ABSTRACT

Automatic question answering System(QAS) is a highly researched topic in the field of Natural Language Processing and Information retrieval. A QAS is used to retrieve the correct answer for a given question from a paragraph. The performance of even the state-of-the-art QA systems is still not up to the level of a human. To improve the accuracy of such systems, it is necessary to implement different approaches only if it improves the performance This paper experiments into a rule-based approach for what-questions using dependency parsing. With the help of extensive dependency relations, we retrieve the exact answers for the factoid questions. Our experimental results show that our method has an accuracy of over 60%. Dependency parsing can be used in addition to other question answering techniques to improve the accuracy of existing state-of-the-art methods.

Keywords: Natural Language Processing, Question answering system, dependency parsing, rule-based, open domain

1. INTRODUCTION

Question answering systems are the next step in information retrieval systems. A QA system is required to succinctly retrieve answers for a question asked from a given document[1]. A QA system will decrease the effort needed to find answers from a long passage or document. An Automated QA System can be applied to various fields like medical, educational and commerce.[2]

A QA system can be classified into three domains, closed domain, open domain and restricted-domain. A closed domain QAS will require domain-specific knowledge. An example of one of the earliest closed domain QA system is LUNAR, that answers questions on soil samples taken from Apollo lunar exploration[3]. An open domain system is not limited to a specific domain. It can use different documents to retrieve the correct answer. Restricted-domain is in between the open domain and the closed domain. It answers domain-specific questions within a document collection.[3].

The main challenge when developing a question answering system is that a certain level of language understanding is required in order to retrieve the correct answers. The goal of NLP is "to accomplish human-like language processing"[4].

This paper deals with a single document open-domain QA system. Given a reading comprehension (single document), the system is able to answer what-type of questions. A subset of the SQuAD dataset is used in this paper. What questions were chosen as they have a variety of possible answers[5]. The questions can be classified by the type of answer expected, like factoid or definition questions[6], [7]. In our work, we follow a human-like approach to retrieve the accurate answers to the questions by using dependency relations.

The design of the QA system consists of a question classification stage, where we decide on the type of answer that is expected. The answer retrieval consists of two parts. The first part deals with the retrieval of the sentence containing the answer and the second part is to find the keyword which will lead us to the answer. The third stage is to retrieve the exact answer by parsing through a dependency tree.

2.LITERATURE REVIEW

[8]Developed QUARC, a rule-based QA system that retrieves the sentence containing the answer to the question. They developed a set of rules that awards a certain score to sentences based on semantic and lexical heuristics. QUARC uses a different set of rules for different Wh-type questions (who, what, why, when, where). The model best performs on WHEN questions with 55% accuracy and worst on WHAT and WHY questions with only 28% accuracy.

[9]Proposed an architecture that uses documents as input and answers multiple choice type of questions. Information extraction is carried out by Lucene information retrieval engine. The engine indexes the document collection and also performs linguistic processing like stemming, parts-of-speech tagging and anaphora resolution. The architecture comprises three modules namely, the document processing module which consists of an XML parser, a query analyzer and a document pre-processing module. The second module is the information extraction which consists of hypothesis generation and information retrieval. Hypothesis generation produces a set of five possible answers by substituting the question keyword with each of the MCQ options. The information retrieval module then returns the relevant passage for each hypothesis and calculates a lexical similarity score. The final module is to answer validation and answer selection.

[10] Discussed a QA system for Frequently asked questions. They developed an open domain QA system that retrieves the list of relevant FAQs that are related to the user's query. They followed a combination of AND/OR search and combinatorics on the QA4FAQ task website. This approach resulted in an exhaustive search and the maximum number of the relevant result is retrieved. The system also uses Apache Nutch, which is a highly scalable web crawler for indexing, searching and ranking of the data. The ranking is done by Nutch’s ranking algorithm.

[11] Proposed two methods for extracting answers for a speech corpora. The first one uses a reranker that uses only lexical information. The second one uses dependency parsing to score the similarity between syntactic structures. First, they introduced a baseline system which uses only a shallow description. This system is then improved using a reranker and then a dependency parser is used to get more informed features.

[12] implemented a statistical model using a statistical chunker to create a phrase-based-query from the question. Then a search engine uses the query and returns the N most relevant articles from the web. The answer is then retrieved by computing answer language model probability and an answer question translational model probability. The answer language model probability

indicates how similar the retrieved answer is when compared to the training corpus. The probability of the question-answer translational model indicates how similar is the retrieved question-answer pair is to other pairs in the training corpus.

