WAMBy: An information retrieval approach to web-based question answering

Christos Samarinas
Aristotle University of Thessaloniki
chris.samarinas@gmail.com

Grigorios Tsoumakas
Aristotle University of Thessaloniki
greg@csd.auth.gr

ABSTRACT

Nowadays, most answers to natural language questions can be found in the first few results of web search engines. We describe the WAMBy question answering system that attempts to extract answers from the top 10 results of Google. We propose a new ranking method for factoid answers that among others takes into account the semantic similarity of the context of each answer candidate with the question and named entities found in the titles and the snippets of search results. The proposed method gave very promising results in a variety of questions. Also we describe the methods used for non-factoid answer extraction and ranking. An important part of the system is a new text similarity measure that extends TF-IDF by utilizing word vectors. The new text similarity measure solves the problem of synonyms and improves the performance of TF-IDF in the paraphrase identification task. The source code of WAMBy is publicly available as well as two datasets that were created for question classification and factoid question answering evaluation.

CCS CONCEPTS

•Information systems →Retrieval models and ranking; Question answering; *Data extraction and integration;*

KEYWORDS

Question Answering, Information Retrieval

1 INTRODUCTION

The last few years, question answering has received a lot of attention thanks to the appearance of personal assistant applications and hardware. The firsts attempts focused on developing domain specific question answering systems (BASEBALL [6] in 1961 and LUNAR [23] in 1977). Later, we saw the development of information retrieval based open domain systems. In Textract [22] system they investigated the role of information extraction in question answering. The question classification was based on rules extracted with syntactic analysis from a set of questions and a smaller set of handcrafted rules. In [24] there was an attempt to improve question answering by using and extending lexical semantic sources like Wordnet. Another system, GuruQA [20], used predictive annotation. The question classification was based on predefined rules. In [21] a probabilistic translation model was created with one million question and answer pairs for non-factoid question answering. In NSIR [19] a probabilistic method was used for re-ranking the answer candidates based on the expected answer type.

There have been many attempts to create web-based question answering systems that utilize web search engines to retrieve documents for answer extraction. One of these was the AskMSR [2] system that used answer patterns to extract factoid answers from the snippets of web search results. A similar approach can be seen in [4] and in the WSQA [3] system that also used question and answer patterns and information from snippets. In WAG [15] system the named entities received scores based on their frequency and based on these scores paragraphs and answers were extracted.

The only open-source question answering system, YodaQA [1], is knowledge-based and extracts answers from structured databases like DBPedia and Freebase.

One of the most developed systems is START [1]. It consists of a text analysis module that creates a knowledge base and another module that generates sentences. These two together with a natural language annotation technique extract answers from many websites. The most recent progress in web-based open domain question answering was made by Google with featured snippets; chunks of text on top of the search results that answer the user's question. From granted patents [16] [25] it seems that a trained model is used for the extraction of possible answer snippets. Then many techniques are used like POS tagging, named entity recognition and a knowledge base is created that connects entities. Probably, the most complex question answering system is the Watson [2] system by IBM. It uses more than 100 different techniques for natural language analysis, finding sources and extracting answers. The main innovation of the system was its ability to run many different algorithms at once to find the most suitable answer.

Most question answering systems that have been proposed are either too limited and rely on question and answer patterns, or too complex and need to run on many computers. In this paper, we describe the architecture of the WAMBy (Web-based Answer Mining Bot) question answering system. WAMBy uses an information retrieval approach to rank answers from the top 10 results of Google and can run in real-time on a single computer giving very promising results especially in factoid questions. The system can be relatively easily modified to work in any language and its source code is publicly available as well as two datasets for question clasification and factoid question answering evaluation.

SETN'18, Rio Patras, Greece
 978-1-4503-6433-1/18/07...$15.00
DOI: 10.1145/3200947.3201023

[1] http://start.csail.mit.edu
[2] http://www.research.ibm.com/deepqa/deepqa.shtml

