Information Retrieval and Question Answering

Individual:
Kishaiyan Vellaichamy Thangaraj

a1819309

The University of Adelaide
4333_COMP_SCI_7417 Applied Natural Language Processing
Lecturer: Dr. Alfred Krzywicki

Table of Contents

1. Abstract

In an age of vast and ever-expanding digital information, efficient retrieval and comprehension of knowledge have become critical. Leveraging Information Extraction (IE) techniques in Question-and-Answer (QnA) systems addresses this imperative by automating the extraction of relevant insights from unstructured textual data. The integration of Information Extraction (IE) techniques into Question-and-Answer (QnA) systems is critical for efficiently retrieving and comprehending knowledge from unstructured text. Our implementation framework makes use of advanced Natural Language Processing (NLP) techniques such as Named Entity Recognition (NER), Coreference Resolution, and Dependency Parsing using Spacy and NLTK packages available in python.

2. Introduction

In today's vast digital landscape, the volume of information available to users is enormous. During this deluge, the need for efficient and accurate information retrieval has become critical. Question and Answering (QnA) systems have emerged as invaluable tools, allowing users to ask questions in natural language and receive precise, relevant responses (Banko & Brill, 2001). Traditional keyword-based search engines frequently struggle to understand the nuances of human language and context, producing suboptimal results. Information extraction (IE) methods provide a potentially useful way to find and retrieve pertinent information from unstructured text data (Manning et al., 2014). Information extraction is a subfield of natural language processing (NLP) that focuses on automatically extracting structured information from unstructured or semi-structured text (Manning et al., 2014). By leveraging linguistic patterns, machine learning algorithms, and domain-specific knowledge, IE systems can identify entities, relationships, and events within textual data.IE encompasses several key tasks, including Named Entity Recognition (NER), Relation Extraction, and Event Extraction (Manning et al., 2014). NER involves identifying and classifying entities such as people, organizations, and locations mentioned in text. Relation Extraction identifies semantic relationships between entities, whereas Event Extraction detects events or actions described in the text.

2.1 Information Retrieval based Question Answering

Modern Question Answering (QA) systems rely heavily on Information Extraction (IE) techniques to extract structured data from unstructured text. Identification of entities, relationships, and events within textual data is made possible by IE's use of procedures like Named Entity Recognition (NER), Relation Extraction, and Event Extraction (Manning et al., 2014). NER is particularly crucial to QA systems since it allows objects stated in user questions and relevant documents to

be identified and categorized. NER could be used, for instance, in a medical QA system to detect certain medical entities like illnesses, symptoms, and therapies. Conversely, Relation Extraction helps QA systems answer complicated questions that need contextual information to be understood by helping them identify semantic links between things (Banko & Brill, 2001). Furthermore, by enabling QA systems to identify and decipher events reported in textual sources, Event Extraction approaches improve their capacity to deliver accurate and pertinent responses (Ji et al., 2010). Researchers have made major strides in information retrieval and natural language understanding by incorporating IE approaches into QA systems, which has resulted in more precise and efficient QA systems across a range of areas.

3. Preprocessing

3.1 Stop Words

The decision to retain stop words in the preprocessing phase was based on the recognition that stop words, although common and frequently occurring, often convey essential semantic meaning within sentences. Stopped words like "the," "is," and "are" help sentences make sense grammatically and contextually. Eliminating them arbitrarily may remove important information required to understand the text. By preserving the stop words, the QA system retains a more nuanced understanding of the language, allowing it to capture subtleties and contextual nuances that would otherwise be lost.

3.2 Lemmatization

Lemmatization, a process that reduces words to their base or dictionary form, was intentionally avoided in the preprocessing phase to maintain the specificity and richness of the language within the QA system. This can lead to information loss even while it can lower the dimensionality of the text data and simplify the vocabulary, especially in languages with complicated morphology. System can better capture the diversity of language usage and guarantee that no nuances or distinctions are lost in the process by keeping original word form.