[13] Proposed an open-domain QA system model based on a semi-structured Wikipedia knowledge model. The architecture consists of various modules: a question analysing module that is used to determine the nature of answers required, a document retrieval module which fetches the right document and multiple answers matching modules namely article content module, article structure module, category structure module, an infobox module and a definition module. Finally, an answer merging module which uses a strategy to determine the combination of answer matching module to be used either in sequence or in parallel.

[14] proposed a closed domain question answering system on tourism. The model is implemented using corpus on Pune tourism. The author describes a 9 step process to retrieve a short string of answers on tourism questions like distance, hotels, attractions etc. A web crawler is used to crawl a list of websites related to tourism in Pune. When a user asks a question, it is first preprocessed before identifying the keywords. Then the web crawler crawls through the websites and all the sub links. The crawled data is then preprocessed. After this, the engine processes the tokens and the query keywords using NLP rules. The system uses Procedure Programming style for extracting answer.

[15] implemented a surface pattern-based question answering system. They have suggested an approach to automatically learn common patterns which can be used to answer certain types of questions. A tagged corpus is built using machine learning technique of bootstrapping. The system then assumes each sentence to be a sequence of words and searches for patterns. The patterns which were automatically learnt are then tested on questions from the TREC-10 set.

[1] Designed a closed domain factoid QA system, focusing on why-type questions. They allow a knowledge-based approach using the term frequency-inverse document frequency algorithm. They have defined four parameters to decide the type of answers for why-type questions namely: because, neither, either and thus, thought. When a user poses a question, the system pre-processes it and searches for the words in the knowledge On the retrieved results, they apply the TF-IDF vectorizer and calculate the cosine similarity to determine the most similar answer.

[16] Proposed a soft pattern matching model for definitional question answering. They have suggested a soft matching model based on bigams and Profile Hidden Markov model, as the hard matching patterns like regular expressions often performs poorly with variation in language. Both these models give a formal probabilistic method to find lexico-syntactic patterns represented by token sequences.

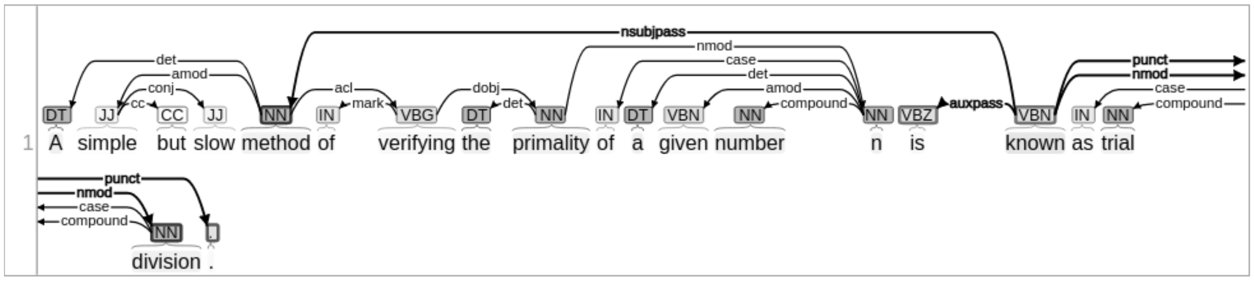
3.METHODOLOGY

In this section, we discuss how dependency relations can be used to retrieve answers for what-type question in reading comprehension.
We first present how sentences are represented using dependency trees and then implement a rule-based approach to extract answers from these sentences.

Dependency trees

A dependency tree can be defined as a directed graph where the words are the vertices and the arcs are the syntactic dependencies between them. The type of dependency is defined by relations. A typical dependency link has a head and a dependent. The dependent is usually the modifier, object or complement[17]. The most important node in the sentence is the root. Every other word is directly or indirectly dependent on it. In our work, we work with a small set of dependency relation patterns which can be used to extract the answer.

Example:
SENTENCE: A simple but slow method of verifying the primality of a given number n is known as trial division.



nsubjpass
nmod
case
det
amod
compound
conj
cc
acl
mark
dobj
auxpass
punct
DT JJ CC JJ NN IN VBG DT NN IN DT VBN NN NN VBZ VBN IN NN
1 A simple but slow method of verifying the primality of a given number n is known as trial
NN .
division .

