Introduction

In the bustling world with modern technology, where we can easily gain information with the tip of our fingers, one of the features of the Internet we use most is searching for information through a search engine. Example questions that we could ask would be “Where is the Taj Mahal located?” or “What is the average period length?”. However, do we, as users, ever wonder how these search engines find the answer to these questions? Does the search engine spew out answers based on random probability, or give answers that are true to the world? And as you are reading this, how is semantics relevant in this field?

In this paper, we will discuss Question Answering (QA), a system built to obtain the answer to the user’s query, either from a search engine, a chatbot, or a query database, from a set of relevant documents. We will also introduce another term, Information Retrieval (IR), a system that will retrieve the documents that are believed to be relevant to the query, and contain the answer. We can imagine IR as the suggested websites returned by the search engine, as they might contain the answers the user is looking for.

The field of semantics is relevant in question answering as the system would have to decide which meaning of the words that the users are specifically asking. Consider the question (1)

(1) Q: Should I wear a cap during interviews?

We as a search engine user would expect an answer to look like (2):

(2) A: No, you should not wear caps as they are informal and not professional…

We would be surprised if the search engine returned results like (3) and (4):

(3) A: Yes, a bottle cap is needed to ensure the content of the bottle does not spill…
(4) A: You can get the cheapest cap on these stores…

One problem we can identify here is that some words in a language can be used in different contexts, and convey different meanings. In the example above, how does the system know which cap are we referring to? As a language user, we would understand that the cap in the question refers to a type of headwear. However, according to Wordnet (Miller, 1995), the word "cap" has multiple meanings, including

cap (N)

1. Tight-fitting headdress
2. A top (as for a bottle)
3. Something serving as a cover or protection

   … (omitted for simplicity)

Moreover, how can the system ensure that they would retrieve the documents that discuss the proper attire for an interview, and not a bunch of websites selling the cheapest cap in town? Could the system figure out the correlation between "cap" and "interview"? This is where semantics come into play.

For the system to return the appropriate response, the system would have to know more about the query, specifically its meaning and how it relates to the real world. For example, the "World Cup" does not refer to the same cup as a coffee cup in the real world. However, a machine does not go to school and learn English their whole lives. Machines do not have the same language intuition as humans do. Machines also have no idea how the real world works, like how a cap looks. How is it possible for machines or systems to learn a language, or know how the real world operates?

Our goal at the end of this paper is to give an overview of the system in general and analyze the semantic features that can be found throughout the system. In this paper, we will discuss an overview of Question Answering and Information Retrieval. In the next section, we will talk about types of questions and answers and also Named Entity Recognition (NER) to identify the question and answer type. We would also talk about word representations, either as word vectors,

as a “bag of words”, or as embeddings. And in the last section, we will talk about using semantic roles in the IR system.

Question Answering (QA) Overview

Question Answering here refers to the task of answering the user’s query. We will discuss the types of questions that can be asked by the user and a general overview of the QA system.

Types of questions and answers

Questions can be categorized based on the type of answers the user wants. In English, typically a user will begin the question with any WH phrases (*who/whom, where, when, why, how*), hence we can easily determine the type of answer. Consider the questions (5) until (11) below:

(5) Q: How long is the distance between Ann Arbor and Flint?
(6) Q: Where is the nearest gas station?
(7) Q: Who is the Prime Minister of Malaysia?
(8) Q: What songs are in the Hamilton album?
(9) Q: Tell me about the independence of Indonesia.
(10) Q: What is acai?
(11) Q: Who is Alex Pereira?

For (5) until (7), we can answer these questions using just a few words, and we do not necessarily need a lot of documentation to get answers. We can probably just use one document and easily get the answer. For these questions, they are categorized as factoid questions. For (8), since there are multiple answers, this is called a list question. And for (9) until (11), since we need more extensive documentation, or a collection of documents to get a concise answer, these questions are classified as definition questions. For questions like (9) above, where it does not use any WH phrases, the QA system will simplify it to “*What X*”, for easier analysis. Although there could be other categories of types of questions, for the sake of simplicity, these are the only categories that we need for this paper.