3.3 Meta Data

To ensure that the QA system focuses solely on the substantive content relevant to answering questions, a preprocessing step was implemented to remove all words preceding the first question mark in each document. Any metadata or introductory language that comes before the content's main body is effectively removed in this step. The QA system may focus solely on the main textual content by eliminating this unnecessary material, which will increase the precision and applicability of the responses it provides to customer enquiries.

```python
def process_document(document):
    # Split the document into words
    words = document.split()

    # Check if the first word is all uppercase
    if words[0].isupper():
        # Find the index of the first question mark if it exists
        try:
            first_question_index = words.index('?')
        except ValueError:
            # '?' not found, set first_question_index to the length of the list
            first_question_index = len(words)

        # Remove words until the first question mark (inclusive)
        words = words[first_question_index:]

    # Join the remaining words back into a string
    result = ' '.join(words)
    # Check if the first character of the result is not '?'
    if result and result[0] != '?':
        return result
    else:
        # Return the result without the first character if it starts with '?'
        return result[1:]


# Assuming data is a DataFrame containing articles in a column named 'article'
# Apply process_document to all documents in the 'article' column
data['processed_article'] = data['article'].apply(process_document)
```

3.4 Sentence Tokenizer

During the preprocessing stage, each sentence in the pertinent context was tokenized using the NLTK sentence tokenizer. Sentence tokenization divides text into discrete sentences so that more in-depth processing and analysis can be done. The QA system can understand and extract information more precisely and with greater flexibility when the text is broken down into smaller units. By treating each sentence as a separate unit of meaning, the system can process user queries more accurately and retrieve pertinent information more quickly. This process is known as sentence tokenization.

4. System Architecture

The Information Extraction-based Question Answering (IE-QA) system framework that has been proposed is structured to manage user queries and extract relevant information from textual documents. The framework follows a methodical process:

User Input:

The Article ID and the question about that article are the first pieces of information that the system receives from the user.

Preprocessing:

The article that corresponds to the supplied Article ID goes through preprocessing procedures after the user input is received. This includes using libraries like Spacy and NLTK for tasks like tokenization, removing unnecessary metadata, and other necessary preprocessing techniques. After the preprocessed article is retrieved, the Coreference Resolution based on neuralcoref on Spacy finds the Coreferences in the article giving us the head and the list of the

Sentence Extraction:

The preprocessed article is then processed to identify relevant sentences pertaining to the user's question. Sentence Transformers and TF-IDF (Term Frequency-Inverse Document Frequency), two sentence embedding techniques, are used to accomplish this. They use cosine similarity to calculate the similarity between each sentence in the article and the user's question.

Context Selection:

The system chooses a range of sentences centered around the sentence that most closely resembles the user's query from the collection of pertinent sentences found in the earlier step. The context for responding to the question is usually formed by the two sentences that come before and after the most similar sentence in this span.

```
def extract_relevant_sentences_hf(context, question,num_context_sentences=0):
    model_name = "paraphrase-MiniLM-L6-v2"
    model = SentenceTransformer(model_name)

    context_sentences = nltk.sent_tokenize(context)
    # Encode the question and context
    question_embedding = model.encode(question, convert_to_tensor=True)
    context_embeddings = model.encode(context_sentences, convert_to_tensor=True)

    # Compute cosine similarity between question and context sentences
    cos_similarities = util.pytorch_cos_sim(question_embedding, context_embeddings)[0]

    cos_similarities = util.pytorch_cos_sim(question_embedding, context_embeddings)[0]

    # Find the index of the sentence with the maximum similarity
    max_similarity_index = cos_similarities.argmax().item()

    start_index = max(0, max_similarity_index - num_context_sentences)
    end_index = min(len(context_sentences) - 1, max_similarity_index + num_context_sentences)

    context_paragraph = ' '.join(context_sentences[start_index:end_index + 1])

    return context_paragraph,cos_similarities[max_similarity_index].item()
```

Question Type Determination:

The system analyzes the user's question to decide its type, considering words such as "what," "why," "how," "which," and "when." This aids in question classification and directs the following steps in the answer extraction procedure.

```python
def questiontype(question):
    # Define question tags
    questiontags = ['WP', 'WDT', 'WP$', 'WRB']

    # Tokenize and tag the question
    question_POS = pos_tag(word_tokenize(question.lower()))

    # Identify question tags
    question_tags = [token for token, tag in question_POS if tag in questiontags]

    # Check if there is only one question tag and it's not "what"
    if len(question_tags) == 1 :
        return True
    else:
        return False
```

