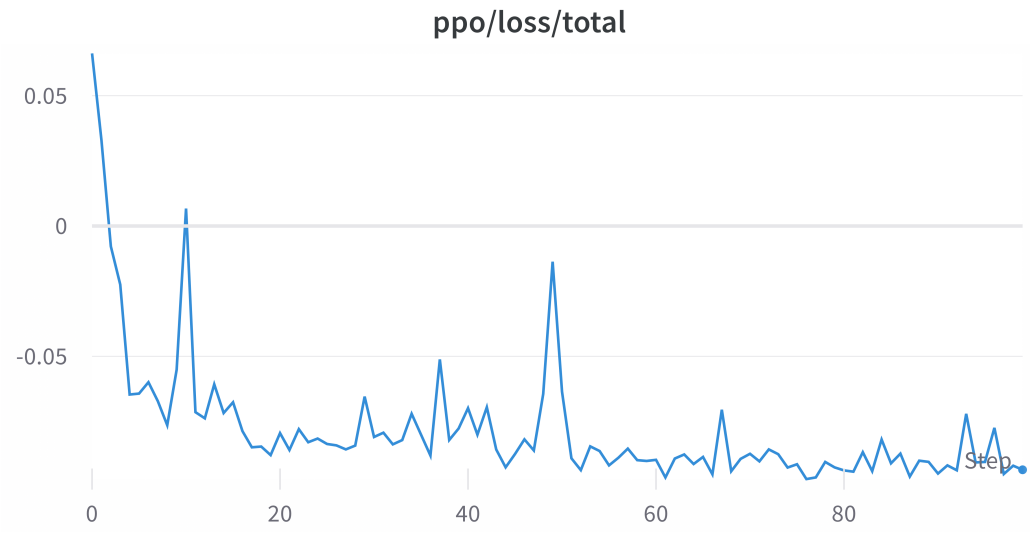
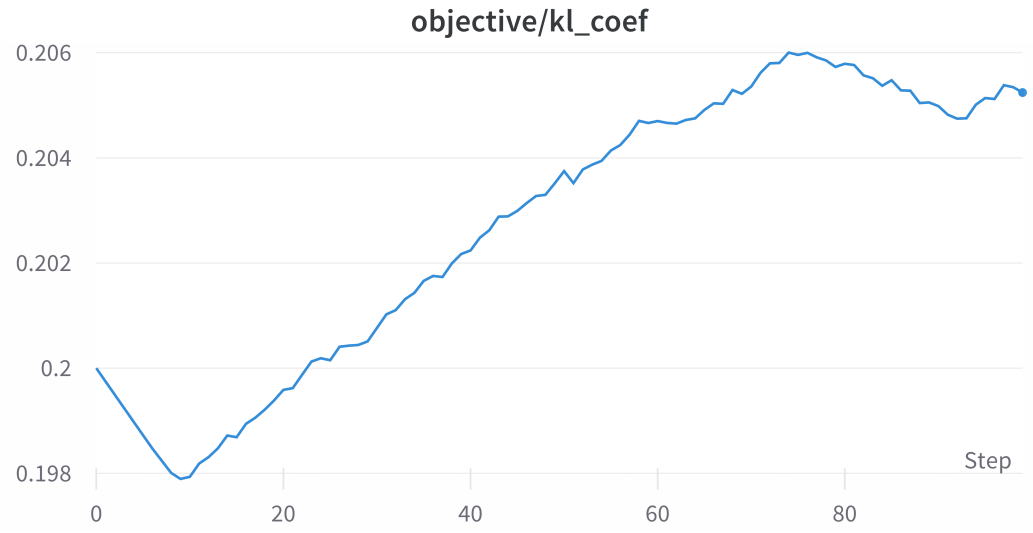
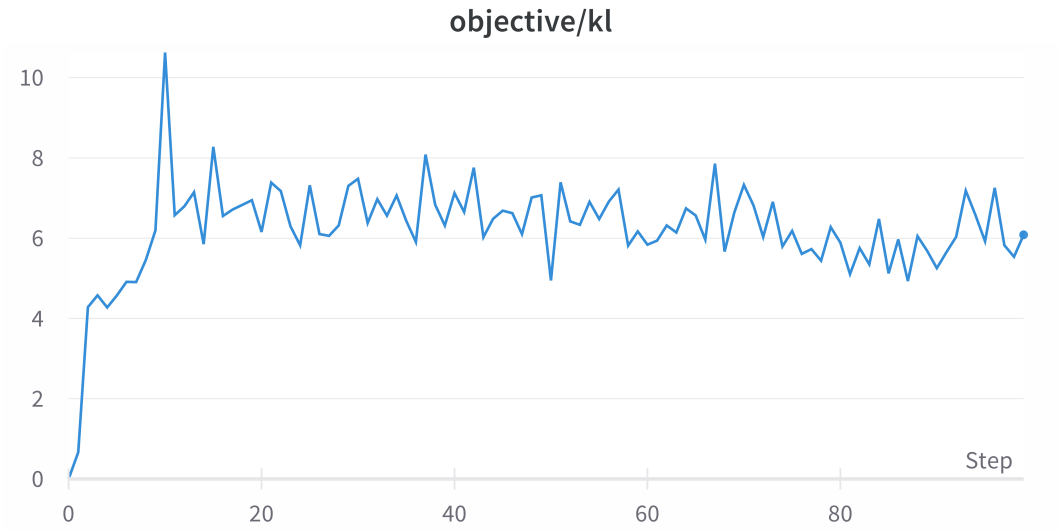


Figure 5.9: PPO Loss



(a) KL coefficient



(b) KL objective

Figure 5.10: KL

The issue with it is obvious that most of the time, the reward is minimum as the unconstrained batch response from GPT-2 will likely to be predicted of a factual value zero in view of fine-tuned GPT-2 model. The fine-tuned model has never seen the batch response style text from vanilla GPT-2 before. During training, it only learn how to recognize triple-version text. From the loss plot of PPO, the curve

is not smooth. If we investigate from the KL objective, the coefficient reached a peak, which does not seems to make a impact on the learning curve.