We also need to categorize the type of answers, whether it would be a numerical answer or a more factual answer. For example, question (5) would have a numerical answer, with distance as the measurement. Question (6) would have a location as its answer, and question (7) would have a person as its answer type. For questions (8) until (11), since it is not possible to give out an answer in just one single word or sentence, their answer type would be an explanation. There are other types of answer types, and that is dependent on the system itself on how they would like to categorize their answers.

Question Answering System

As we introduced some question types and answer types, this would be relevant in understanding how the Question Answering system works. Figure 1 below shows a basic diagram of the QA system.

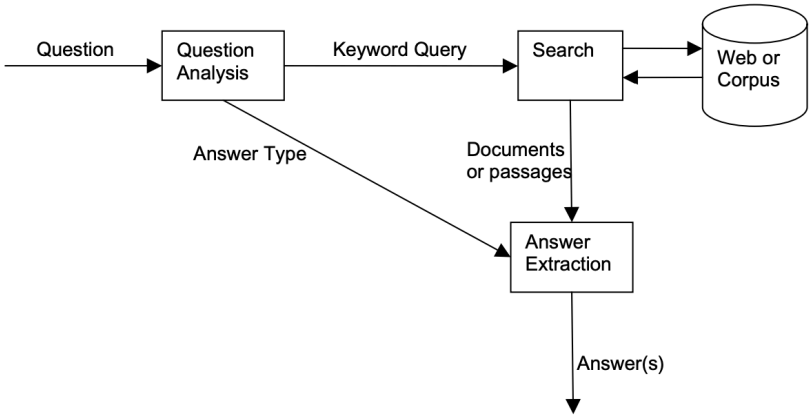

Figure 1: Basic diagram of the QA system (Prager, 2006)

As the user puts in a query, the question will be analyzed in the question analysis part to ensure that we get the correct answer type, question type, and the focus of the question. In this stage, the system will also transform the query into a keyword query that fits the QA domain. The transformation will sometimes include the droppings of stop words, explained in the IR section, and WH phrases. Some phrases in the original query might also be dropped or modified depending on how the phrase might be unhelpful for the search. The question analysis part will

spit out the keyword query as we discussed above and the answer type, which will be passed to the search and answer extraction stage respectively. The search stage here is where Informational Retrieval will come into play, and we will explain this in more detail in the next section. The search section is not only limited to the IR system, as it can also use other techniques such as Natural Language Processing (NLP), a knowledge base, or a hybrid of some systems (Soares & Parreiras, 2020), but in this paper, we would only focus on IR.

In the answer extraction stage, after the system retrieved the set of documents that are believed to contain the answer to the query, it will try to find the answer based on the answer type, and it will return a list of answers that are ranked based on their relevancy to the query. There are a few ways for QA to find the answers, including heuristics, pattern-based, relationship-based, and logic-based (Prager, 2006).

We will discuss relationship-based answer extraction here as this is closely related to semantics. By finding the semantic relationship between the words in the query, and also the words in the documents, we can find more accurate answers. Consider the question (12) below

(12) Who designed the Malaysian flag?

The relationship would be (13) below

(13) `[X.design], [design, “Malaysian flag”]`

This looks like the predicate representation of a verb. And by putting it into relationship representation, we can see how far the relationship goes. This is especially important when recognizing the accurate answer. Consider the candidate answers below:

(14) Mohamed Hamzah created the Malaysian flag.
(15) The design of the Malaysian Flag is modeled by the USA flag
(16) The government hosted a design contest nationwide for design submissions of the Malaysian flag