The above figure shows the dependency parsing of a sentence. The root word is *known.*

DEPENDENCY	ACRONYM	DEFINITION
nmod	nominal modifier	The nmod relation is used for nominal modifiers of nouns or clausal predicates
dobj	direct object	The dobj of a VP is the noun phrase which is the (accusative) object of the verb.
amod	adjectival modifier	An amod of an NP is any adjectival phrase that

		serves to modify the meaning of the NP
nsubj	nominal subject	A nsubj is a noun phrase which is the syntactic subject of a clause.
acl	clausal modifier of a noun	acl is used for finite and non-finite clauses that modify a noun.
advcl	Adverbial clause modifier	An advcl of a VP or S is a clause modifying the verb
obj	object	An obj is the object of a verb
compound	compound	noun compounds
conj	conjunct	A conj is a relation between two elements connected by a coordinating conjunction, such as “and”, “or”, etc
nmod: poss	possessive nominal modifier	nmod: poss is used for a nominal modifier which occurs before its head in the specifier position used for ‘s possessives.
appos	appositional modifier	An appos of an NP is an NP immediately to the right of the first NP that serves to define or modify that NP. It includes parenthesized examples.

Table of the list of dependency relations used in this paper[18][19].

QA system modules

Our proposed QA system has three modules. The first module is question analysing, to identify the type of what-question. After classifying the question as either descriptive or non-descriptive, the next step is to retrieve the sentence containing the answer from the reading comprehension passage. This is done by ranking each sentence in the paragraph based on how similar it is to the question. This approach was followed because, in a reading comprehension test, the sentence containing the keywords from the question is most likely to contain the answer. The third module is answer extraction, where we apply the necessary dependency rules to extract answers from the retrieved sentence. In addition to the dependency rules, we have also used some POS-tag chunkers.

Answer Extraction

We have used dependency parsing to extract answers for factoid what-type questions. Firstly, the main keyword from the question is identified. Most often when a human is solving a reading comprehension test, the first thing he looks for is the keyword. The clue to the answer is mostly found in the question. Correctly identifying this keyword helps us easily retrieve answers using dependency parsing. This keyword can be identified by

1. A noun present in both the sentence and the question
2. Dependency relations of the ROOT word from the question

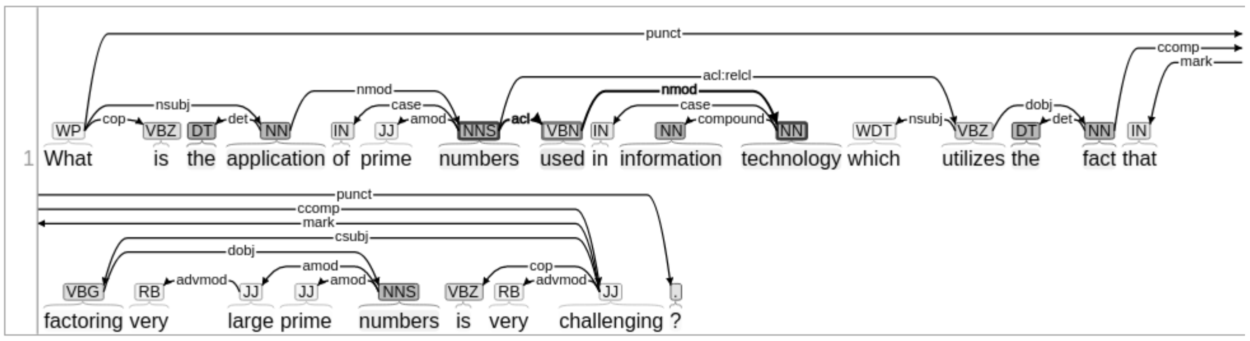
Algorithm for finding the keyword:

```
1. If (ROOT of the sentence is a NOUN)
    a. If the word or synonym of the word is in question:
        i. return keyword
2. else if ( ROOT of the sentence is a VERB):
    a. In the question, check for dependency: ROOT-> NMOD
    b. If the word or synonyms of the word in the node NMOD is present in the
       question,
    c. return keyword
3. else if (ROOT of the sentence is a NOUN but not present in the question)
    a. In the question, check for dependency: ROOT->NSUBJ
    b. If the word or synonyms of the word in the node NSUBJ is present in the
       question,
    c. return keyword
4. else return ROOT of sentence
```

Example:

QUESTION: What is the application of prime numbers used in information technology which utilizes the fact that factoring very large prime numbers is very challenging?