- Another version we experiment with is to make factual decision by entity linking model. Without using any prepossessing like stop word removal on triple, the reward is $10 * \frac{\text{number of linked words}}{\text{sequence length}}$. So the reward is scaled $\in [0, 10]$, which is expected to be around 5. We still use the GPT-2 with the batch continuation.

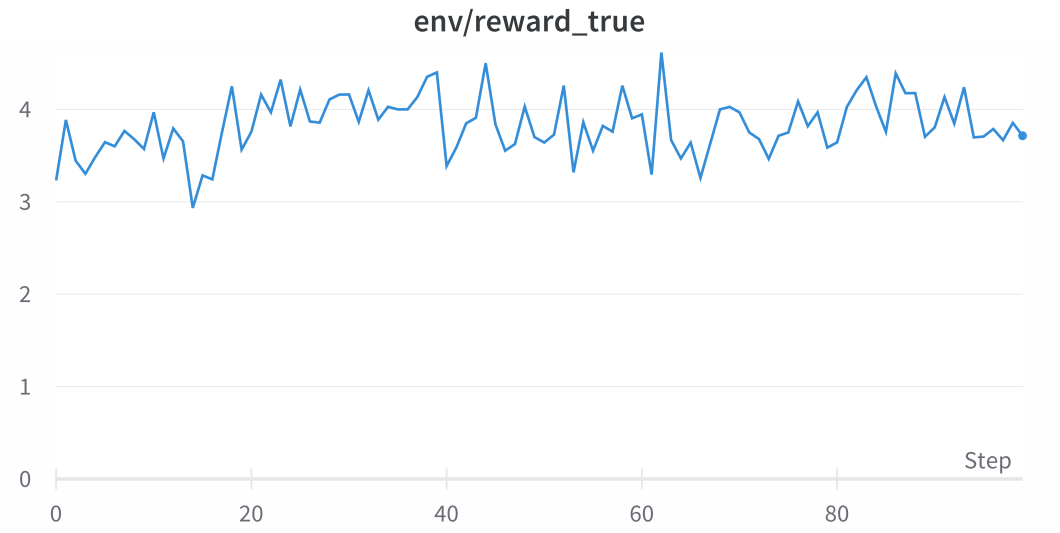


Figure 5.11: Linking Reward

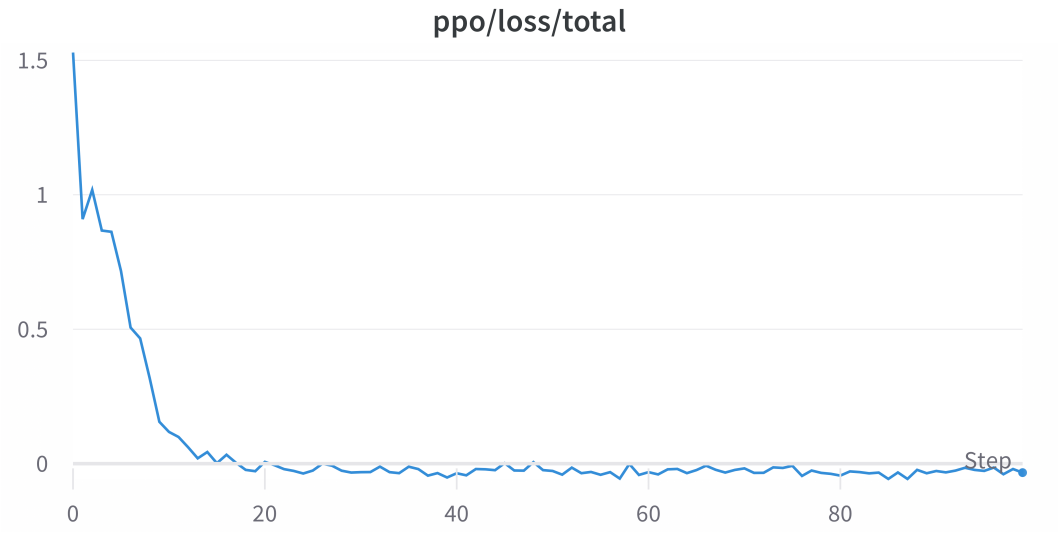
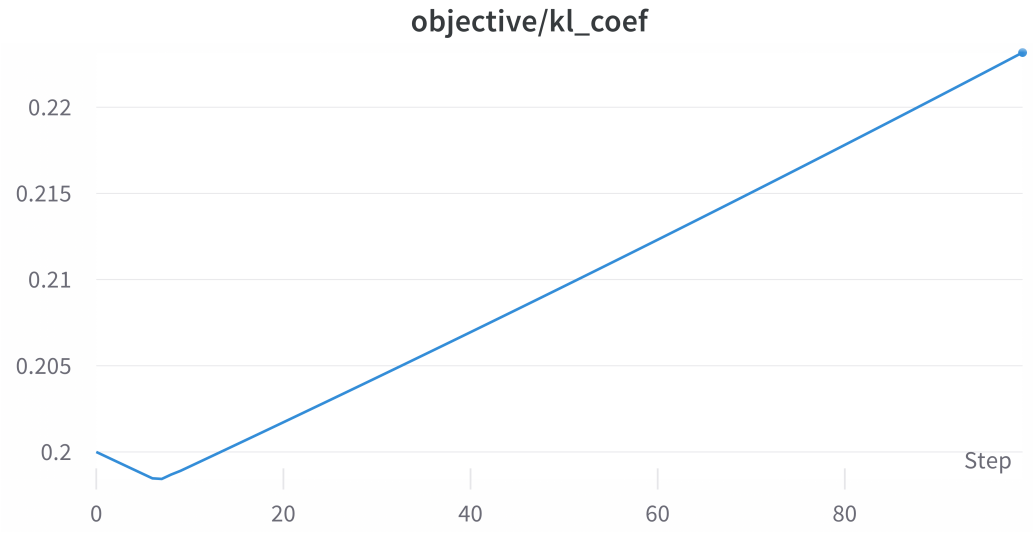
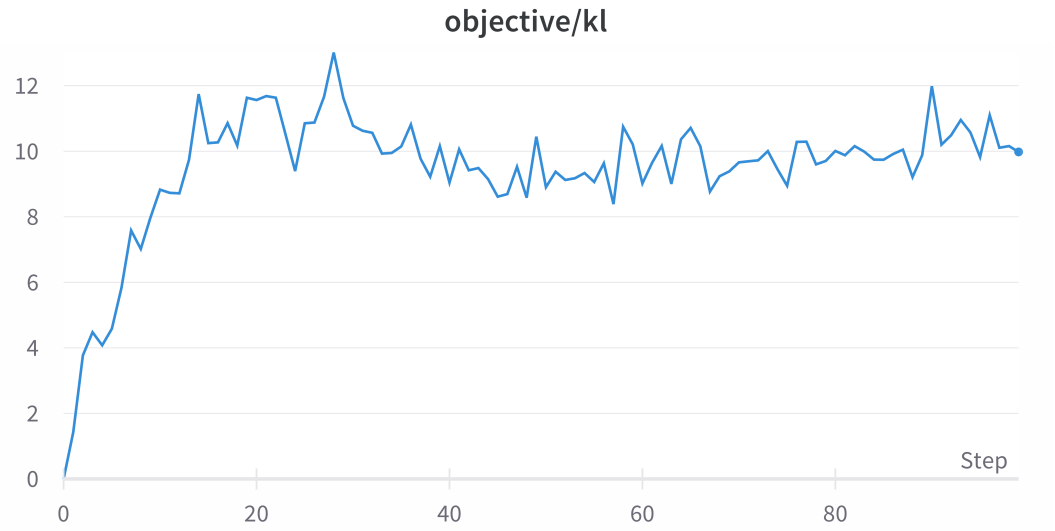