The relationships are represented respectively below

(17) `["Mohamed Hamzah".design], [design, "Malaysian flag"]`

(18) `[design, "Malaysian flag"], [design, model], [model, "USA flag]`

(19) `[design, "Malaysian flag"], [design, submission] [design, contest]...`

All the answer passages contain the word that appeared in the question. However, not all answers contained the answer. The system will calculate two scores, one being the passage score, which is the number of relationships without an "X" or unknown variable. Another score would be the intermediate candidate score, which is the number of relationships that matched with the variable. The final candidate score would be the weighted mean of the passage score and the intermediate candidate score. The more the relationships in an answer candidate (relationship here refers to the number of brackets in the representation), the lower the mean for that candidate's answer. For example, (17) would have a higher weighted score than (18) and (19).

There are other approaches for relationship-based answer extraction. One way would be using dependency trees, as represented by Figure 2 below. From the dependency tree, we would find the minimum approximate matching tree using the edit distance and the tree matching algorithm. Another approach would be using keywords, where phrases are indexed in preference to keywords (Arampatzis et al., 1998) from the bag of words. This is more well-known in IR systems and will be discussed in the IR section.

Named Entity Recognition (NER)

Named Entity Recognition (NER) here is used to determine the types of questions and answers. This is done by analyzing the type of WH-questions used in the query. By using NER, we get to ensure that we get to reduce the amount of documents retrieved and increase the accuracy of the answer returned. It also uses Wordnet to ensure the hyponyms of the answer type entity. Table 1 below shows the rules to match the answer type.

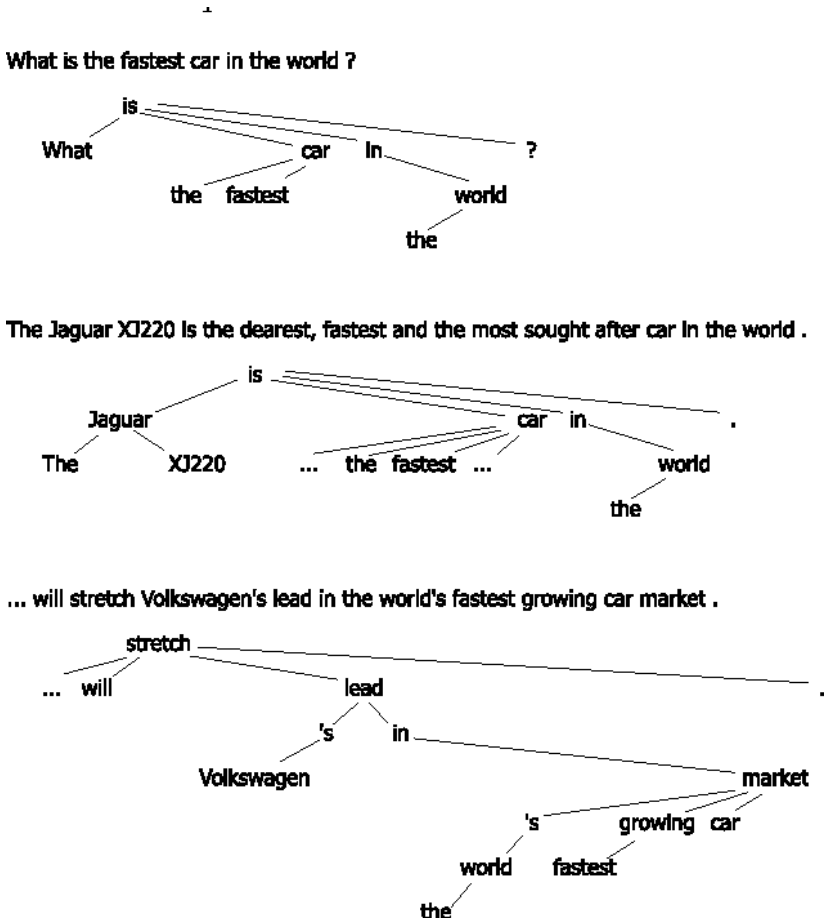

Figure 2: Example of dependency trees (Punyakanok et al. 2004)

Rules	Answer Type
Who + hyponym (person)	Person
Where + hyponym (location)	Location
Which + hyponym (organization)	Organization

Table 1: Rules to match the answer type.