SENTENCE: Primes are used in several routines in information technology, such as public-key cryptography, which makes use of properties such as the difficulty of factoring large numbers into their prime factors.


1 What is the application of prime numbers used in information technology which utilizes the fact that factoring very large prime numbers is very challenging ?

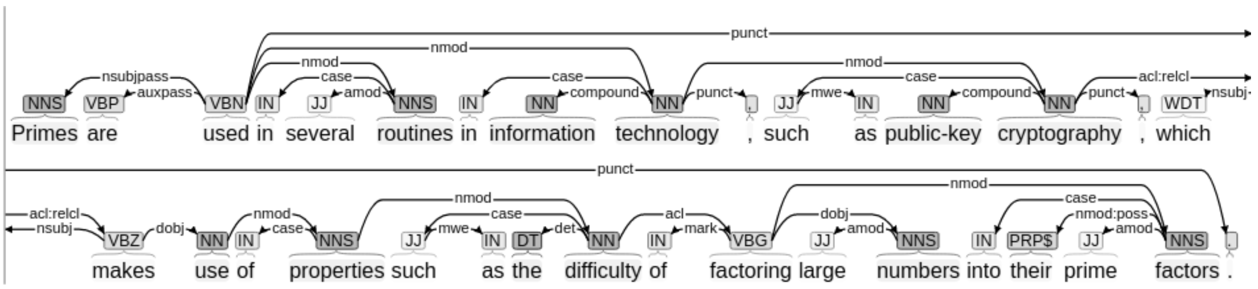
The keyword extracted using RULE #2: technology

Once the keyword is retrieved, we then use dependency rules to extract the answers.

Dependency rules

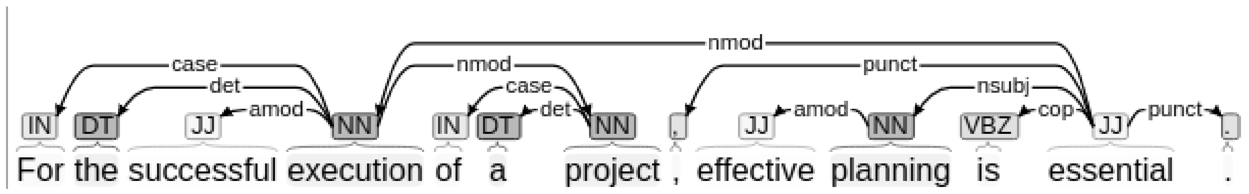
1. keyword -> NMOD -> COMPOUND
2. keyword -> NMOD -> CONJ
3. keyword -> NMOD: POSS -> NMOD
4. AMOD <- NSUB <- keyword
5. COMPOUND <- NSUB <- keyword
6. NSUB <- keyword
7. keyword -> OBJ -> AMOD
8. keyword -> ACL/ADVCL ->DOBJ ->NMOD ->CONJ

These dependency rules were extracted after evaluating different kinds of questions.


Primes are used in several routines in information technology , such as public-key cryptography , which makes use of properties such as the difficulty of factoring large numbers into their prime factors .

The answer *public-key cryptography* was extracted using dependency rule #1

Question: What is essential for the successful execution of a project?
Extracted Sentence: For the successful execution of a project, effective planning is essential.
Ground answer: effective planning



nmod
case
det
amod
nmod
case
det
punct
nsubj
amod
cop
punct
IN
DT
JJ
NN
IN
DT
NN
,
JJ
NN
VBZ
JJ
.
For the successful execution of a project , effective planning is essential .

The answer ***effective planning*** can be extracted using dependency rule #4

The dependency relations along with POS chunkers can be used to extract answers. One main limitation while using dependency relations is that it does not perform well for descriptive answers or long compound sentences.

4.EXPERIMENTS AND RESULTS

We have described the hand-crafted dependency rules used in extracting the keyword and the answers. Now we turn to the implementation of the QA system. Our research question is in proposing a new solution to extract answers for what-type questions. The motive behind choosing a rule-based approach is because it provides a better insight into how dependency relations can be used to extract answers. Since we have proposed a new method of extracting answers, a rule-based approach also enabled easy interpretation of the proposed system. We aimed to follow a method which closely follows a human-like approach to solving reading comprehension tests.