Figure 5.12: PPO Loss



(a) KL coefficient



(b) KL objective

Figure 5.13: KL

This is the best version of trl experiment. We can spot that the reward stays in a higher value area after a few steps and the loss for PPO seems to reach the convergence. But if we investigate from the KL objective, it seems not reaching

the target KL-divergence yet.

5.4 Summary

It is anticipated that fine-tuning the GPT-2 over the factual dataset would not work with good performance, even if we increase the layers and dataset size. If a simple fine-tuning is sufficient for language model to produce truthful output, we would not think of such a long-way detour to accomplish it. In the evaluation of trl framework, we can see that each version of factual reward and response has its own benefit but none of them is giving a satisfied result. The reward model in the linking version is effective to the policy gradient. The classification based response is not suitable for PPO to optimize as the classification we made upon GPT-2 output logit restrict the language space of it.

5.5 Reflection and Limitation

- For any research in this nature, it is a priority to construct an appropriate dataset. However, collecting a dataset of truthful sentence and making sentences recognizable to knowledge graph easily is extremely hard and time consuming. The sentences for one sample should be coherent of each other and should include occurrence of subjective tongue as unrecognizable neural examples. We also does not consider how to determine if a sentence is check worthy. The shortage of such a dataset limit how we did evaluation.

- The triple fact-checking modular has extremely high time complexity. Through experiment, the time cost for one triple input takes for 40 minutes on commercial-level cpu capacity, hence it does not support running for a large sample size.

- The experiment only take place on small proportion of DBpedia 2014 dump. With complete view of different KGs, some component implementation needs modification. Although both DBpedia and Wikidata have high connection to Wikipedia, other large KG like Yago is different to DBpedia. The evaluation does not consider the difference between various mainstream KGs, which is another on-going research topic Pillai et al. (2019). Although, we claimed that taking a detour in factual pipeline would reserve the potential on applying to other KGs and LMs. But, how PPO works in other LMs remains to be discovered. The evaluation does not consider the impact between different large components. For example, the NER components could neglect meaningful entities in the view of link component.

- The knowledge graph is incomplete to any sense. The assumption we have is that the information missing in KG is incorrect, which is untrue. Since any knowledge graph get update on a period of time, some information missing is not necessarily untrue. We also does not consider how to infer new facts from given facts in knowledge graph.

- Another limitation is that we consider every piece of text is in sentence level and assume no correlation between text. We failed to considering how long-term relation happen across sentence. The triple should be the basic units that assemble the sentence. But we doing the trivial way of detecting entity and comprise candidate triples by forming relations through iterating through all pairs of entities.

Concluding Remarks

6.1 Conclusion

Back to discussion of the related work, we have seen enormous approaches in the broad literature review, where one paper constructed the foundation for another paper, even if they are in different sub-domains of KG. The point for doing literature in such extensive details from broad to narrow aspect is to outline how different sub-domains intertwined together, which makes the importance of knowledge-aware task with NLP. If we compare our work to the KG-LM(Logan IV et al., 2019) published in ACL, we already make advancement in entity linking and fact-checking by proposing a comperhansive factual pipeline. Logan IV et al. (2019) adopted human hand-craft and heuristic based entity linking, and they also ignore the impact of relation extraction on the factual information.

This thesis aims to solve the fact-aware language modeling that provides grounds for constraining AI's action. We have to admit that it still has distance to a satisfied approach that the gap in the academia remains to be shorten so that this grand goal can be meet in the future. We provide a bird-view of how to assemble models published in top-tier paper into a powerful pipeline. We have shown from a base-level why the proposed model indeed work and when it would not work. Combining all those together and testing the pipeline if it can fit to the RL framework is rather challenging. We are confident that when the following future directions are solved, a combination of these research directions would comprise a satisfying factual pipeline.