As we see above, if the query uses the wh-phrase Who and is paired with a person, the answer type must be a person. Even though this is intuitive in English, these rules are much more useful

in other languages where the Wh-phrases might be vague and can be used in different entities, as seen in Table 2 below, where these rules are used in Spanish QA systems.

Rules	Answer Type
que (what/which) + hyponym (person)	Person
que (what/which) + hyponym (localization)	Localization
que (who) + hyponym (organization)	Organization

Table 2: Rules to match the answer type in a Spanish QA. Adapted from Toral et al. (2005)

Information Retrieval (IR) Overview

In this section, we will talk about how the IR system specifically works in the QA system. It is important to discuss IR when we talk about QA because while QA returns the answers to a user's query, IR will return the set of documents believed to contain the answer to the user's question. IR will return the documents that have been indexed according to their relevancy to the question.

The first process would be to assign weights to each word that is in our query. The system will remove all punctuation marks and compare each word to the documents in our database. If these words are found in more documents, they will have a higher weight as they have higher frequency. Words with higher weights are less helpful than those with lower weights as if they have lower weights, we can ensure that the documents would be relevant to the user's query. One example would be the word "Biden" would have less relevant documents and hence be more useful to the user rather than common names like "Michael" or "Sara". For the sake of simplicity, we won't dive into the computational aspects of this task.

From the task above, we would introduce another term, stop words. Stop words are words with high weights and occur often in documents. Examples of stop words would be determiners and auxiliary verbs like *the, a, is,* and *will.* These stop words are removed from the query before we assign the weights as we know these stop words will not help in the search process.

The system will then index the documents to indicate the most relevant document. We can imagine this as the top link returned by the search engine, among the other links it will return. This is done by vector calculations, where the words in the query are put in a vector, along with the words from the document. We will discuss more about vector semantics in the following sections. Calculations are omitted, however, those with higher scores are ranked higher in the document indexing.

Vector Semantics

Vector semantics is modeling words as vectors. Since most of the process involves computations, it will be easier to represent a word as a vector than a word as computers do not understand words. We can represent a word as a point in a multidimensional semantic space. The vectors for representing words are called embeddings, which come from the mathematical term of mapping from one space to another. Figure 3 below shows a two-dimensional projection of embeddings of words.

Figure 3: A two-dimensional projection of embeddings of words (Jurafsky & Martin, 2023)

From the figure above, notice how words are grouped. The green-colored words are positive, the red words are negative, and the blue are neutral function words. These are embedded using word similarity, and these embeddings are powerful especially when used in NLP. Words can be embedded not only by similarity but also by synonymy, connotation, and semantic roles.

When dealing with documents, we would use a term-document matrix to represent the number of times the word occurs in a document. Table 3 shows an example of a term-document matrix, where a row is defined by the words, and the columns are defined by Shakespeare's play, or equivalent to a document. Hence, the word *good* appears 114 times in *As You Like It*.

	As You Like It	**Twelfth Night**	**Julius Caesar**	**Henry V**
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Table 3: Term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the word (row) occurs in the document (column). Adapted from Jurafsky and Martin (2023)

We can represent the document as a vector, using the columns of the term-document matrix. So each document would have a vector of size $|V|$, where V represents the vocabulary size. If we use the table above as an example, the document *As You Like It* would have a vector of 4 dimensions, with the first dimension being the number of times the word *battle* occurs in that play. The same goes for every play on the table above. By representing the documents as a vector, we can see how similar each document is when we are retrieving documents for answering a query.