We used the SQuAD dataset for our QA system. It consists of questions posed by crowd workers on a set of Wikipedia articles, where the answer to each question is a segment of text (or span) from the corresponding reading passage[20]. From this dataset, we created a subset of passage-question pairs containing what-type questions. The SQuAD dataset only provides the passage and the question, hence allowing the system to choose an answer from the passage instead of from a multiple choice. The Stanford CoreNLP library was used for dependency parsing and named-entity recognition.

We need to first pre-process the passage and the questions, for which we make use of tokenization and stopword removal. In order to find the sentence containing the answer, we find the most similar sentence to the question using a model trained using Word2Vec. Once we have the question-sentence pair, we head on to analysing the question to determine the type of answers it requires.
We then apply the dependency relations in addition to POS-tag patterns to retrieve the keyword. In order to include the cases where the keyword is a synonym of the extracted word, we have

also used the synset library and lemmatization. Using this keyword, we applied the above mentioned 8 dependency rules using a dependency parser to extract the answers.

We ran some experiments on these dependency rules to determine its contribution to the question answering system. We found that the dependency relations have relatively low accuracy for descriptive questions and sentences containing long-distance dependencies. In addition to these, dependency relations are not required for named-entity answers (like city, location, person, organization, name) as they can be directly retrieved using a named-entity recogniser.

Steps in the algorithm are briefly described below:

```
1) Preprocess question and passage
2) Determine question type
3) Calculate similarity to get the most similar sentence.
4) If question type = "non descriptive"
    a) Keyword = dependency_keyword(question, sentence)
    b) return dependency_parsing(keyword,sentence), else return None
    c) if None
    d) return named_entity(question, sentence)
    e) else
    f) Return chunkers(sentence), else return None
5) If question type = "descriptive"
    a) Keyword = dependency_keyword(question, sentence)
    b) return dependency_parsing(keyword,sentence), else return None
    c) if None
    d) Return chunkers(sentence), else return None
    e) else:
    f) Return sentence
```

5.RESULTS AND ERROR ANALYSIS

In this section, we present the results to evaluate our proposed dependency parsing method for reading comprehension test questions

Question type	Number of questions	Correctly answered	Accuracy
Descriptive	21	12	57%
Factoid	55	35	63.64%

Total questions	76	47	61.84%

The results of the proposed question answering system are shown in the above table. The accuracy is satisfactory for the factoid questions. It is difficult to determine the correct dependency relations for descriptive questions as it does not perform well in case of long-distance dependencies.
Out of the 55 factoid questions, 35 were answered with the help of above dependency rules and named entity recognition. After a thorough review of each passage-question pair, we noticed that the similarity metric was not accurate enough to retrieve the correct sentence. It is necessary to extract the sentence containing the answer in the first place. As a possible solution, we can implement a better similarity metric.
Another problem we found was that it is difficult to determine the correct answer when two or more dependency relations retrieve the answer. Implementing a scoring system could be a possible solution, but there is ambiguity on what basis to score the retrieved answer.
The reason for the lower accuracy of definition type questions is because the extracted were inexact. The answers were only partly retrieved and sometimes contained irrelevant parts of the sentences.

6.CONCLUSION AND FUTURE WORK

This research was focussed on finding a different approach to retrieve answers for what-type questions. The power of dependency parsing for extracting answers was understood by devising a rule-based approach. A rule-based approach was preferred as it aided in getting a clear analysis of how these rules work and the was it could be implemented in the future. The present question answering systems are still nowhere close to the accuracy level of a human. To achieve human-like accuracy, it is necessary to make use of multiple strategies to find answers.

In this paper, the dependency rules were devised by hand after thoroughly examining the reading comprehension questions. Hence we have used only a limited number of dependency parsing rules. These simple rules are probably not sufficient to handle different types of factoid questions. As the next step, we would like to apply these rules to other types of wh questions and find more such rules. We also hope to explore the possibility to automatically learn these dependency relation patterns that can be used to extract answers. Finally, we want to experiment on how these dependency parsing rules will work in combination with the existing question answering system models.

7.REFERENCES

[1] A. M. Pundge and C. Namrata Mahender, “Evaluating Reasoning in Factoid based Question Answering System by Using Machine Learning Approach,” in *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, Oct. 2018, pp. 821–825, doi: 10.1109/CESYS.2018.8724085.