6.2 Future Direction

We can now summarise all the future directions that still requires more attention.

- Aggregation
  It is worth exploring that using graph learning method like GNN, GCN to im-

prove the reasoning of KG. We have already seen from the related work, various approaches tend to keep a local structured graph for reasoning a input question. In my own experiment dealing with DBpedia 2014 dump, two nested loops is sufficient for computer to be stuck in execution. It is a complexity limitation to all KG approach that one should think carefully before construct any model. The common approach in the academia is to shrink the size around one millions of triples. There should be mainly two challenges. The message passing with edge features can only be encoded in a implicit way by GCN, which may not be sufficient for the multihop reasoning. It is complexity infeasible for graph learning method to be applied on a giant graph like a KG. We mentioned about relation aggregation as a limitation in the prior sections more than once. In order to achieve semantic reasonable relation aggregation, some mechanism of graph learning and modified knowledge representation methods have not yet been developed.

By the date of this thesis written, this issue have encountered its advancement in the submission for ICLR2023, which confirms our works aligned with the academia's direction of this field. We can only know these paper's abstract information, and they have not finished reviewing.

- The title is retrieve-and-read framework for knowledge graph reasoning, that aims to infer new fact from existing triples. The author claimed that reasoning with standard message-passing leads to the consequence of getting the representation to be over-smoothing. They proposed a way of using transformer-based GNN on a relevant sub-graph.
- The title is unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. They designed a semantic matching mechanism based on LMs that propagation through edge with respect to question relation. It seems much alike to QA-GNN(Yasunaga et al., 2021) that we have discussed.

- Reasoning in Fact-Checking
  We have seen in redundant details that how to corporate rule-based method into reasoning against facts. The triple fact-checking is the least-researched gap that should receive more attention. The existing methods, either rule-based or embedding-based, are far from satisfying. The embedding-based method is doing more like validating, not scoring on a factual triple. The rule-based method is very constrained by the incompleteness of the KG. When it comes to complex reasoning, the relational path and symbolic logic are promising directions. Zhang et al. (2020a) also pointed out the efficiency of applying probabilistic inference for uncertainty.
- Unified Relation in Knowledge Representation
  Knowledge representation learning models that have been developed are not unified. Some of them deals with the link prediction task in knowledge graph completion. However, framework for efficient relation extraction is less explored. It

seems that the aim of knowledge representation model is developed mostly for benchmark like FB15k, thus making them benchmark-oriented. When we are searching for unified framework for relation and triple representations so that we can conduct experiment on message passing, it is frustrated that we cannot find many combined-purpose papers except (Han et al., 2018a).

- Preserving Long-Term Relation
  We know that the past decade of NLP development centered with word embedding. The fact that the relation cannot exist without the starting and end entity. The aggregation on relations is a novel view when considering how to decompose text into meaningfully combination of triples. It means that some long-term relation is dominating other short-term trivial relations. We judge that the exist knowledge representation and word embedding method is not applicable to capture relations in a document-level.

Bibliography

Al-Moslmi, T.; Ocaña, M. G.; Opdahl, A. L.; and Veres, C., 2020. Named entity extraction for knowledge graphs: A literature overview. *IEEE Access*, 8 (2020), 32862–32881. [Cited on page 27.]

Atanasova, P., 2018. Llus marquez, alberto barrón-cedeno, tamer elsayed, reem suwaileh, wajdi zaghouani, spas kyuchukov, giovanni da san martino, and preslav nakov. 2018. overview of the clef-2018 checkthat! lab on automatic identification and verification of political claims, task 1: Check-worthiness. In *Working Notes of the Conference and Labs of the Evaluation Forum, CLEF*, vol. 18. [Cited on page 28.]

Baek, J.; Lee, D. B.; and Hwang, S. J., 2020. Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction. *Advances in Neural Information Processing Systems*, 33 (2020), 546–560. [Cited on page 24.]

Balazevic, I.; Allen, C.; and Hospedales, T., 2019. Multi-relational poincaré graph embeddings. *Advances in Neural Information Processing Systems*, 32 (2019). [Cited on page 21.]

Bauer, L.; Wang, Y.; and Bansal, M., 2018. Commonsense for generative multi-hop question answering tasks. *arXiv preprint arXiv:1809.06309*, (2018). [Cited on page 29.]

Bender, E. M.; Gebru, T.; McMillan-Major, A.; and Shmitchell, S., 2021. On the dangers of stochastic parrots: Can language models be too big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 610–623. [Cited on page 9.]

Bordes, A.; Glorot, X.; Weston, J.; and Bengio, Y., 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94, 2 (2014), 233–259. [Cited on page 22.]

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O., 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26 (2013). [Cited on pages 20, 21, 23, 40, and 41.]

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y., 2011. Learning structured embeddings of knowledge bases. In *Twenty-fifth AAAI conference on artificial intelligence*. [Cited on page 22.]

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al., 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33 (2020), 1877–1901. [Cited on page 9.]

Cai, R.; Zhang, X.; and Wang, H., 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 756–765. [Cited on page 24.]

Chami, I.; Wolf, A.; Juan, D.-C.; Sala, F.; Ravi, S.; and Ré, C., 2020. Low-dimensional hyperbolic knowledge graph embeddings. *arXiv preprint arXiv:2005.00545*, (2020). [Cited on page 21.]

Chekol, M.; Pirrò, G.; Schoenfisch, J.; and Stuckenschmidt, H., 2017. Marrying uncertainty and time in knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31. [Cited on page 24.]

Chen, Y.; Wu, L.; and Zaki, M. J., 2019. Bidirectional attentive memory networks for question answering over knowledge bases. *arXiv preprint arXiv:1903.02188*, (2019). [Cited on page 29.]

