2019

WikiHow QA

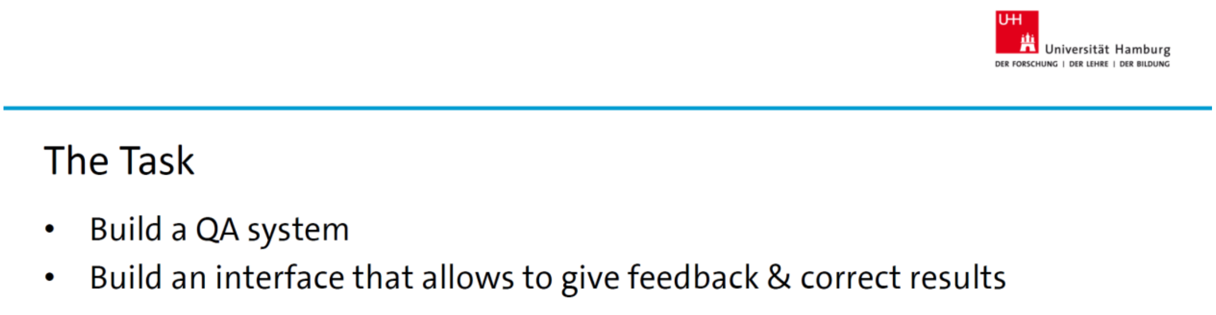
PROJECT REPORT

BY TIM FISCHER

GitHub:	https://github.com/bigabig/wikihow-qa/
Live Demo:	http://ltdemos.informatik.uni-hamburg.de/wikihowqa/

Table Of Contents

The Task


Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

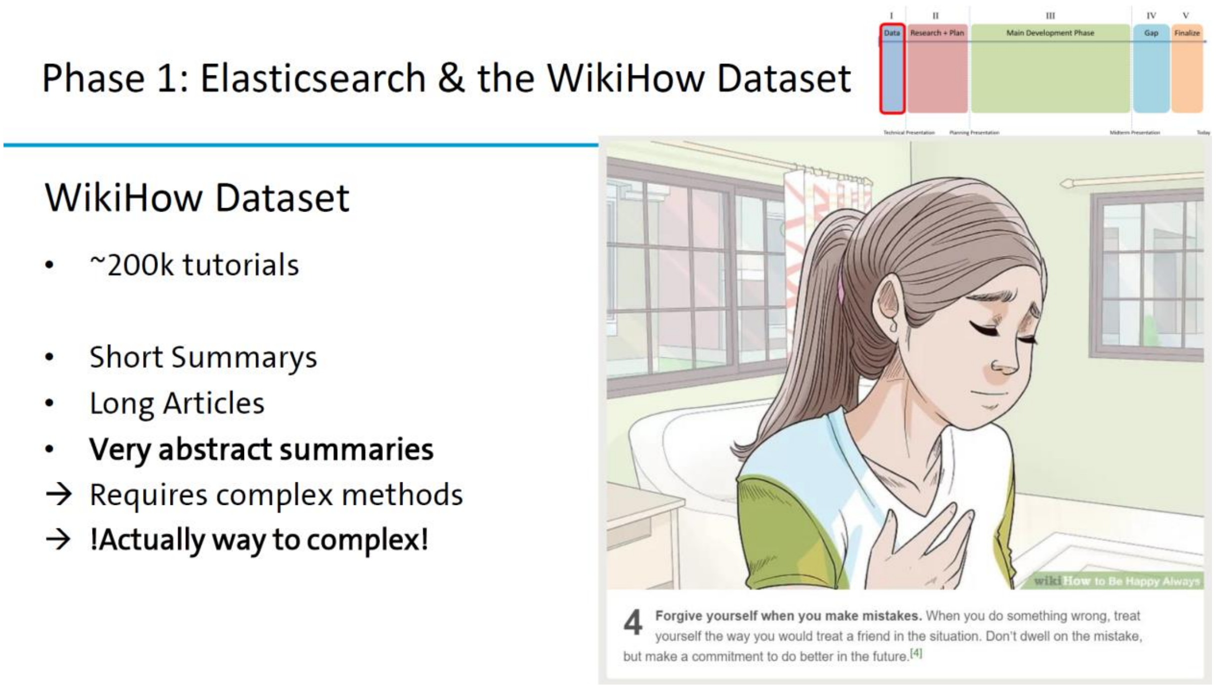
The Task

- Build a QA system
- Build an interface that allows to give feedback & correct results