We can also represent words as a vector called a word vector. This is similar to the document vector, except that the dimension for the vector will be the number of documents. Hence for the word *wit*, the vector would be [20, 15, 2, 3], with each number representing the Shakespeare play, and this goes for all the words in the vocabulary. By using word vectors, we can see that similar words tend to have the same vectors as they often appear in similar documents.

It is also useful to represent words based on similarity. However, we first must define similarity. In the language sense, we tend to think of similarity as synonyms, for example, *big* is a synonym with the word *huge*. However, it can also be useful to think of similarity as words belonging to

the same category. For instance, cherry would be similar to strawberry in the sense that they are both fruits. We can represent this similarity as vectors, by calculating the amount of times these words appear in instances with other words. This vector is called the word-word co-occurrence matrix. Table 4 below shows the word-word co-occurrence matrix, where each number in the table represents the number of times the word on the word occurs with the word on the column. The word strawberry, for example, would be a vector of [0, …, 0, 0, 1, 60, 19, …]

	aardvark	…	computer	data	result	pie	sugar	…
cherry	0	…	2	8	9	442	25	…
strawberry	0	…	0	0	1	60	19	…
digital	0	…	1670	1683	85	5	4	…
information	0	…	3325	3982	378	5	13	…

Table 4: The word-word co-occurrence matrix, choosing only 6 words for simplicity. It is important to note that real vectors would have a larger dimension, and will be more sparse (there will be more 0 values in the vector). Adapted from Jurafsky and Martin (2023)

From the table below, we can see how both words *strawberry* and *cherry* would appear with the word *pie* and this shows that the *cherry* is similar to the *strawberry* based on the number of occurrences both words have with the word *pie*. Similarly, *digital* is similar to *information* as it often occurs with the word *data* and *computer*.

By representing the words and documents as vectors and matrices, we can easily make calculations to get to the query, based on similarity and frequency of the words. There are various calculations to measure the similarity, however, these calculations are omitted in this paper. Jurafsky and Martin (2023) provide a good overview of the vector calculations in their book.

Bag of words

Bag of words is a method where all the words in a document are jumbled up, rather than keeping their ordering in the sentence, and we only count the number of times the word occurs in the document. For example, rather than keeping the sentence “I love AI” as an embedding, we would instead count that the word “I” has a frequency of 1, and this goes for every word in the document. By using the bag of words method, we get to classify a document as a certain class, according to the features that we want to have in our model. This is often used with the Naive Bayes classification (Jurafsky & Martin, 2023). Even though we would not discuss this approach in detail, the bag of words is pivotal in the text classification task.

Some challenges can be addressed with this method. Since all the words in the “bag” do not correspond to their original position in the sentence, we would assume that every word has the same connotation, when in reality, every word has a different meaning depending on how they are used. For example, the word “cap”, even though could be generalized as a cover, would mean differently when used in a document that utilizes the modern slang, “No cap!”. This can also be an issue when classifying the documents. For example, suppose that we associate the word “great” as a positive feature, and hence a document that has a high frequency of the word “great” would have a positive feature. However, if the word “great” is used as sarcasm instead in some instances in the document, the system would not recognize that, and this would mean that the document would be classified wrongly.

Most problems however arise from linguistic variation. For example, it cannot handle cases where different words are used for the same semantical meaning. If a query uses the word “spectacles” to refer to the eyewear for example, we would not retrieve the documents that used the word “glasses” or “eyeglasses”, even though they are synonyms. This phenomenon is termed as a lexical variation. Other phenomena would be semantical variations, where it would not recognize words that have multiple interpretations. We can use the “No cap!” example, where if a boomer is trying to search this phrase up to get its meaning, they will end up with the headwear cap instead. There is also an issue when words are represented differently, termed as morphological variation, for example, “girl” and “girl’s”. Syntactical variation is also an issue,

for instance in a document with the sentence, “In the city, the fish population in the river is lower”, this sentence does not talk about the city population.