Ciampaglia, G. L.; Shiralkar, P.; Rocha, L. M.; Bollen, J.; Menczer, F.; and Flammini, A., 2015. Computational fact checking from knowledge networks. *PloS one*, 10, 6 (2015), e0128193. [Cited on page 28.]

Dai, Z.; Li, L.; and Xu, W., 2016. Cfo: Conditional focused neural question answering with large-scale knowledge bases. *arXiv preprint arXiv:1606.01994*, (2016). [Cited on page 29.]

Das, R.; Dhuliawala, S.; Zaheer, M.; Vilnis, L.; Durugkar, I.; Krishnamurthy, A.; Smola, A.; and McCallum, A., 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*, (2017). [Cited on page 23.]

Das, R.; Neelakantan, A.; Belanger, D.; and McCallum, A., 2016. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*, (2016). [Cited on page 23.]

Dasgupta, S. S.; Ray, S. N.; and Talukdar, P., 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2001–2011. [Cited on page 24.]

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S., 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32. [Cited on page 22.]

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, (2018). [Cited on pages 4, 5, and 6.]

Ding, N.; Hu, S.; Zhao, W.; Chen, Y.; Liu, Z.; Zheng, H.-T.; and Sun, M., 2021. Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*, (2021). [Cited on page 10.]

Feng, J.; Huang, M.; Zhao, L.; Yang, Y.; and Zhu, X., 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of the aaai conference on artificial intelligence*, vol. 32. [Cited on page 24.]

Ganea, O.-E. and Hofmann, T., 2017. Deep joint entity disambiguation with local neural attention. *arXiv preprint arXiv:1704.04920*, (2017). [Cited on pages 27, 54, 63, and 69.]

Gao, T.; Han, X.; Zhu, H.; Liu, Z.; Li, P.; Sun, M.; and Zhou, J., 2019. Fewrel 2.0: Towards more challenging few-shot relation classification. *arXiv preprint arXiv:1910.07124*, (2019). [Cited on page 41.]

Gardner, M.; Talukdar, P.; Krishnamurthy, J.; and Mitchell, T., 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 397–406. [Cited on pages 22 and 23.]

Goel, R.; Kazemi, S. M.; Brubaker, M.; and Poupart, P., 2020. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 3988–3995. [Cited on page 24.]

Guo, S.; Wang, Q.; Wang, B.; Wang, L.; and Guo, L., 2015. Semantically smooth knowledge graph embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 84–94. [Cited on page 22.]

Guo, S.; Wang, Q.; Wang, L.; Wang, B.; and Guo, L., 2018. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32. [Cited on page 23.]

Guo, Z.; Zhang, Y.; and Lu, W., 2019. Attention guided graph convolutional networks for relation extraction. *arXiv preprint arXiv:1906.07510*, (2019). [Cited on page 24.]

Gupta, N.; Singh, S.; and Roth, D., 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2681–2690. [Cited on page 39.]

Gupta, P.; Wu, C.-S.; Liu, W.; and Xiong, C., 2021. Dialfact: A benchmark for fact-checking in dialogue. *arXiv preprint arXiv:2110.08222*, (2021). [Cited on page 68.]

Habib, M. B. and Van Keulen, M., 2016. Twitterneed: A hybrid approach for named entity extraction and disambiguation for tweet. *Natural language engineering*, 22, 3 (2016), 423–456. [Cited on page 26.]

Han, X.; Gao, T.; Yao, Y.; Ye, D.; Liu, Z.; and Sun, M., 2019. Opennre: An open and extensible toolkit for neural relation extraction. *arXiv preprint arXiv:1909.13078*, (2019). [Cited on page 69.]

Han, X.; Liu, Z.; and Sun, M., 2018a. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32. [Cited on page 85.]

Han, X.; Yu, P.; Liu, Z.; Sun, M.; and Li, P., 2018b. Hierarchical relation extraction with coarse-to-fine grained attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2236–2245. [Cited on page 24.]

Han, X.; Zhao, W.; Ding, N.; Liu, Z.; and Sun, M., 2021. Ptr: Prompt tuning with rules for text classification. *arXiv preprint arXiv:2105.11259*, (2021). [Cited on page 10.]

He, S.; Liu, K.; Ji, G.; and Zhao, J., 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, 623–632. [Cited on page 21.]

He, Z.; Zhong, J.; Wang, C.; and Hu, C., 2020. Collective entity disambiguation based on deep semantic neighbors and heterogeneous entity correlation. In *CCF International Conference on Natural Language Processing and Chinese Computing*, 193–205. Springer. [Cited on page 27.]

Jenatton, R.; Roux, N.; Bordes, A.; and Obozinski, G. R., 2012. A latent factor model for highly multi-relational data. *Advances in neural information processing systems*, 25 (2012). [Cited on page 22.]

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J., 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, 687–696. [Cited on page 21.]

Ji, G.; Liu, K.; He, S.; and Zhao, J., 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 31. [Cited on page 24.]

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Philip, S. Y., 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33, 2 (2021), 494–514. [Cited on page 19.]

Jiang, T.; Liu, T.; Ge, T.; Sha, L.; Li, S.; Chang, B.; and Sui, Z., 2016a. Encoding temporal information for time-aware link prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2350–2354. [Cited on page 24.]

Jiang, X.; Wang, Q.; Li, P.; and Wang, B., 2016b. Relation extraction with multi-instance multi-label convolutional neural networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 1471–1480. [Cited on page 24.]

Jiang, Y.; Bordia, S.; Zhong, Z.; Dognin, C.; Singh, M.; and Bansal, M., 2020. Hover: A dataset for many-hop fact extraction and claim verification. *arXiv preprint arXiv:2011.03088*, (2020). [Cited on page 68.]