The main task of the WikiHow QA project was to build a question answering system based on some data source. Possible data sources are WikiHow, which is a website dedicated for answering complex question by giving detailed instructions on how to solve a certain problem, or for example Reddit’s community “Explain it like I am 5”, in short “ELI5”, which is a community that explains complex question using very simple language. Of course, another possibility is to crawl the web and find answers there. For example, crawling “Frequently Asked Questions” from various websites as well as crawling websites that are designed for question answering, for example Quora, would be a good way to start. However, this project simply utilizes the WikiHow dataset, which is described in detail in the next chapter.

The second task for the WikiHow QA project was to develop an interface that enables possible users to give feedback as well as an option to correct results or simply to label results as for example good and bad. By doing so, the final system will be able to improve with the help of users, which is a great idea and definitely a goal to strive for.

Dataset



Phase 1: Elasticsearch & the WikiHow Dataset

WikiHow Dataset

- ~200k tutorials

- Short Summarys
- Long Articles
- **Very abstract summaries**

→ Requires complex methods
→ **!Actually way to complex!**

4 **Forgive yourself when you make mistakes.** When you do something wrong, treat yourself the way you would treat a friend in the situation. Don't dwell on the mistake, but make a commitment to do better in the future.[4]

The WikiHow dataset was released in October 2018 together with the paper "WikiHow: A Large Scale Text Summarization Dataset" written by Mahnaz Koupaee and William Yang Wang. The dataset is available on GitHub (https://github.com/mahnazkoupaee/WikiHow-Dataset) and is still updated regularly according to the authors.

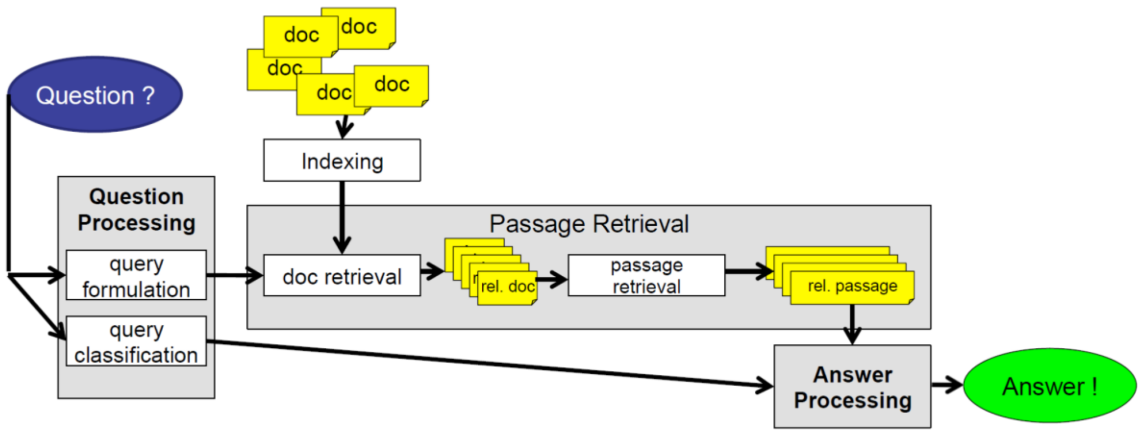
As the title of the paper suggests, the dataset was extracted from the online knowledge base "WikiHow", which is written by many different human authors. Also, the dataset is meant to be used as a dataset for summarization.

The dataset consists of more than 230,000 articles and for each article there exists a summary. These article and summary pairs were extracted automatically from "WikiHow", which is shortly described next.

A WikiHow article typically consists of a headline, which is the question to be answered, followed by various steps, that explain how to solve the problem at hand. Each step consists of a main sentence, that could stand for itself, followed by some more sentences explaining this step in a more detailed way. The summary is then generated by concatenating all main sentences, whereas the article is constructed by concatenating all explaining sentences.

Therefore, this dataset requires very complex summarization techniques to achieve good results. The authors state that current state-of the art sequence-to-sequence models are not able to perform well on this dataset, which calls for new innovative models.

Phase 2: Research, Planning & First UI



Since WikiHow QA is a Question Answering system, it was necessary to look at Question Answering system architectures before starting the development. A typical Question Answering architecture can be seen above.

