Web Data Processing System Course Report

Course Code: XM_40020
Academic Year: 2023

Group 30

Name	Student ID	Email
Leonardo Dominici	2829197	l.dominici@student.vu.nl
Omid Afroozeh	2802676	o.afroozeh@student.vu.nl
Aynaz Abdollahzadeh	2800921	a.abdollahzadeh@student.vu.nl
Daniele Quartinieri	2767570	d.quartinieri@student.vu.nl

Project Resources

- Project GitHub repository: https://github.com/Leodom01/XM_40020-WedDataProcessingSystems
- DockerHub container image: `leodom01/webdataprocessingsystems_factchecker`

1 Introduction

In this project, our goal is to post-process the outputs of a Large Language Model (LLM) using NLP tools. We aim to pose questions to the LLM, then post-process its outputs by extracting answers and validating their correctness against a Knowledge Graph. For this task, we've selected Meta's LLaMA 2, a model with 7 billion parameters, and Wikidata as our knowledge graph of choice. Please refer to Figure 1 for an illustration of our pipeline.

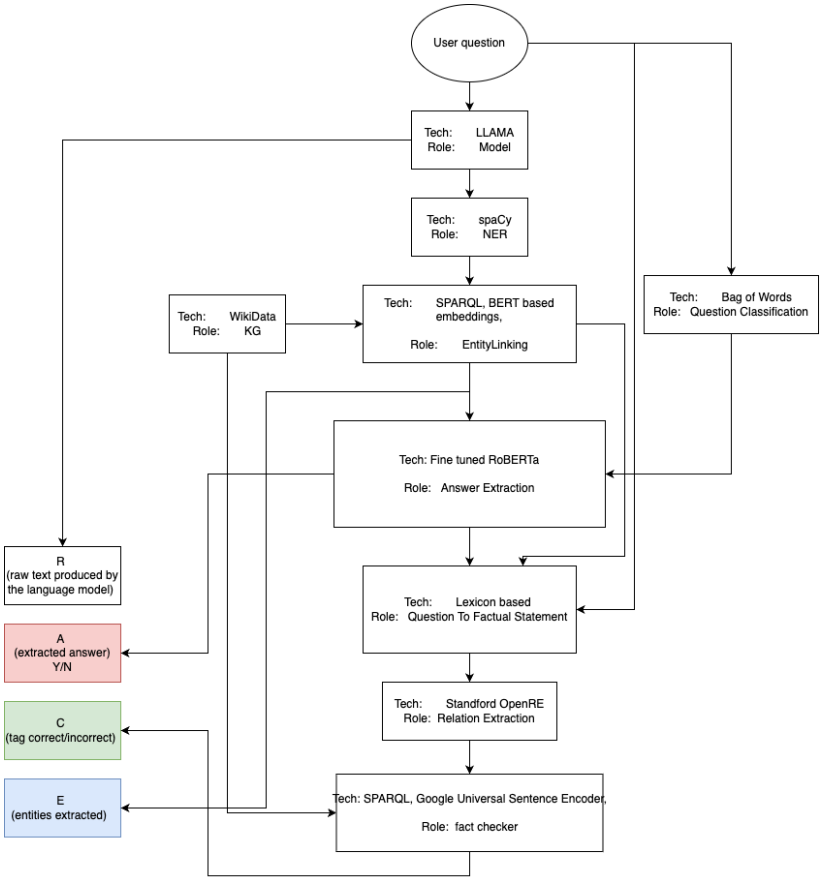

Figure 1: Fact checking project pipeline

2 Named Entity Recognition

For the named entity recognition we are using SpaCy. Even if SpaCy contains a complete set of NLP tools, in this case, we are only using the NER. The SpaCy NER pipeline, `en_core_web_sm`, uses token vectorization, POS tagging, dependency parser, handwritten rules, and lemmatizer. Given the great quality of the NER preprocessing we've noticed that our own preprocessing was not necessary better nor helpful and often led to worse performance.

3 Named Entity Linking

In the task of entity linking, we follow a three-phase approach.

Phase 1: Candidate Generation - For this, we utilize a SPARQL endpoint to query Wikidata. This query searches for entities based on a specified term, fetching their labels, descriptions, and the count of their associated Wikipedia

links. The results are then grouped and ordered by the number of links to prioritize more prominent entities.
Phase 2: Candidate Ranking - In this phase, we adopt a vector space model approach. The process involves embedding both the context in which the entity is mentioned and the descriptions of entities from Wikidata using a sentence transformer. These embeddings are then represented as vectors. We compare these vectors using cosine similarity, a straightforward yet effective method, to determine similarity. The entity with the most similar embedding vector to the context is selected as the best candidate.
Phase 3: Unlinkable Mention Prediction - To accomplish this, we first prune our list of candidate entities, removing any entries that lack a description or have too few citation links. If this pruning process results in an empty candidate list, we then classify the mention as unlinkable.

4 Answer Extraction

For answer extraction we've decided to leverage the accuracy of RoBERTa. the goal of this part of the pipeline is to distill the answer provided by the LLM which could be a boolean answer (true or false) or an entity answer. This distinction has been provided by the assignment description. RoBERTa is a LLM build with the same identical BERT architecture but with varying key hyperparameters, training data size and learning rate (the paper goal was to show that BERT could have been trained in a better manner). In our specific case, RoBERTa has been fine tuned for two separate, slightly different tasks: boolean and entity question answering. In the boolean variation, it had to find if the answer was Yes or No and in the entity variation, it had to find the entity returned as an answer. We recognize the kind of answer based on the question and the answer themselves using a basic BOW approach. For the entity questions, RoBERTa has been fine tuned on the SQuAD2.0 dataset (Stanford Question Answering Dataset) consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text or unanswered. For the boolean questions, it is instead used the Boolq dataset, created by DeepMind to test boolean answer extraction.

5 Factual Statement Generation

Here our goals is to generate a factual statement that combines the answer extracted with the question. This will then allow us to use Stanford OpenIE to generate triples for fact checking. In order to accomplish this task, we slightly modify the structure of the question, for instance replacing the WH words with the answered entity name or creating an affirmative response instead of a boolean question. We employ basic grammatical and syntactical procedure here.

6 Fact Checking

For our fact-checking method, we focus on evaluating triples composed of two entities connected by one relation. The process involves examining the Wikidata page of the object entity and analyzing its properties. Our goal is to identify the property that most closely aligns with the subject entity of the relation. To achieve this, we use embedding techniques: we embed the textual description of the relation and also embed the labels of all properties associated with the object entity. These embeddings are then compared using cosine similarity to get the most similar property. Once the most relevant property is identified, we examine its value; if it includes the subject entity, we conclude that the fact is true. Otherwise, we determine it to be false.

7 Empirical test

Our testing dataset, essential for assessing our system, has been created by combining manual efforts and a bootstrapping-like method using a Large Language Model. The questions are divided in 5 categories: Capitals, Art, Cars, Animals, History. All the question answers have been manually verified and tweaked accordingly to guarantee sufficient variety. The dataset, can be found at the path `task_data/dataset_in.txt` and `task_data/dataset_out.txt`.