Kim, J. and Choi, K.-S., 2020. Unsupervised fact checking by counter-weighted positive and negative evidential paths in a knowledge graph. In *Proceedings of the 28th International Conference on Computational Linguistics*, 1677–1686. [Cited on page 28.]

Kim, J.-S. and Choi, K.-S., 2021. Fact checking in knowledge graphs by logical consistency. (2021). [Cited on pages 64 and 65.]

Kipf, T. N. and Welling, M., 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, (2016). [Cited on page 22.]

Kolitsas, N.; Ganea, O.-E.; and Hofmann, T., 2018. End-to-end neural entity linking. *arXiv preprint arXiv:1808.07699*, (2018). [Cited on page 27.]

Lao, N. and Cohen, W. W., 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81, 1 (2010), 53–67. [Cited on page 23.]

Le, P. and Titov, I., 2018. Improving entity linking by modeling latent relations between mentions. *arXiv preprint arXiv:1804.10637*, (2018). [Cited on pages 27, 54, 63, and 69.]

Leblay, J. and Chekol, M. W., 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, 1771–1776. [Cited on page 24.]

Lee, N.; Li, B. Z.; Wang, S.; Yih, W.-t.; Ma, H.; and Khabsa, M., 2020. Language models as fact checkers? *arXiv preprint arXiv:2006.04102*, (2020). [Cited on page 29.]

Lester, B.; Al-Rfou, R.; and Constant, N., 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, (2021). [Cited on page 10.]

Li, M.; Xing, Y.; Kong, F.; and Zhou, G., 2022. Towards better entity linking. *Frontiers of Computer Science*, 16, 2 (2022), 1–13. [Cited on page 54.]

Lin, B. Y.; Chen, X.; Chen, J.; and Ren, X., 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint arXiv:1909.02151*, (2019). [Cited on page 29.]

Lin, X. V.; Socher, R.; and Xiong, C., 2018. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*, (2018). [Cited on page 23.]

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X., 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*. [Cited on pages 20 and 23.]

Lin, Y.; Shen, S.; Liu, Z.; Luan, H.; and Sun, M., 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2124–2133. [Cited on page 24.]

Liu, H.; Wu, Y.; and Yang, Y., 2017. Analogical inference for multi-relational embeddings. In *International conference on machine learning*, 2168–2178. PMLR. [Cited on pages 20 and 22.]

Liu, H.; Zhou, S.; Chen, C.; Gao, T.; Xu, J.; and Shu, M., 2022. Dynamic knowledge graph reasoning based on deep reinforcement learning. *Knowledge-Based Systems*, 241 (2022), 108235. [Cited on page 23.]

Liu, Q.; Jiang, H.; Evdokimov, A.; Ling, Z.-H.; Zhu, X.; Wei, S.; and Hu, Y., 2016. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*, (2016). [Cited on page 22.]

Liu, W.; Zhou, P.; Zhao, Z.; Wang, Z.; Ju, Q.; Deng, H.; and Wang, P., 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2901–2908. [Cited on pages 29 and 40.]

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, (2019). [Cited on page 4.]

Logan IV, R. L.; Liu, N. F.; Peters, M. E.; Gardner, M.; and Singh, S., 2019. Barack's wife hillary: Using knowledge-graphs for fact-aware language modeling. *arXiv preprint arXiv:1906.07241*, (2019). [Cited on pages 29, 39, 41, and 83.]

Luo, G.; Huang, X.; Lin, C.-Y.; and Nie, Z., 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 879–888. [Cited on page 26.]

Lv, X.; Gu, Y.; Han, X.; Hou, L.; Li, J.; and Liu, Z., 2019. Adapting meta knowledge graph information for multi-hop reasoning over few-shot relations. *arXiv preprint arXiv:1908.11513*, (2019). [Cited on page 24.]

Ma, Y.; Tresp, V.; and Daxberger, E. A., 2019. Embedding models for episodic knowledge graphs. *Journal of Web Semantics*, 59 (2019), 100490. [Cited on page 24.]

Martins, P. H.; Marinho, Z.; and Martins, A. F., 2019. Joint learning of named entity recognition and entity linking. *arXiv preprint arXiv:1907.08243*, (2019). [Cited on page 27.]

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J., 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, (2013). [Cited on page 6.]

Mikolov, T.; Karafiát, M.; Burget, L.; Cernockỳ, J.; and Khudanpur, S., 2010. Recurrent neural network based language model. In *Interspeech*, vol. 2, 1045–1048. Makuhari. [Cited on page 40.]

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J., 2013b. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26 (2013). [Cited on page 41.]

Miwa, M. and Bansal, M., 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*, (2016). [Cited on page 24.]

Mohammed, S.; Shi, P.; and Lin, J., 2017. Strong baselines for simple question answering over knowledge graphs with and without neural networks. *arXiv preprint arXiv:1712.01969*, (2017). [Cited on page 29.]

Moro, A.; Raganato, A.; and Navigli, R., 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2 (2014), 231–244. [Cited on page 27.]

Nathani, D.; Chauhan, J.; Sharma, C.; and Kaul, M., 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. *arXiv preprint arXiv:1906.01195*, (2019). [Cited on page 22.]

Neelakantan, A.; Roth, B.; and McCallum, A., 2015. Compositional vector space models for knowledge base completion. *arXiv preprint arXiv:1504.06662*, (2015). [Cited on page 22.]

Nguyen, D. B.; Theobald, M.; and Weikum, G., 2016. J-nerd: joint named entity recognition and disambiguation with rich linguistic features. *Transactions of the Association for Computational Linguistics*, 4 (2016), 215–229. [Cited on page 26.]

Nguyen, T. H. and Grishman, R., 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st workshop on vector space modeling for natural language processing*, 39–48. [Cited on page 24.]

Nickel, M.; Rosasco, L.; and Poggio, T., 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30. [Cited on pages 20 and 22.]