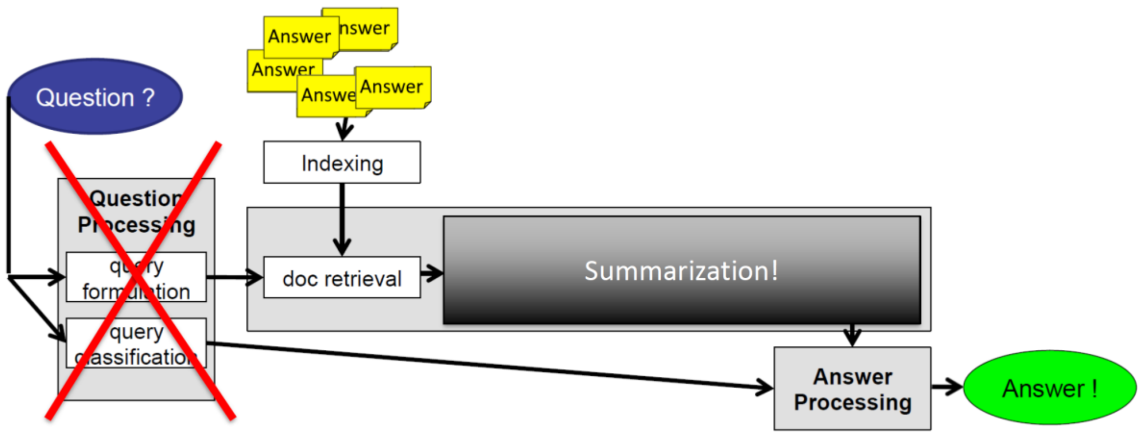
First, a user inputs a question into the system. During the Question Processing step, this question is further analyzed. Typically, the system tries to extract the question type, the expected answer type, which is later useful for ranking different possible answers, important named entities, the main topic of the question and so on.

With the help of this extracted information, a query is formulated, which is then used to find relevant documents in the document index. The document index could be for example a crawl of the “Frequently Asked Questions” of various websites.

Having the top relevant documents, the next step is to find relevant passages in these documents that might contain the answer to the given question. A passage is typically a sentence but could also be a paragraph or a section. To find the most relevant passages, all passages are ranked for example based on the number of correct Named Entities (that appear in question and passage) or for example based on the number of keywords (that appear in question and passage).

The final step is to utilize the top ranked passages as well as hints from the question processing to construct the final answer to the question.

Phase 2: Research, Planning & First UI



As briefly described before, building a Question Answering system requires many subsystems. For Question Processing a classifier for question type classification and a classifier for answer type classification is needed. Also, a Named Entity Recognition component is helpful as well as a Keyword Extraction component. Moreover, to find and determine the topic of the question, for example a topic model would be necessary. Next, to build a document index, it is necessary to develop a crawler as well as to set up a database, where all documents are stored, indexed and ranked. For passage retrieval, a component that is able to extract the most relevant passages based on the Question Processing is needed. Lastly, a final component that constructs the answer is needed as well as a frontend, so that a possible user could also use this system. In total, this are 9 different subsystems, most of which are not of-the-shelf NLP systems.

Since this seemed way out of scope for this short project, the typical question answering architecture had to be changed. As shown in the image above, the Question Processing component was completely removed. Also, the index that contains various documents was replaced by an index that contains answers only, which in this case are answers from the WikiHow dataset. Moreover, the Passage Retrieval component was replaced by a Summarization component that sums up the WikiHow articles to give a short and precise answer to the question. In total, the new architecture consists of 3 components (Answer Index, Summarization, Frontend) which seemed like a much more reasonable scope for this project.