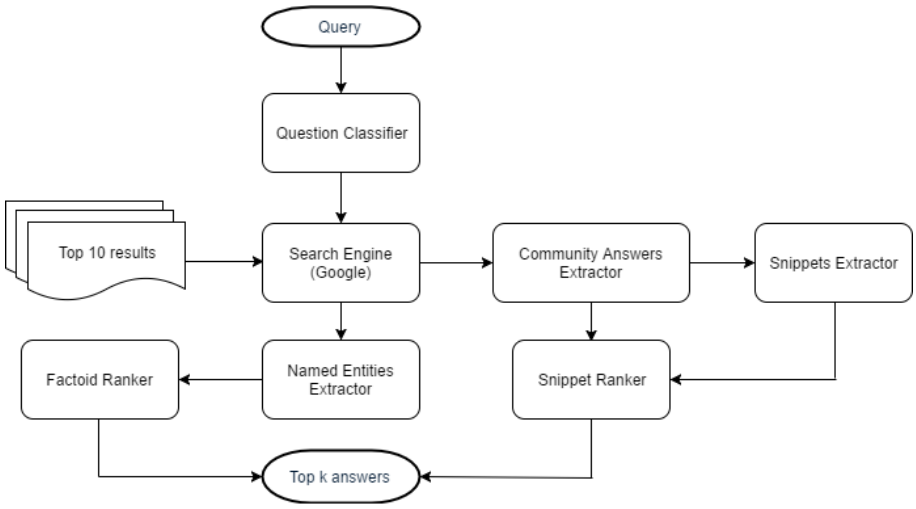

Figure 1: The architecture of WAMBy system.

2 SYSTEM ARCHITECTURE

Figure 1 depicts the architecture of the developed question answering system. The extraction pipeline of answer candidates consists of 4 or 5 steps. In the first step we classify the question to determine the expected answer type; factoid (person, location, number, date etc.) or non-factoid (description). Then we retrieve the first 10 results from Google for the given question. We remove the boilerplate from the downloaded web pages and proceed with the extraction process. If the expected answer type is factoid, we extract named entities, rank them and return the top k with the highest score. For non-factoid answers we first extract possible answer chunks, we rank them and then return the top k.

3 QUESTION CLASSIFICATION

One of the most important parts of a question answering system is the question classification module. Our system recognizes 11 types of questions: date, duration, location, money, ordinal, organization, percent, person, set, time and description (non-factoid). For classification we used a hybrid model that uses a set of predefined rules in the first stage and a recurrent neural network classifier in the second stage to cover more complex question forms.

3.1 Classification rules

The rules used in the first stage can be seen in Table 1.

Question starts with	Question Type
WHO + To Be Verb + PERSON	DESCRIPTION
WHO	PERSON
WHEN	DATE
WHERE	LOCATION
WHY	DESCRIPTION
WHAT + To Be Verb + Noun	DESCRIPTION
Modal Verb	DESCRIPTION
To Be Verb	DESCRIPTION
HOW + Verb	DESCRIPTION

Table 1: Simple classification rules

3.2 LSTM classifier

3.2.1 The dataset. The only available dataset for question classification [3] [12] had 5500 questions (from TREC tracks) with their type. That was far from enough to train a model that can handle many forms of questions. That is why we created a new one from the SQUAD dataset [4]. SQUAD is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles where the answer to every question is a segment of text from the corresponding reading passage. However the answers do not have any label for their type. We assigned a type to every answer using the Stanford Named Entity Recognizer [13]. The new dataset has 40,197 question - answer type pairs. This method for determining the type of the answer is not always accurate thus some assigned labels are incorrect. Questions that are covered with rules are removed from the final dataset.

3.2.2 The model. The cases not covered in the first stage are classified by a Long Short Term Memory Network [7] with one hidden recurrent layer of 20 memory units and a softmax output layer as seen in Figure 2. At each time step we give as input the vector of each word of the question (Figure 3). We used pretrained word vectors with the GloVe model [17] on the Wikipedia 2014 and Gigaword 5 datasets. We trained the model with 50d, 100d and 300d vectors and did not notice any difference in accuracy, so we chose the 50d vectors. We used the ADAM optimizer, batch size 25 and learning rate 0.15 and trained the network for 150 epochs.

In Table 2 we can see the performance of the network on the two datasets with the questions covered by rules removed and with all the questions. If we test the final question classifier that uses both

[3] http://cogcomp.cs.illinois.edu/Data/QA/QC
[4] https://rajpurkar.github.io/SQuAD-explorer

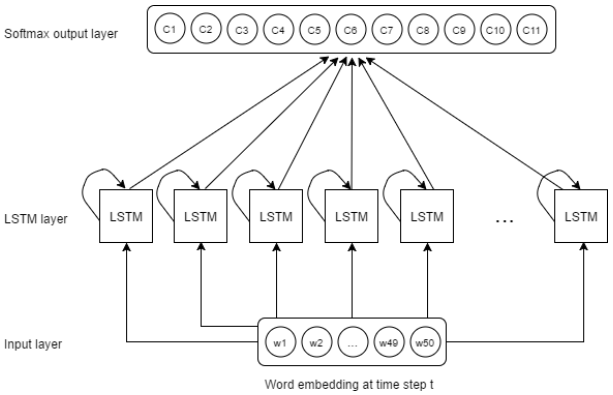

Figure 2: LSTM network question classifier.