Embeddings

Instead of representing words as a vector, or by using a “bag” to get their frequency, we can also use embeddings as a way to represent the words. We can see in word vectors, how we would get a vector with sparse values as there could be words that do not correlate to each other at all and will only be filled with a lot of 0s. By using embeddings, our vectors will be dense, since instead of having a dimension $|V|$, the dimensions will be a lot smaller, around 500 to 1000. We would only use words that are neighbors with the current word as a dimension, and add other words from the lexicon to the vectors. We will then use a classifier to learn the difference between the positive words (words that are neighbors to our target words) and the negative words.

We start by looking at sentence (20) below.

(20) `… sugar, a spoonful of strawberry puree, a slice …`

Assume we want our target word to be *strawberry*, and that we would like to look at the 4 closest neighbors of that word. The neighboring words would then be *spoonful, of, puree,* and *a*. These 4 words would now be classified as positive words. We would then shift the words around, and these words would be the negative words since it does not correspond to our current target word. These words would then be in a vector, where all words in the vector are classified as positive or negative words. With a classifier, we train our model to learn the similarity of words in terms of context. For example, with the sentence above, *strawberry* would come in the same context as *puree*, and by using a classifier, we get to identify other fruits that are similar to *strawberry*, such as pineapple.

Semantic Role Labeling (SRL)

From the sections below, we can see how words are classified, either in vector or using a "bag". However, this does not help in answering the user's query. Although we might retrieve documents that contain the same words as the query, it does not necessarily mean that the document will contain the answer. Consider sentence (21) below:

(21) Sam likes Riley

If (20) is considered to be a query, IR would retrieve documents that contain the keywords *Sam*, *likes,* and *Riley*. However, this would mean documents that contain those keywords but do not answer the query would be retrieved as well. Sentences (22) until (26) below are sentences that the current system could not distinguish:

(22) Sam likes Riley
(23) Riley likes Sam, but Sam dislikes Riley
(24) Sam's best friend likes Riley's cousin
(25) Riley and Sam likes chocolate
(26) Sam likes Riley as a friend

To solve this problem, we can assign semantic roles to these keywords. Semantic Role Labeling is one of the methods that can be used in IR systems to answer the user's query. Although this method is still a work in progress, we can expect that this method will lower the number of documents retrieved, making the set of retrieved documents more cognate of the query. We can see that by using semantic roles, we can answer the common WH questions. Consider sentence (27) below:

(27) Corey gifted the graduating seniors a custom-made mug.

From a semantic class, we can learn that *Corey* has the Agent role, *the graduating seniors* have the Recipient role, and *a custom-made mug* would have the Theme role. By using semantic roles, we can answer questions (28) and (29) below:

(28) What did Corey gift the graduating seniors?
(29) Who gifted the graduating seniors a custom-made mug?

We can learn that the Theme role would answer question (28) and the Agent role would answer question (29).

We can see how semantic roles here can help answer the query. As we discussed in the types of questions and answers section, we can easily analyze what the question wants based on the type of question and what roles the question asks for. As we see in (29), it uses the keyword "who", hence we know it is asking for either an Agent or Recipient role, and that it will never ask for the Theme role. Using semantic roles will greatly reduce the number of documents retrieved by the IR system, and ensure that only the relevant documents are returned. This can be easily applied by assigning the roles to the words in the sentence and using heuristics to determine the roles warranted by the query. Figure 4 below shows the set of roles that can be assigned to words or embeddings in the sentences, and Table 5 shows the sets of heuristics that can be implemented in the IR system.