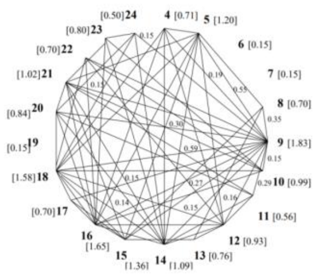
Summarization

Phase 2: Research, Planning & First UI

Summarization

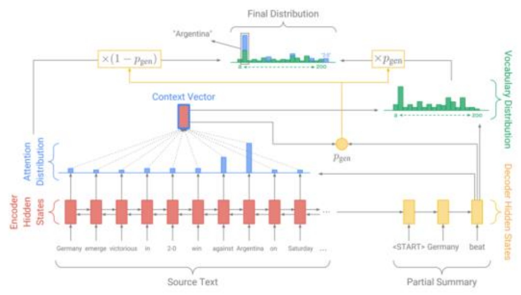
Extractive

- consists of the original phrases and sentences



Abstractive

- text which is different from the original document



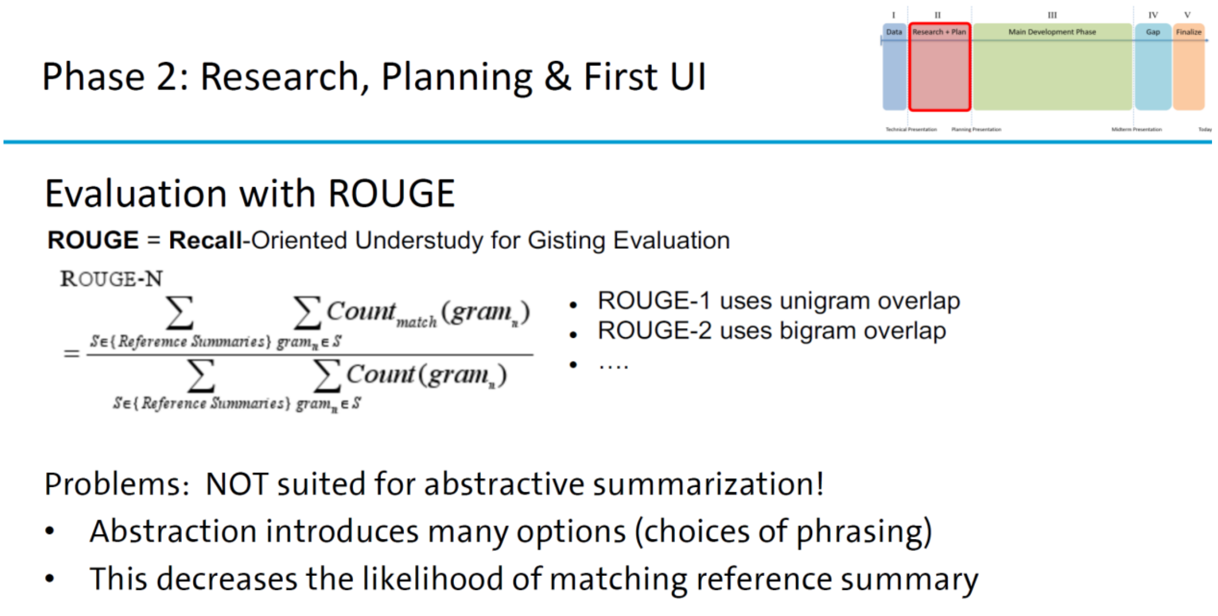
As described in the chapter before, the WikiHow QA system utilizes a summarization component. Therefore, it is helpful to take a look at different summarization techniques.

Summarization is a long-researched topic of the NLP community and is typically divided into two types: extractive and abstractive summarization. Extractive summarization systems produce extracts, whereas abstractive summarization systems produce abstracts. An extract consists of the original phrases and sentences from the document that should be summarized, whereas an abstract is a text which is different from the original document. Hence, extractive summarization is regarded as the easier approach to summarization.

However, approaches to summarization can be further categorized. For example, there are single document summarization systems, as opposed to multi-document summarization. Also, there are generic summarization systems that do not account the user's information need, as opposed to query-focused summarization systems.

Regarding the WikiHow QA project, the goal was to have both summarization techniques available. At least on extractive and one abstractive summarization component were developed to be able to compare these two different approaches. The developed summarization components are generic, single-document summarization systems.

Summarization Evaluation


I
II
III
IV
V
Data
Research + Plan
Main Development Phase
Gap
Finalize
Technical Presentation
Planning Presentation
Midterm Presentation
Today

Phase 2: Research, Planning & First UI

Evaluation with ROUGE

ROUGE = **Recall**-Oriented Understudy for Gisting Evaluation

$$\text{ROUGE-N} = \frac{\sum_{S \in \{Referemce\ Summaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

- ROUGE-1 uses unigram overlap
- ROUGE-2 uses bigram overlap
- ….

Problems: NOT suited for abstractive summarization!