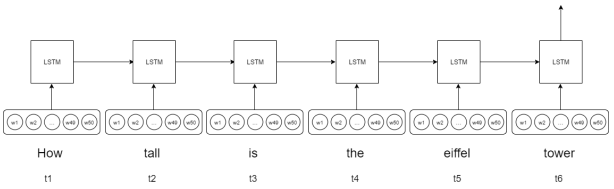

Figure 3: Unfolded LSTM unit in time.

Dataset	Accuracy	Precision	Recall	F1
TREC	59.33%	59.92%	44.27%	50.91%
TREC (all questions)	77.03%	77.72%	58.23%	66.58%
SQUAD	48.78%	42.21%	29.81%	34.94%
SQUAD (all questions)	68.24%	68.67%	55.45%	61.35%

Table 2: Performance of LSTM question classifier

the rules and the LSTM network we get **78.8%** accuracy in the two datasets combined.

4 TEXT SIMILARITY MEASURES

The proposed answer ranking method uses some text similarity measures for semantic relatedness and common words order. For semantic relatedness we propose a new extension of TF-IDF that utilizes word embeddings to solve the problem of synonyms. We tested the new text similarity measure with other TF-IDF variants in Microsoft Research Paraphrase Corpus [5] and noticed an improvement in paraphrase identification with our variant.

4.1 An extension of TF-IDF

In TF-IDF each text (or document) is represented as a vector of TF-IDF weights. Each weight is proportional to the importance (term frequency) of each word in the text:

$$tf(w,d) = |\{t \in d : t = w\}|$$

[5]https://www.microsoft.com/en-us/download/details.aspx?id=52398

The inverse document frequency (IDF) lowers the weight of words that are common in many texts and is usually defined as:

$$idf(w,D) = log\frac{|D|}{|\{d \in D : w \in d\}|}$$

where D is the collection of all documents.
The similarity of two texts is calculated by taking the cosine similarity of their TF-IDF vectors:

$$sim(q,d) = \frac{q^T \cdot d}{|q||d|} = \\ = \frac{\sum_{w \in W(q) \cap W(d)} (tf(w,q) \cdot idf(w))(tf(w,d) \cdot idf(w))}{\sqrt{\sum_{w \in W(q)} (tf(w,q) \cdot idf(w))^2 \sum_{w \in W(d)} (tf(w,d) \cdot idf(w))^2}}$$

where $W(q), W(d)$ the sets of words of q and d. The main problem with this approach is that it calculates similarity based on exact matches of words between the two texts. Synonyms and closely related words are not considered. There have been some attempts to extend TF-IDF with the use of word embeddings. The first similarity measure was proposed in [14]:

$$sim(q,d) = \frac{\sum_{w \in W(q)} \max_{u \in W(d)} cos(w,u) \cdot idf(w)}{2 \sum_{w \in W(q)} idf(w)} + \\ + \frac{\sum_{w \in W(d)} \max_{u \in W(q)} cos(w,u) \cdot idf(w)}{2 \sum_{w \in W(d)} idf(w)} \quad (1)$$

This measure does not consider the term frequencies and gives high similarity score even when the two texts are not similar at all. It was proposed mainly for short texts. Another measure was proposed in [9] and is a variation of the BM25 formula where the term frequency is replaced by the maximum cosine similarity of the word with the words in the query q:

$$sim(q,d) = \sum_{w \in W(d)} \frac{idf(w) \cdot \max_{u \in W(q)} cos(w,u) \cdot (k_1 + 1)}{\max_{u \in W(q)} cos(w,u) + k_1 \cdot (1 - b + b \cdot \frac{|q|}{\text{avg length of d}})} \quad (2)$$

This measure also does not consider the term frequencies and has two parameters, b and k_1, that must be properly defined. Another measure proposed in [5] uses the similarity matrix S between words and calculates cosine similarity as follows:

$$sim(q,d) = \frac{q^T S d}{|q^T S| \cdot |d|} \quad (3)$$

In their approach they used a word similarity measure based on Wordnet. However, because we want to find a measure that can

be easily used in any language, we replaced in our tests the Wordnet based word similarity with the cosine similarity of the word vectors. The last similarity measure we tested, calculates the cosine similarity of the average word embeddings of the two texts. Instead of just taking the average word embedding for each text, we can take a weighted average, where each weight is the TF-IDF weight of each word. With this change we can capture the distributional characteristics of words:

$$v_q = \frac{\sum_{w \in W(q)} tf(w,q) \cdot idf(w) \cdot vector(w)}{\sum_{w \in W(q)} (tf(w,q) \cdot idf(w))} \tag{4}$$