Nickel, M.; Tresp, V.; and Kriegel, H.-P., 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, 271–280. [Cited on page 22.]

Nie, F.; Cao, Y.; Wang, J.; Lin, C.-Y.; and Pan, R., 2018. Mention and entity description co-attention for entity disambiguation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32. [Cited on page 26.]

Ortona, S.; Meduri, V. V.; and Papotti, P., 2018. Robust discovery of positive and negative rules in knowledge bases. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 1168–1179. IEEE. [Cited on pages 64 and 65.]

Pillai, S. G.; Soon, L.-K.; and Haw, S.-C., 2019. Comparing dbpedia, wikidata, and yago for web information retrieval. In *Intelligent and Interactive Computing*, 525–535. Springer. [Cited on page 80.]

Qin, P.; Wang, X.; Chen, W.; Zhang, C.; Xu, W.; and Wang, W. Y., 2020. Generative adversarial zero-shot relational learning for knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 8673–8680. [Cited on page 24.]

Qin, P.; Xu, W.; and Wang, W. Y., 2018a. Dsgan: Generative adversarial training for distant supervision relation extraction. *arXiv preprint arXiv:1805.09929*, (2018). [Cited on page 24.]

Qin, P.; Xu, W.; and Wang, W. Y., 2018b. Robust distant supervision relation extraction via deep reinforcement learning. *arXiv preprint arXiv:1805.09927*, (2018). [Cited on page 24.]

Qu, M. and Tang, J., 2019. Probabilistic logic neural networks for reasoning. *Advances in neural information processing systems*, 32 (2019). [Cited on page 23.]

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al., 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1, 8 (2019), 9. [Cited on page 4.]

Sarzynska-Wawer, J.; Wawer, A.; Pawlak, A.; Szymanowska, J.; Stefaniak, I.; Jarkiewicz, M.; and Okruszek, L., 2021. Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Research*, 304 (2021), 114135. [Cited on page 4.]

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Berg, R. v. d.; Titov, I.; and Welling, M., 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, 593–607. Springer. [Cited on page 22.]

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P., 2015. Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR. [Cited on page 51.]

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O., 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, (2017). [Cited on pages 35, 52, and 53.]

Sevgili, Ö.; Panchenko, A.; and Biemann, C., 2019. Improving neural entity disambiguation with graph embeddings. In *Proceedings of the 57th annual meeting of the association for computational linguistics: student research workshop*, 315–322. [Cited on page 26.]

Shen, T.; Mao, Y.; He, P.; Long, G.; Trischler, A.; and Chen, W., 2020. Exploiting structured knowledge in text via graph-guided representation learning. *arXiv preprint arXiv:2004.14224*, (2020). [Cited on page 29.]

Shen, Y. and Huang, X.-J., 2016. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2526–2536. [Cited on page 24.]

Shi, B. and Weninger, T., 2016. Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-based systems*, 104 (2016), 123–133. [Cited on page 28.]

Shiralkar, P.; Flammini, A.; Menczer, F.; and Ciampaglia, G. L., 2017. Finding streams in knowledge graphs to support fact checking. In *2017 IEEE International Conference on Data Mining (ICDM)*, 859–864. IEEE. [Cited on page 28.]

Shu, K.; Mahudeswaran, D.; Wang, S.; and Liu, H., 2020. Hierarchical propagation networks for fake news detection: Investigation and exploitation. In *Proceedings of the international AAAI conference on web and social media*, vol. 14, 626–637. [Cited on page 27.]

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A., 2013. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26 (2013). [Cited on pages 20 and 22.]

Spitkovsky, V. I. and Chang, A. X., 2012. A cross-lingual dictionary for english wikipedia concepts. (2012). [Cited on page 69.]

Stephen, M.; Caiming, X.; James, B.; and Socher, R., 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. [Cited on page 39.]

Stiennon, N.; Ouyang, L.; Wu, J.; Ziegler, D.; Lowe, R.; Voss, C.; Radford, A.; Amodei, D.; and Christiano, P. F., 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33 (2020), 3008–3021. [Cited on pages 34, 35, 37, 38, and 39.]

Sun, Y.; Lin, L.; Tang, D.; Yang, N.; Ji, Z.; and Wang, X., 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Twenty-fourth international joint conference on artificial intelligence*. [Cited on page 26.]

Sun, Y.; Wang, S.; Li, Y.; Feng, S.; Tian, H.; Wu, H.; and Wang, H., 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 8968–8975. [Cited on page 29.]

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J., 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, (2019). [Cited on pages 20 and 40.]

Trivedi, R.; Dai, H.; Wang, Y.; and Song, L., 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*, 3462–3471. PMLR. [Cited on page 24.]

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G., 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, 2071–2080. PMLR. [Cited on pages 20 and 22.]

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I., 2017. Attention is all you need. *Advances in neural information processing systems*, 30 (2017). [Cited on pages 4, 30, and 31.]

Wang, Q.; Huang, P.; Wang, H.; Dai, S.; Jiang, W.; Liu, J.; Lyu, Y.; Zhu, Y.; and Wu, H., 2019. Coke: Contextualized knowledge graph embedding. *arXiv preprint arXiv:1911.02168*, (2019). [Cited on page 29.]

Wang, X.; Gao, T.; Zhu, Z.; Zhang, Z.; Liu, Z.; Li, J.; and Tang, J., 2021a. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9 (2021), 176–194. [Cited on pages 29, 40, and 41.]

Wang, X.; Liu, Q.; Gui, T.; Zhang, Q.; Zou, Y.; Zhou, X.; Ye, J.; Zhang, Y.; Zheng, R.; Pang, Z.; et al., 2021b. Textflint: Unified multilingual robustness evaluation toolkit for natural language processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International*

Joint Conference on Natural Language Processing: System Demonstrations, 347–355. [Cited on page 10.]