- Abstraction introduces many options (choices of phrasing)
- This decreases the likelihood of matching reference summary

As stated in the chapter before, one goal of this project was to develop two different summarization components, one extractive and one abstractive, and to compare these two approaches. To evaluate and compare different summaries of the same document, NLP researchers have used the ROUGE-N metric for a long time. ROUGE is short for Recall-Oriented Understudy for Gisting Evaluation. The formula can be seen in the image above.

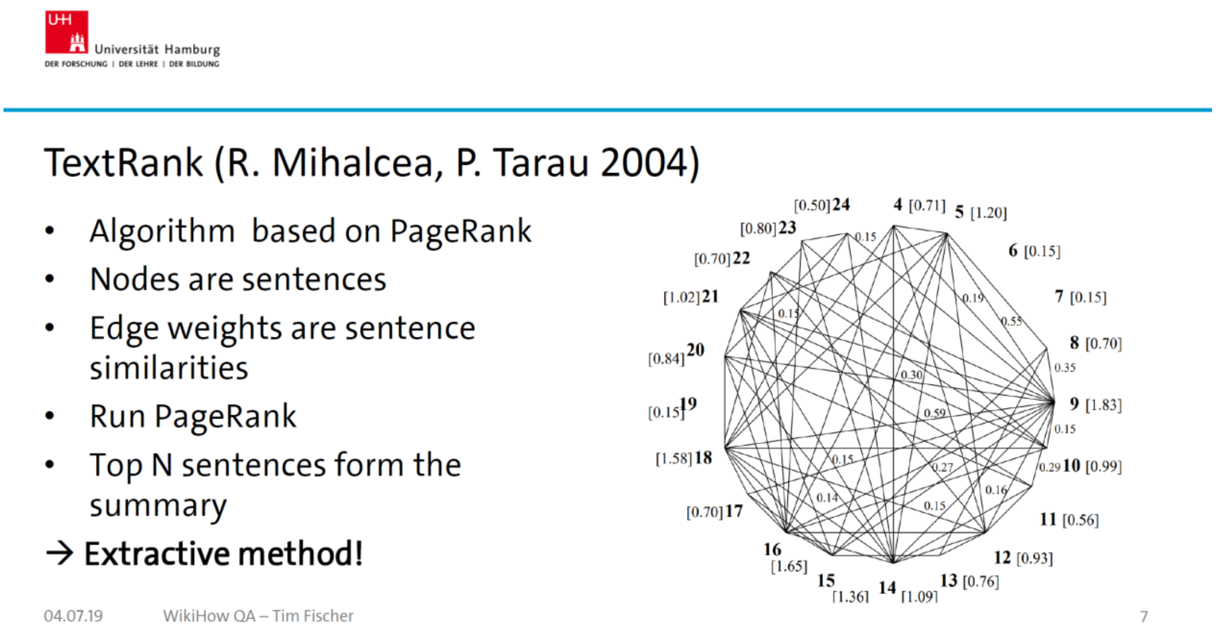
The ROUGE evaluation is based on a very simple mechanic: Given a gold standard summary, which is considered as the “correct” summary, and the system summary, calculate the n-gram overlaps between these two summaries. This basically leads to: “the more and the longer n-gram overlaps, the better”.

Unfortunately, this measure is not suited for abstractive summarization systems. Abstractive summarizations introduce many new words, that may neither appear in the document nor in the gold standard summary. So even though the meaning of the abstractive summaries may be the same, the words may not. Therefore, the n-gram overlaps are less and shorter, which consequently leads to worse scores. Using ROUGE as evaluation often leads to extractive summarization systems performing better than abstractive summarization systems.

Even though this problem is well known among the researches, ROUGE is still used as it works well enough to compare different summarization methods. However, for the WikiHow QA project, a new goal rose: Provide a better & user-friendly method to compare different summarization methods.

Summarization methods

TextRank



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

TextRank (R. Mihalcea, P. Tarau 2004)

- Algorithm based on PageRank
- Nodes are sentences
- Edge weights are sentence similarities
- Run PageRank
- Top N sentences form the summary

→ **Extractive method!**

04.07.19 WikiHow QA – Tim Fischer 7

[0.50] 24
4 [0.71]
5 [1.20]
[0.80] 23
6 [0.15]
[0.70] 22
7 [0.15]
[1.02] 21
8 [0.70]
[0.84] 20
9 [1.83]
[0.15] 19
10 [0.99]
[1.58] 18
11 [0.56]
[0.70] 17
12 [0.93]
16 [1.65]
13 [0.76]
15 [1.36]
14 [1.09]

TextRank is an extractive summarization method developed by Rada Mihalcea and Paul Tarau in 2004. It is a simple, but very effective way of using graph-based algorithms for summarization.