We propose a new text similarity measure. The basic idea is that we define the term frequency (TF) differently. If a word is not present in one text, then its frequency is defined as a percentage of the frequency of the closest word. The percentage is the maximum cosine similarity of the word embedding of the non existent word with the words of the text. We also set a synonymy threshold k so that only closely related words contribute to the similarity score:

$$sim(q,d) = \frac{sim_p(q,d) + sim_p(d,q)}{2} \tag{5}$$

$$sim_p(q,d) = \frac{\sum_{w \in W(q)} (tf(w,q) \cdot idf(w))(tf_s(w,d) \cdot idf(w))}{\sqrt{\sum_{w \in W(q)} (tf(w,q) \cdot idf(w))^2 \sum_{w \in W(q)} (tf_s(w,d) \cdot idf(w))^2}}$$

$$tf_s(w,d) = \begin{cases} tf(w,d) & w \in W(d) \\ \max_{u \in W(d)} cos(w,u) \cdot (tf(u,d)) & w \notin W(d), \max_{u \in W(d)} cos(w,u) > k \\ 0 & w \notin W(d), \max_{u \in W(d)} cos(w,u) \le k \end{cases}$$

After experiments, we found that a good value for the synonymy threshold k is 0.75. We compared our proposed measure with the other four in the paraphrase identification task (Table 3) after defining the optimal paraphrase threshold for each of them. The new measure performed better than most proposed unsupervised models in this task [6] achieving 81.8% F1 score.

4.2 Common words order similarity

In some parts of our answer ranking method we use a similarity measure for the order of common words between two texts [8]:

$$sim_o(q,d) = \begin{cases} 1 - \frac{2\sum_{i=1}^{\delta} |x_i - y_i|}{\delta^2} & \text{if } \delta \text{ is even} \\ 1 - \frac{2\sum_{i=1}^{\delta} |x_i - y_i|}{\delta^2 - 1} & \text{if } \delta \text{ is odd and } \delta > 1 \\ 1 & \text{if } \delta = 1 \end{cases} \tag{6}$$

Let C be the common words (δ in total) of two texts q and d. x_i are the positions of the words of q. Each y_i is the position of x_i-th word from q in d.

4.3 n-grams similarity

The last measure used in our ranking indicates the percentage of common n-grams between two texts. It is based on the measure proposed in [10]:

$$ngsim(q,d) = \sum_{n=2}^{4} w_n \frac{|ngrams(q,n) \cap ngrams(d,n)|}{|ngrams(q,n)|} \tag{7}$$

$$\sum_{n=2}^{4} w_n = 1$$

We count the percentages of 2-grams, 3-grams and 4-grams of the shorter text q found in the longer text d. We give greater weights to longer n-grams so that longer common sequences contribute more to the similarity score. In our system we used the following weights: $w_2 = 0.14, w_3 = 0.28, w_4 = 0.58$.

5 ANSWERS EXTRACTION

5.1 Boilerplate removal

All websites contain, apart from their main content, many other elements like navigation links and advertisements. These elements make up the boilerplate, which does not contain useful information for the answer extraction process. In fact, if we do not remove the boilerplate, we may select incorrect answers. WAMBy removes boilerplate with Algorithm 1, which is quite similar to *jusText*[7] [18], with the only difference being that we do not check the stopwords density. The algorithm iterates through all the blocks (div, p, h1-6, ...) of the given web page and classifies each block as content if it has link density below a threshold and number of words above a threshold. Otherwise, the block is classified as boilerplate. The link density of a block is the percentage of words that are inside anchor tags (links). In our system we used maxLinksDensity=0.9 and minWordsNumber=2. Figure 4 presents the output of the algorithm for a Wikipedia article.

5.2 Extracting factoid answers

After removing the boilerplate from the web pages, we use the Named Entity Recognizer of the Stanford CoreNLP library [13] to extract named entities of the expected type. The next step is the ranking and merging of the factoid answers.