[2] I. Thalib, Widyawan, and I. Soesanti, “A Review on Question Analysis, Document Retrieval and Answer Extraction Method in Question Answering System,” in *2020 International Conference on Smart Technology and Applications (ICoSTA)*, Surabaya, Indonesia, Feb. 2020, pp. 1–5, doi: 10.1109/ICoSTA48221.2020.1570614175.

[3] A. Mishra and S. K. Jain, “A survey on question answering systems with classification,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 28, no. 3, pp. 345–361, 2016, doi: https://doi.org/10.1016/j.jksuci.2014.10.007.

[4] R. Barskar, G. F. Ahmed, and N. Barskar, “An Approach for Extracting Exact Answers to Question Answering (QA) System for English Sentences,” *Procedia Eng.*, vol. 30, pp. 1187–1194, 2012, doi: 10.1016/j.proeng.2012.01.979.

[5] E. Riloff and M. Thelen, “A Rule-based Question Answering System for Reading Comprehension Tests,” p. 7.

[6] M. A. Calijorne Soares and F. S. Parreiras, “A literature review on question answering techniques, paradigms and systems,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 32, no. 6, pp. 635–646, Jul. 2020, doi: 10.1016/j.jksuci.2018.08.005.

[7] O. Kolomiyets and M.-F. Moens, “A survey on question answering technology from an information retrieval perspective,” *Inf. Sci.*, vol. 181, no. 24, pp. 5412–5434, Dec. 2011, doi: 10.1016/j.ins.2011.07.047.

[8] E. Riloff and M. Thelen, “A Rule-based Question Answering System for Reading Comprehension Tests,” 2000, Accessed: Dec. 06, 2020. [Online]. Available: https://www.aclweb.org/anthology/W00-0603.

[9] H. Gómez-Adorno, D. Pinto, and D. Vilariño, “A Question Answering System for Reading Comprehension Tests,” in *Pattern Recognition*, vol. 7914, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, J. S. Rodríguez, and G. S. di Baja, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 354–363.

[10] D. Bhardwaj, P. Pakray, J. Bentham, S. Saha, and A. Gelbukh, “Question Answering System for Frequently Asked Questions,” in *EVALITA. Evaluation of NLP and Speech Tools for Italian*, P. Basile, F. Cutugno, M. Nissim, V. Patti, and R. Sprugnoli, Eds. Accademia University Press, 2016, pp. 129–133.

[11] P. Comas, J. Turmo, and L. M. i Villodre, “Using dependency parsing and machine learning for factoid question answering on spoken documents,” *undefined*, 2010. /paper/Using-dependency-parsing-and-machine-learning-for-Comas-Turmo/398529fe1628c85ccdf09abe83a1a66ffb8b18e7 (accessed Dec. 06, 2020).

[12] R. Soricut and E. Brill, “Automatic Question Answering: Beyond the Factoid,” p. 8.

[13] P.-M. Ryu, M.-G. Jang, and H.-K. Kim, “Open domain question answering using Wikipedia-based knowledge model,” *Inf. Process. Manag.*, vol. 50, no. 5, pp. 683–692, Sep. 2014, doi: 10.1016/j.ipm.2014.04.007.

[14] V. Bhoir and M. A. Potey, “Question answering system: A heuristic approach,” in *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, Bangalore, India, Feb. 2014, pp. 165–170, doi: 10.1109/ICADIWT.2014.6814704.

[15] D. Ravichandran and E. Hovy, “Learning surface text patterns for a Question Answering system,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL ’02*, Philadelphia, Pennsylvania, 2001, p. 41, doi: 10.3115/1073083.1073092.

[16] H. Cui, M.-Y. Kan, and T.-S. Chua, “Soft pattern matching models for definitional question

answering," *ACM Trans. Inf. Syst.*, vol. 25, no. 2, p. 8, Apr. 2007, doi: 10.1145/1229179.1229182.

[17] M. Covington, "A Fundamental Algorithm for Dependency Parsing," *undefined*, 2004. /paper/A-Fundamental-Algorithm-for-Dependency-Parsing-Covington/418d52298299314905e56b2ac9bb0888e62bf242 (accessed Dec. 10, 2020).

[18] "Universal Dependencies." https://universaldependencies.org/ .

[19] M.-C. de Marneffe and C. D. Manning, "Stanford typed dependencies manual," p. 28.

[20] V. Ingale, P. Singh, and A. Bhardwaj, "Datasets for Machine Reading Comprehension: A Literature Review," *SSRN Electron. J.*, 2019, doi: 10.2139/ssrn.3454037.