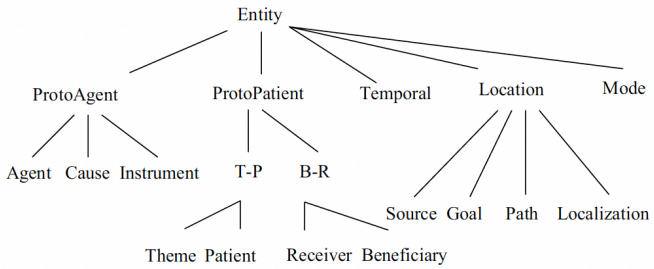

Figure 4: Set of roles that can be assigned (Moreda et al., 2005)

Question	Role		No role
Where In where In what + exp At what + exp	Location		ProtoAgent Mode Temporal Cause ProtoPatient
When In what + exp What + exp	Temporal		ProtoAgent Mode Location Cause ProtoPatient
How	Mode Theme (if it is a diction verb)		ProtoAgent Location Temporal Cause Patient Beneficiary
Who	Agent - ProtoAgent Patient - ProtoPatient		Mode Temporal Location Theme Beneficiary
What	Cause Theme		
Whose	Receiver Beneficiary Patient	ProtoPatient	Agent Location Mode Temporal Theme Cause

Table 5: Set of heuristics. Adapted from Moreda et al. (2005)

Conclusion

From this paper, we get to see a few of the semantic features that are relevant to the QA and IR systems. Even though there are other ways to implement this system, for example by using NLP, or by training our model, as we see in the embeddings example, we cannot solely work on calculations alone as language is not based on numbers, but rather a set of rules that cannot be hardcoded easily. Language is unpredictable, hence by using semantic features, we can ensure that we would output sentences that make sense, and are true to the real world. In the future, even if we get to implement a system that is only dependent on calculations, we will see some of the rules or heuristics that have semantic features in it, especially if our system deals with

language specifically. This shows how important semantics is especially in our current world where we use a lot of generative Artificial Intelligence (AI) and the web to get information. Without semantics, we can imagine how our search experience would be, our results would be all jumbled up and we probably would not understand the answers given by the search engine. Hence, semantics is vital in any field that utilizes language, especially if we are dealing with technology or computers that need to learn about the language.

References

Arampatzis, A., Tsoris, T., Koster, C. H. A., & Van Der Weide, T. (1998). Phrase-based information retrieval. *Information Processing and Management*, *34*(6), 693–707. https://www.informatik.uni-trier.de/~ley/db/journals/ipm/ipm34.html

Jurafsky, D., & Martin, J. H. (2023). *Speech and Language processing. An introduction to natural language processing, computational linguistics, and speech recognition*.

Miller, G. A. (1995). WordNet. *Communications of the ACM*, *38*(11), 39–41. https://doi.org/10.1145/219717.219748

Montoyo, A., Munoz, R., & Métais, E. (2005). *Natural language processing and information systems: 10th International Conference on Applications of Natural Language to Information Systems, NLDB 2005, Alicante, Spain, June 15-17, Proceedings*. Springer.

Moreda, P., Navarro, B., & Palomar, M. (2005). Using semantic roles in information retrieval systems. In *Lecture notes in computer science* (pp. 192–202). https://doi.org/10.1007/11428817_18

Prager, J. (2006). Open-Domain Question–Answering. *Foundations and Trends in Information Retrieval*, *1*(2), 91–231. https://doi.org/10.1561/1500000001

Punyakanok, V., Roth, D., & Yih, W. (2004). Natural Language inference via dependency Tree Mapping: an application to question answering. *Computational Linguistics*. https://www.ideals.illinois.edu/handle/2142/11100

Soares, M. A., & Parreiras, F. S. (2020). A literature review on question answering techniques, paradigms and systems. *Journal of King Saud University. Computer and Information Sciences/Mağalaẗ Ğam'aẗ Al-malīk Saud : Ùlm Al-ḥasib Wa Al-ma'lumat*, *32*(6), 635–646. https://doi.org/10.1016/j.jksuci.2018.08.005

Toral, A., Noguera, E., Llopis, F., & Muñoz, R. (2005). Improving question answering using named entity recognition. In *Lecture notes in computer science* (pp. 181–191). https://doi.org/10.1007/11428817_17