5.3 Ranking factoid answers

The ranking of factoid answers is done in three steps. In the first step we give an initial score based on four measures: context score, title score, web position score and n-grams score. If F is the group of factoids with the same value f then:

[6] https://aclweb.org/aclwiki/index.php?title=Paraphrase_Identification_(State_of_the_art)

[7] http://corpus.tools/wiki/Justext

Model	Accuracy	Precision	Recall	F1
TF-IDF	71.07%	**76.49%**	81.5%	78.91%
TF-IDF variant (Rada Mihalcea et. al 2006) (1)	63.76%	70.79%	77.39%	73.94%
BM25 variant (Tom Kenter et. al 2015) (2)	70.14%	71.41%	91.79%	80.33%
Our TF-IDF variant (5)	**72.63%**	73.27%	92.58%	**81.8%**
Matrix Similarity (Samuel Fernando et. al 2008) (3)	67.01%	67.51%	97.03%	79.62%
Average word vectors (4)	66.43%	66.43%	**100%**	79.83%

Table 3: Comparison of unsupervised text similarity models

Algorithm 1 Boilerplate removal algorithm

```
String blockText = "";
int linkedWords = 0;
for each htmlTag t do
    if t.type ∈ {div, p, h1, h2, ..., h6, tr, ul, ol} then
        if blockText not empty then
            Block b = new Block(blockText);
            linkDensity = b.linkedWords/b.wordsCount;
            if b.linkDensity < maxLinksDensity then
                if b.wordsCount > MinWordsNumber then
                    content = content ∪ {b}
                else
                    boilerplate = boilerplate ∪ {b}
            blockText = "";
            linkedWords = 0;
    else if t.type = a and t is opening tag then
        linkedWords += t.wordsCount;
    else if t.type = text and t is opening tag then
        blockText += t.text;
```

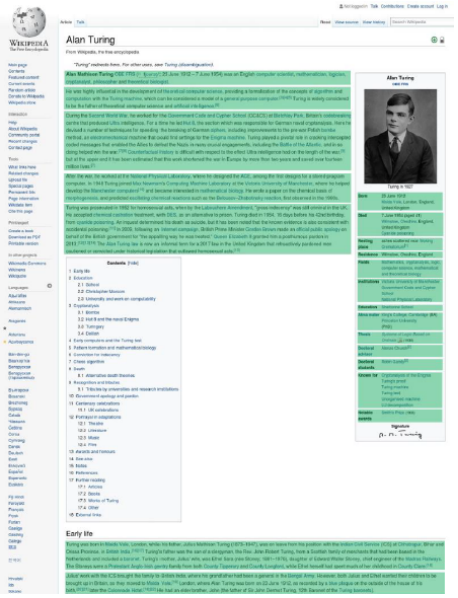

Figure 4: Sample output of the boilerplate removal algorithm. Retained content is highlighted in green.

$$\text{context-score}(f) = h_1(f) = \max_{f' \in F} \left(\frac{\sum_{w \in C(f')} fs(w) \cdot idf(w)/dist(w, f')}{\sum_{w \in C(f')} idf(w)/dist(w, f')} \right) \quad (8)$$

$$fs(w) = \max\left(maxSim(w, query), maxSim(w, features)\right) \quad (9)$$

$$maxSim(w, S) = \max_{s \in W(S)} cos(w, s) \quad (10)$$

$$dist(w, f') = |\{v \in C(f') : v \text{ between } w \text{ and } f'\}| + 1 \quad (11)$$

$C(f')$ is a window of words left and right to f'. In our system we use a window of 200, that is 100 words left and 100 words right. $query$ is the question posed by the user and $features$ is a transformation of the query that keeps only verbs, nouns and numbers and adds keywords based on the query (using rules). For instance if the query starts with the words 'how tall' then we add to the features the words: height, m, metre, ft, foot, in, inch. We use the features as an alternative to a more complete named entity recognizer. $maxSim(w, S)$ calculates the maximum cosine similarity of the word embedding w with the set of word embeddings S. The logic behind the context score is that we give high scores to answer candidates with context relevant to the question. Words that are closer to the factoid in the context window contribute more to the context score. For example a word right next to the answer (with zero distance) has weight 1, while a word that has a distance of 3 words from the answer has weight 0.25. The title score calculates the similarity of the question with the title of the website containing each answer candidate:

$$\text{title-score}(f) = h_2(f) = \max_{f' \in F} \left(w_1 \cdot sim(query, title(f')) + w_2 \cdot sim_o(query, title(f'))\right) \quad (12)$$

$$w_1 + w_2 = 1$$

where sim is (5), sim_o is (6), $title(f')$ is the title of the web page containing the factoid answer f' and w_1, w_2 are weights to be defined. In our system we used $w_1 = 0.8$ and $w_2 = 0.2$. The web

position score gives a number between 0 and 1 based on the position of the website that contains each answer candidate in the search engine results:

$$\text{web-position-score}(f) = h_3(f) = \max_{f' \in F} \left(1 - \frac{webPosition(f') - 1}{totalResults}\right) \tag{13}$$

where *totalResults* is the total number of web results processed by the system (10 in our implementation) and $webPosition(f')$ the position of the web page that contains the factoid f' inside the web results (from 1 to *totalResults*). The ngrams score calculates the ngrams similarity between the question and the context of each factoid answer candidate:

$$\text{ngrams-score}(f) = h_4(f) = \max_{f' \in F} (ngsim(query, C(f'))) \tag{14}$$

where *ngsim* is (7). The final score of a factoid in the first step is a weighted average of the four measures:

$$\text{f-score}(f) = \sum_{i=1}^{4} w_i \cdot h_i(f) \tag{15}$$

$$\sum_{i=1}^{4} w_i = 1$$

In our system we have $w_1 = 0.5$, $w_2 = w_3 = 0.2$ and $w_4 = 0.1$. Every group of factoids F with the same value f is represented once with the above score in the merged factoids set F_M. This set contains the top k best merged factoids (k=100 in our system). If the expected answer type is person, organization, location or date, we have a second step of re-ranking. In this step every factoid gets a new score based on its similarity with all the other factoids. This helps when we have many factoids that refer to the same entity. The most common example is names that can appear as full names, like Einstein and Albert Einstein. Another example is when the answers must be cities in a specific country. The new score is calculated as follows:

$$\text{f-score}(f) = w_1 \cdot \text{f-score}(f) + w_2 \cdot \sum_{f' \in F_M - \{f\}} \frac{sim(f, f') \cdot \text{f-score}(f')}{|F_M - \{f\}|} \tag{16}$$

$$w_1 + w_2 = 1$$

All the new scores are calculated using the old ones only. In the final step we have re-ranking based on the factoids found on the titles and snippets of the search results. At first we extract factoids from titles and snippets and rank them as follows:

$$\text{f-score-r}(f) = w_1 \cdot h_1(f) + w_2 \cdot h_3(f) + w_3 \cdot \text{results-score}(f) \tag{17}$$

$$\text{results-score}(f) = \frac{|\{r \in R : f \in W(r)\}|}{|R|} \tag{18}$$

$$\sum_{i=1}^{3} w_i = 1$$

where R is the set of web results processed by the system and $W(r)$ the set of words of the web page r. In our implementation we have $w_1 = 0.25$, $w_2 = 0.45$ and $w_3 = 0.3$. The final score is:

$$\text{f-score-final}(f) = s(f) \cdot \text{f-score}(f) \cdot (w_1 + w_2 \cdot (w_3 \cdot (\text{f-score-r}(f) \text{ in } T) + w_4 \cdot (\text{f-score-r}(f) \text{ in } S))) \tag{19}$$

$$s(f) = \begin{cases} 1 & ic(f) \geq e \\ 0 & ic(f) < e \end{cases} \tag{20}$$

$$ic(f) = \frac{1}{|W(f)|} \sum_{w \in W(f)} 1 - maxSim(w, query) \tag{21}$$

$$w_1 + w_2 = w_3 + w_4 = 1$$

where T the set of factoids found in titles, S the set of factoids found in snippets and $W(f)$ the set of words of the factoid f. The factor $s(f)$ becomes 0 when a factoid has low information content $ic(f)$, that is it contains words that already exist in the question (query) or closely related words to the words in the question. In our system we defined $w_1 = w_2 = w_3 = w_4 = 0.5$ and $e = 0.05$.

5.4 Extracting non-factoid answers

When the expected answer type is description (non-factoid), we first try to detect answers from online communities. Most community question answering websites, use the type attribute with value 'schema.org/Answer' to determine the block that contains an answer. We search for this type of blocks. If at least one is found, we rank only these blocks. If not, we follow a simple process to extract chunks of text (snippets). The snippets extraction process is done in two steps. In the first step we select a set S of sentences from each website that have a similarity above a threshold with the question:

$$S = \{s : sim(query, s) > k\} \tag{22}$$

where $k = 0.1$ in our implementation. In the second step, we iterate through the sentences in order of their appearance in each web page and cluster the sentences that have a distance bellow a threshold. The distance considers the similarity of the sentences as well as the number of words between them. The distance is defined as follows:

$$sd(s_1, s_2) = \sum_{s \in S(s_1, s_2)} \frac{|\{w : w \in W(s)\}|}{\max(sim(s_1, s), sim(s_2, s)) + b} \tag{23}$$

where $S(s_1, s_2)$ is the set of sentences between s_1 and s_2 and $W(s)$ the set of words of the sentence s. $b = 1$ in our case. When $sd(s_1, s_2) < k$ ($k = 50$ in our system) the sentences s_1, s_2 and the ones between them are clustered in one answer candidate.