Answer Type Determination:

The method of classifying the answer type according to the content and organization of the question is essential for directing the system's search for pertinent data. The system can focus its search and retrieve relevant material within the stated context by identifying the intended entity or information that the user is seeking. For example, if the enquiry is about historical events, the system might identify "historical events" as the answer type and give priority to obtaining data for this category. The system can provide the user with more relevant and customized answers by dynamically adjusting to their query and determining the type of answer that is most suited.

```python
def answertype(context, question):
    nlp = spacy.load('en_core_web_sm')

    if questiontype(question):
        t = 'DESCRIPTIVE'
        word = word_tokenize(question.lower())

        if any(x in word for x in person_question_words):
            t = 'PERSON'
        elif 'where' in word:
            t = 'GPE'
        elif any(x in word for x in date_related_words):
            t = 'DATE'
        elif 'when' in word or any(x in word for x in date_related_words):
            t = 'DATE'
        elif any(x in word for x in money_words):
            t= 'MONEY'
```

Answer Extraction:

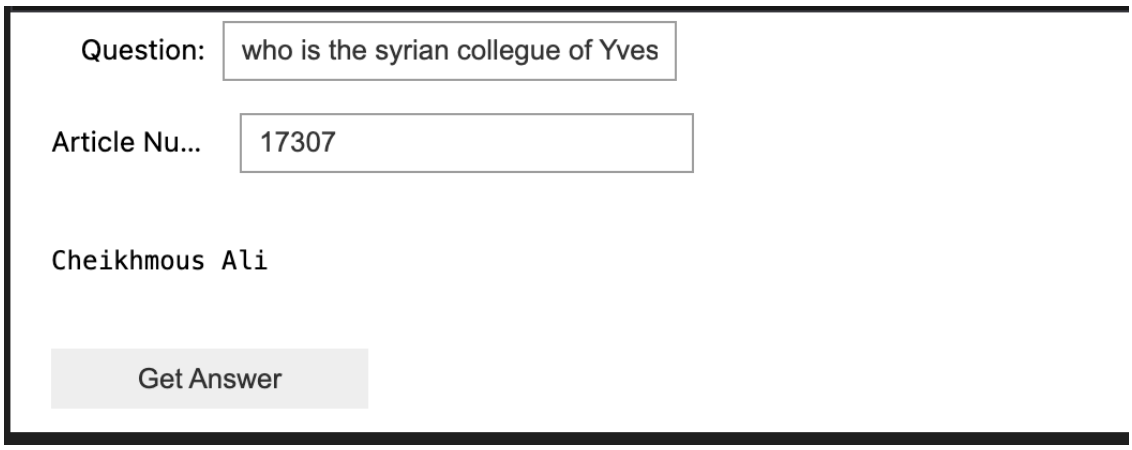
The system uses the identified answer type as a starting point to search and extract the pertinent answer entity from the context. To guarantee that the response is right and pertinent, this entails choosing the answer from the context according to the recognized answer type. Furthermore, the system chooses the head of the coreference cluster as the answer if the solution is found within any of the coreference clusters. By doing this, you can be sure that the system will correctly and coherently respond, capturing any references or pronouns in the context. Similarly, the system modifies its search criteria based on whether the question relates to specific geographic locations or scientific topics. This adaptive strategy increases the accuracy of the responses given while simultaneously improving the efficiency of the information retrieval process. The extracted answer is the system's final response to the user query if the answer cannot be found in any coreference cluster. By using this method, the system is better able to manage coreference resolution and can guarantee that the response provided to the user is right and in line with the context.

```
    if t == 'DESCRIPTIVE':
        # Perform descriptive processing
        # For now, return the question itself
        return question

    else:
        # Perform processing based on entity extraction
        # Extract entities of the identified answer type from the context


        doc = nlp(context)

        entities = [ent.text for ent in doc.ents if ent.label_ == t]
        #events = extract_events(doc)  # Extract events from the context
        #relations = extract_relations(doc)  # Extract relations between entities

        if coref_clusters:
            # Check if the first entity is in any head cluster
            first_entity = entities[0] if entities else None
            for cluster in coref_clusters:
                if first_entity and first_entity in cluster.main.text:
                    return cluster.main.text  # Return the head of the coreference cluster

        # If entities of the answer type are found, return the first one
        if entities:
            return entities[0]
```

5. User Interaction with the System

To use the system downloaded locally onto a machine with Python 3.7.16, it's crucial to ensure that all the necessary requirements specified at the top of the notebook are installed.