First, a graph needs to be created to initialize the algorithm. Therefore, each sentence is considered as a node, whereas edges represent the sentence similarities. Sentence similarities can be calculated for example by using cosine-similarity.

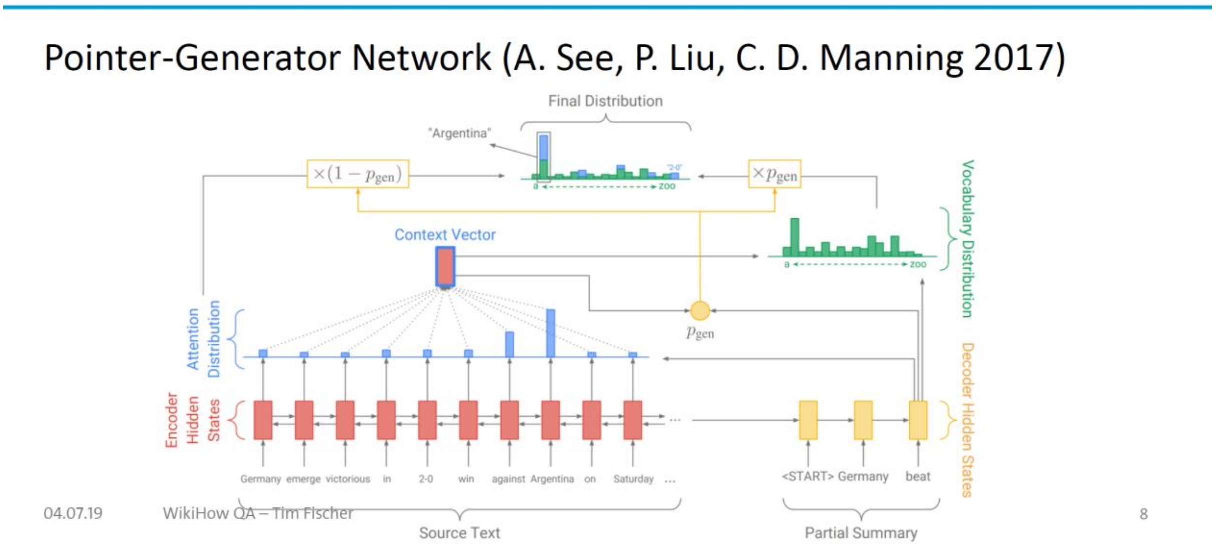
Next, PageRank, a popular graph-based ranking algorithm, is executed on this sentence graph until it converges. PageRank basically gives nodes a higher ranking the more in-links they have, with the addition that in-links from important nodes are more important.

Finally, the top N sentences are selected, ordered and form the summary. A simple heuristic is to order the sentences is simply to order them as they appear in the original document.

TextRank comes with one disadvantage. It is necessary to specify the parameter N (the number of sentences that form the final summary). In some applications, it is not known beforehand, how many sentences or words the generated summary should have. So, this could be a problem.

Pointer-Generator Network

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG



Pointer-Generator Network (A. See, P. Liu, C. D. Manning 2017)