5.5 Ranking non-factoid answers

The ranking of the non-factoid answers is done using (24):

$$\text{s-score}(s) = \sum_{i=1}^{5} w_i \cdot g_i(s) \tag{24}$$

$$\text{relevance-score}(s) = g_1(s) = \frac{\sum_{t \in s} sim(query, t)}{\max_{s' \in S} \left(\sum_{t \in s'} sim(query, t) \right)} \quad (25)$$

$$\text{title-similarity}(s) = g_2(s) = sim(query, title(s)) \quad (26)$$

$$\text{max-sentence-similarity}(s) = g_3(s) = \max_{s' \in s} (sim(query, s')) \quad (27)$$

$$\text{web-position-score}(s) = g_4(s) = 1 - \frac{webPosition(s) - 1}{totalResults} \quad (28)$$

$$\text{ngrams-score}(s) = g_5(s) = ngsim(query, s) \quad (29)$$

$$\sum_{i=1}^{5} w_i = 1$$

where s is the set of sentences of the non-factoid answer candidate, S the set of all non-factoid answer candidates, $title(s)$ the title of the web page that contains s. In our system we used $w_1 = 0.25$, $w_2 = 0.3$, $w_3 = 0.2$, $w_4 = 0.2$ and $w_5 = 0.05$.

6 EVALUATION

The evaluation of WAMBy was done only with factoid questions. We created a dataset of 120 questions and answers, 30 from each of the following categories: person, date, location, number. We used the Mean Reciprocal Rank (MRR) for the top 20 ranked answers. MRR gives values from 0 to 1. The closer it is to 1 the higher the correct answer is ranked. For example if the correct answer to a question is ranked 5th, then this question gets 0.2 score. We calculate the score for all the questions and take the average:

$$\text{MRR} = \frac{\sum_{i=1}^{N} \frac{1}{rank_i}}{N} \quad (30)$$

where N is the number of questions and $rank_i$ the position of the correct answer to the i-th question. The results can be seen bellow:

Question Type	MRR	Correct answers
PERSON	0.934	27/30
DATE	0.739	19/30
LOCATION	0.809	21/30
NUMBER	0.674	16/30
ALL	0.789	83/120

Table 5: Evaluation of factoid answers

We observe that the number category has the lowest MRR. We could greatly improve the performance in this category by extending the named entity recognizer so that it recognizes many different types of numbers like height, weight and distance. In the date category we can have an improvement if we remove answers like: today, now, tomorrow, that are unlikely to be the correct answer. In Table 4 we can see some examples of factoid questions that have the correct answer ranked first.

7 CONCLUSIONS

We developed a web-based open domain question answering system that extracts answers from the top 10 results of Google. The system gave very promising results in factoid answers, especially in person category achieving 0.934 MRR in our test dataset. In the other categories it can reach higher accuracy by extending the named entity recognizer. In non-factoid questions, it can provide relevant answers from community question answering websites. An important result of our work is also a new way to calculate the text similarity between two texts that extends TF-IDF by utilizing word embeddings. The proposed measure improves the performance of TF-IDF in the paraphrase identification task achieving 81.8% F1 score, higher than almost all the proposed unsupervised models. The source code of the developed system is publicly available as well as the two datasets that were created for question classification and factoid question answering evaluation.

8 FUTURE WORK

As we already mentioned, the first improvement can be done by extending the named entity recognizer so that it detects other types of factoids like height, weight and distance. Of course this would also require to train the question classifier with more question types. Our ranking method uses weights for different measures that were defined experimentally. After creating a larger test set of questions, we could automatically learn these weights with a logistic regression model. We could further improve the accuracy of our system by using knowledge bases like Wikidata or DBPedia. However, these sources contain information mainly in English. The final answer selection, after getting a ranking of 20 questions, could be done using a recurrent neural model like a dynamic memory network [11]. The network could be trained on the SQUAD dataset. Each time for the final answer selection (out of the top 20), we would feed the sentences that contain the answers (maybe with some more sentences before and after the ones that contain the answer) and the network would output the final answer.

REFERENCES

[1] Petr Baudiš and Jan Šedivy. 2015. Modeling of the question answering task in the yodaqa system. *Experimental IR Meets Multilinguality, Multimodality, and Interaction* (2015), 222–228.