Wang, Y.; Gemulla, R.; and Li, H., 2018. On multi-relational link prediction with bilinear models. In *Thirty-Second AAAI Conference on Artificial Intelligence*. [Cited on page 22.]

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z., 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 28. [Cited on pages 20, 22, and 23.]

Wenhu Chen, J. C. Y. Z. H. W. S. L. X. Z., Hongmin Wang and Wang, W. Y., 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*. Addis Ababa, Ethiopia. [Cited on page 68.]

Xiao, H.; Huang, M.; Hao, Y.; and Zhu, X., 2015a. From one point to a manifold: Orbit models for knowledge graph embedding. (2015). [Cited on page 21.]

Xiao, H.; Huang, M.; Hao, Y.; and Zhu, X., 2015b. Transg: A generative mixture model for knowledge graph embedding. *arXiv preprint arXiv:1509.05488*, (2015). [Cited on page 21.]

Xiao, H.; Huang, M.; Meng, L.; and Zhu, X., 2017. Ssp: semantic space projection for knowledge graph embedding with text descriptions. In *Thirty-First AAAI conference on artificial intelligence*. [Cited on page 22.]

Xie, Q.; Ma, X.; Dai, Z.; and Hovy, E., 2017. An interpretable knowledge transfer model for knowledge base completion. *arXiv preprint arXiv:1704.05908*, (2017). [Cited on page 21.]

Xie, R.; Liu, Z.; Sun, M.; et al., 2016. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*, vol. 2016, 2965–2971. [Cited on pages 22 and 40.]

Xiong, W.; Hoang, T.; and Wang, W. Y., 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*, (2017). [Cited on page 23.]

Xu, Y.; Mou, L.; Li, G.; Chen, Y.; Peng, H.; and Jin, Z., 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, 1785–1794. [Cited on page 24.]

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L., 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, (2014). [Cited on page 22.]

Yao, L.; Mao, C.; and Luo, Y., 2019a. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*, (2019). [Cited on page 29.]

Yao, Y.; Ye, D.; Li, P.; Han, X.; Lin, Y.; Liu, Z.; Liu, Z.; Huang, L.; Zhou, J.; and Sun, M., 2019b. Docred: A large-scale document-level relation extraction dataset. *arXiv preprint arXiv:1906.06127*, (2019). [Cited on page 4.]

Yasunaga, M.; Ren, H.; Bosselut, A.; Liang, P.; and Leskovec, J., 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*, (2021). [Cited on pages 1 and 84.]

Yu, Y.; He, K.; and Li, J., 2021. Adversarial training for supervised relation extraction. *Tsinghua Science and Technology*, 27, 3 (2021), 610–618. [Cited on page 24.]

Zeng, D.; Liu, K.; Chen, Y.; and Zhao, J., 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, 1753–1762. [Cited on page 24.]

Zeng, D.; Liu, K.; Lai, S.; Zhou, G.; and Zhao, J., 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers*, 2335–2344. [Cited on page 24.]

Zeng, X.; He, S.; Liu, K.; and Zhao, J., 2018. Large scaled relation extraction with reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*. [Cited on page 24.]

Zhang, N.; Deng, S.; Sun, Z.; Wang, G.; Chen, X.; Zhang, W.; and Chen, H., 2019a. Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. *arXiv preprint arXiv:1903.01306*, (2019). [Cited on page 24.]

Zhang, S.; Tay, Y.; Yao, L.; and Liu, Q., 2019b. Quaternion knowledge graph embeddings. *Advances in neural information processing systems*, 32 (2019). [Cited on page 20.]

Zhang, W.; Paudel, B.; Wang, L.; Chen, J.; Zhu, H.; Zhang, W.; Bernstein, A.; and Chen, H., 2019c. Iteratively learning embeddings and rules for knowledge graph reasoning. In *The World Wide Web Conference*, 2366–2377. [Cited on page 23.]

Zhang, Y.; Chen, X.; Yang, Y.; Ramamurthy, A.; Li, B.; Qi, Y.; and Song, L., 2020a. Efficient probabilistic logic reasoning with graph neural networks. *arXiv preprint arXiv:2001.11850*, (2020). [Cited on pages 23 and 84.]

Zhang, Y.; Dai, H.; Kozareva, Z.; Smola, A. J.; and Song, L., 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-second AAAI conference on artificial intelligence*. [Cited on page 29.]

Zhang, Y.; Zhong, V.; Chen, D.; Angeli, G.; and Manning, C. D., 2017. Position-aware attention and supervised data improve slot filling. In *Conference on Empirical Methods in Natural Language Processing*. [Cited on page 41.]

Zhang, Z.; Cai, J.; Zhang, Y.; and Wang, J., 2020b. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 3065–3072. [Cited on page 20.]

Zhang, Z.; Han, X.; Liu, Z.; Jiang, X.; Sun, M.; and Liu, Q., 2019d. Ernie: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*, (2019). [Cited on pages 29 and 40.]

Zhou, P.; Shi, W.; Tian, J.; Qi, Z.; Li, B.; Hao, H.; and Xu, B., 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)*, 207–212. [Cited on page 24.]

Zhou, X. and Zafarani, R., 2020. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Computing Surveys (CSUR)*, 53, 5 (2020), 1–40. [Cited on page 27.]

Zhu, G. and Iglesias, C. A., 2018. Exploiting semantic similarity for named entity disambiguation in knowledge graphs. *Expert Systems with Applications*, 101 (2018), 8–24. [Cited on page 26.]

Ziegler, D. M.; Stiennon, N.; Wu, J.; Brown, T. B.; Radford, A.; Amodei, D.; Christiano, P.; and Irving, G., 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, (2019). [Cited on page 34.]

Zuo, H.; Yin, Y.; and Childs, P., 2021. Patent-kg: Patent knowledge graph use for engineering design. *arXiv preprint arXiv:2108.11899*, (2021). [Cited on page 19.]