04.07.19 WikiHow QA – Tim Fischer 8

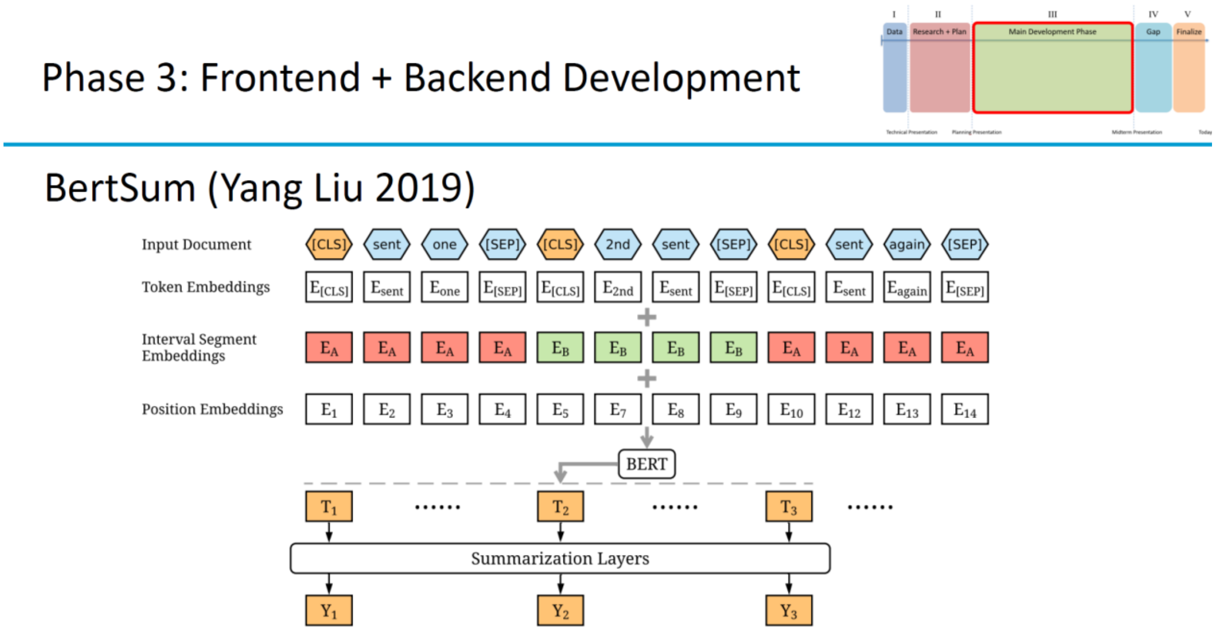
The Pointer-Generator Network is an abstractive summarization method and was developed by A. See, P. Liu and Christopher Manning in 2017. At the time of its development, this model achieved state-of-the-art performance and is still considered as a top performing model for abstractive summarization.

At first glance, this neural network is a simple encoder-decoder model. Basically, the document to be summarized is encoded sentence by sentence by the encoder into a neural representation. This neural representation of a sentence of the document is now decoded by the decoder into a summary sentence.

In addition to that, the neural network utilizes a so-called context vector, which is created by aggregating the attentions of the encoder. This context vector basically represents “what has already been said”. Therefore, the context vector helps the model to avoid repetition.

Lastly, the Pointer-Generator Network makes use of a special variable called p_gen, which is responsible for making this model an abstractive summarization method. This variable is directly influenced by the context vector as well as the decoder and represents the probability of choosing a word from the document or choosing a word from the vocabulary. In other words, this variable decides whether a word is copied from the original document, or a new word is used from the vocabulary.