Once all the requirements are successfully installed, the system can be run by executing the Python script or notebook. This starts the process of execution, whereby the system carries out its intended operations. The user will see an interactive widget interface when the execution is finished. The main point of contact for the user is this interface. The user can input an article ID and a question using this feature. To extract the required data from the system, these inputs are essential. The user has the option to submit the query after supplying the required data. The system then processes the inputs and performs the necessary operations to retrieve the answer corresponding to the provided question and article ID. This answer is subsequently displayed to the user within the interactive widget interface, providing them with the information they requested. Furthermore, the interactive widget interface offers a seamless and intuitive user experience, allowing individuals to interact with the system effortlessly. Effective information retrieval is made possible by the ability for users to investigate different enquiries and receive real-time results. Users can also customize their enquiries more precisely by using the interface's options for extending or narrowing the search parameters. To improve user happiness and engagement, the system may also include features like error management and feedback systems inside the interface. To improve user happiness and engagement, the system may also include features like error management and feedback systems inside the interface.



6. Conclusion

The current system, while functional, demonstrates certain limitations in its ability to provide accurate responses to user queries. The way user questions are handled is one obvious place for development. Lemmatization and stop word removal are two strategies that the system could use to better grasp the main idea of the question and improve its ability to identify important details and context. Lemmatization would help by breaking down words into their dictionary or more basic

form, allowing for a more thorough examination of the question. In a similar vein, eliminating stop words would eliminate common terms with little semantic significance, enabling the system to concentrate on the crucial elements of the query. Furthermore, adding a knowledge graph has the potential to improve system performance. By leveraging a structured representation of information and relationships between entities, the system could navigate complex knowledge domains more adeptly, leading to more accurate and contextually relevant answers. The implementation of a knowledge graph would facilitate the system's ability to contextualize data, deduce relationships between ideas, and derive responses from a wider range of sources. Therefore, by putting these improvements into practice, the system might be able to get over its current shortcomings and provide users with a more advanced and accurate information retrieval experience.

Along with improving the system's abilities with language stop word elimination as well as the unification of an expertise chart, it is crucial to review the efficiency of these enhancements quantitatively. One method to evaluate the system's precision is by utilizing analysis metrics such as F1-score. The F1-score supplies a well-balanced procedure of a system's accuracy plus recall, considering both incorrect positives and downsides. By contrasting the system's anticipated solutions versus a collection of ground fact information the F1-score uses understandings right into the system's efficiency in properly determining appropriate info.

During testing, the system would be presented with a dataset containing questions along with their corresponding correct answers. After processing these questions using the implemented improvements, the system's predicted answers can be compared against the ground truth answers. Any discrepancies between the predicted and correct answers are then evaluated using the F1-score metric. Typically, an F1-score around 50% to 60% indicates moderate accuracy, suggesting that the system correctly identifies relevant information in approximately half of the cases. However, it's essential to interpret the F1-score in conjunction with other metrics and qualitative assessments to gain a comprehensive understanding of the system's performance.

7. References

[1] Banko, M., & Brill, E. (2001). Mitigating the paucity-of-data problem: Exploring the effect of training corpus size on classifier performance for natural language processing. Proceedings of the First International Conference on Human Language Technology Research.

[2] Ji, H., Li, S., & Li, Q. (2010). Overview of the TAC 2010 knowledge base population track. Text Analysis Conference.

[3] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.

[4] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. Proceedings of 52nd annual meeting of the Association for Computational Linguistics: system demonstrations, 55-60.

[5] Jurafsky, D., & Martin, J. H. (2019). Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. Pearson.