[2] Eric Brill, Susan Dumais, and Michele Banko. 2002. An Analysis of the AskMSR Question-answering System. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10 (EMNLP '02)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 257–264. DOI: http://dx.doi.org/10.3115/1118693.1118726

[3] Silviu Cucerzan and Eugene Agichtein. 2005. Factoid Question Answering over Unstructured and Structured Web Content.. In *TREC*, Vol. 72. 90.

[4] Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better?. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 291–298.

Question	Answer
Who is the inventor of alternating current	Nikola Tesla
Who was the director of Inception	Christopher Nolan
Who painted The Starry Night	Van Gogh
Who is the father of geometry	Euclid
When was Google founded	1998
When was Isaac Newton born	4 January 1643
When was Apollo 11 launched	July 16, 1969
When did The Great Depression start	1929
Where is the Eiffel Tower	Paris
What is the second largest city in Greece	Thessaloniki
Where is the highest point on earth	Mount Everest
The most populated country in the world	China
How many legs does a beetle have	six
How many colors do you need to color a planar graph	four
How many children did Shakespeare have	three
How many hearts does a worm have	five

Table 4: Examples of correct factoid answers

[5] Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*. 45–52.

[6] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. 1961. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*. ACM, 219–224.

[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[8] Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 2, 2 (2008), 10.

[9] Tom Kenter and Maarten de Rijke. 2015. Short Text Similarity with Word Embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15)*. ACM, New York, NY, USA, 1411–1420. DOI:http://dx.doi.org/10.1145/2806416.2806475

[10] Grzegorz Kondrak. 2005. N-gram Similarity and Distance. In *Proceedings of the 12th International Conference on String Processing and Information Retrieval (SPIRE'05)*. Springer-Verlag, Berlin, Heidelberg, 115–126. DOI:http://dx.doi.org/10.1007/11575832_13

[11] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.), Vol. 48. PMLR, New York, New York, USA, 1378–1387. http://proceedings.mlr.press/v48/kumar16.html

[12] Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 1–7.

[13] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Prismatic Inc, Steven J. Bethard, and David Mcclosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 55–60.

[14] Rada Mihalcea, Courtney Corley, Carlo Strapparava, and others. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, Vol. 6. 775–780.

[15] Günter Neumann and Feiyu Xu. 2004. Mining natural language answers from the web. *Web Intelligence and Agent Systems: An International Journal* 2, 2 (2004), 123–135.

[16] J. Oh, K. Torisawa, C. Hashimoto, T. Kawada, S. De Saeger, J. Kazama, and Y. Wang. 2015. Non-factoid question-answering system and computer program. (Jan. 22 2015). https://www.google.com/patents/US20150026106 US Patent App. 14/377,999.

[17] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[18] Jan Pomikálek. 2011. *Removing Boilerplate and Duplicate Content from Web Corpora*. Ph.D. Dissertation. https://is.muni.cz/th/45523/fi_d/phdthesis.pdf

[19] Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. 2002. Probabilistic Question Answering on the Web. In *Proceedings of the 11th International Conference on World Wide Web (WWW '02)*. ACM, New York, NY, USA, 408–419. DOI:http://dx.doi.org/10.1145/511446.511500

[20] Dragomir R. Radev, John Prager, and Valerie Samn. 2000. Ranking Suspected Answers to Natural Language Questions Using Predictive Annotation. In *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLC '00)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 150–157. DOI:http://dx.doi.org/10.3115/974147.974168

[21] Radu Soricut and Eric Brill. 2006. Automatic Question Answering Using the Web: Beyond the Factoid. *Inf. Retr.* 9, 2 (March 2006), 191–206. DOI:http://dx.doi.org/10.1007/s10791-006-7149-y

[22] Rohini Srihari and Wei Li. 2000. A Question Answering System Supported by Information Extraction. In *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLC '00)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 166–172. DOI:http://dx.doi.org/10.3115/974147.974170

[23] William A Woods and R Kaplan. 1977. Lunar rocks in natural English: Explorations in natural language question answering. *Linguistic structures processing* 5 (1977), 521–569.

[24] Scott Wen-tau Yih, Ming-Wei Chang, Chris Meek, and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models. ACL – Association for Computational Linguistics. https://www.microsoft.com/en-us/research/publication/question-answering-using-enhanced-lexical-semantic-models/

[25] L. ZOU, T. Liu, Y. LU, H. Liu, and R. HUANG. 2017. Natural language question answering method and apparatus. (Jan. 18 2017). https://encrypted.google.com/patents/EP3117345A1?cl=en EP Patent App. EP20,150,761,024.