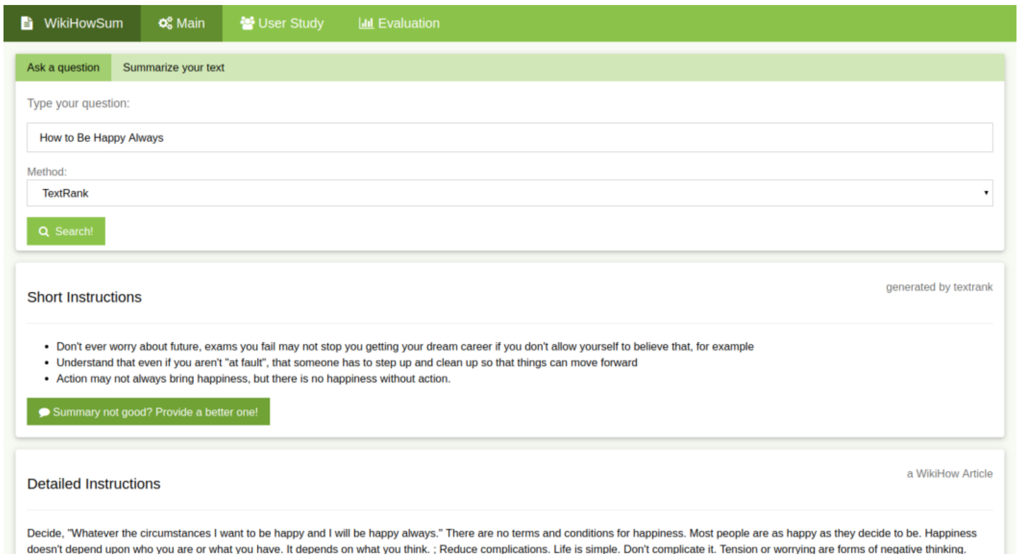
BertSum



The last method used by WikiHow QA to summarize answers is called BertSum and was developed by Yang Liu in 2019. This is an extractive summarization method that utilizes BERT and is currently considered as the state-of-the-art model for extractive summarization. To understand this method, it is first important to briefly understand BERT. BERT is a famous pretrained transformer that creates context-based word and sentence embeddings. These embeddings carry a huge amount of contextual information, which is the reason why BERT was able to achieve great performance in many NLP tasks. BERT takes word-pieces (special BERT word tokens) as input, whereby the first token must be the special [CLS] token, marking the beginning of the sentence, and the last token must be the special [SEP] token, marking the end of the sentence. Typically, BERT is used with one or two sentences, as it was pretrained with next-sentence prediction and masked language modeling. Then, a sentence embedding is obtained by taking the embedding of the [CLS] token. Yang Liu, however, kind-of “tricked” BERT by inserting multiple sentences, each with its own [CLS] token, as it can be seen in the image above. In this way, the proposed BertSum model can calculate multiple sentence embeddings at a time. In short, the BertSum model inputs every sentence of the document to be summarized into BERT. Next, the model gets each sentence embedding. Lastly, each sentence embedding is forwarded into another neural network, for example a simple linear layer (but this layer can be complex as well), that calculates a ranking of each sentence. The final summary is then created by concatenating the top N ranked sentences.

Features

Feature 1: Answering “How to …” Questions



Type your question: How to Be Happy Always

Method: TextRank

Short Instructions — generated by textrank

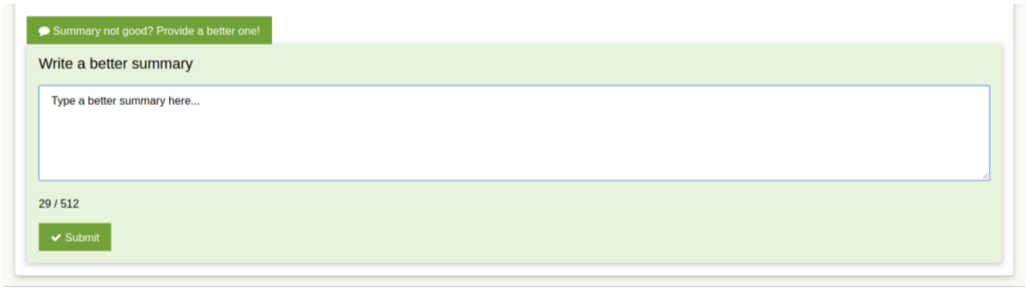
- Don't ever worry about future, exams you fail may not stop you getting your dream career if you don't allow yourself to believe that, for example
- Understand that even if you aren't "at fault", that someone has to step up and clean up so that things can move forward
- Action may not always bring happiness, but there is no happiness without action.

Detailed Instructions — a WikiHow Article

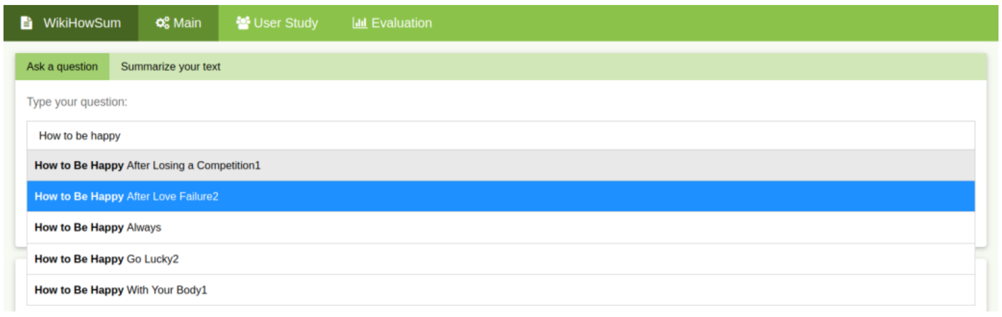
Decide, "Whatever the circumstances I want to be happy and I will be happy always." There are no terms and conditions for happiness. Most people are as happy as they decide to be. Happiness doesn't depend upon who you are or what you have. It depends on what you think. ; Reduce complications. Life is simple. Don't complicate it. Tension or worrying are forms of negative thinking.

The image above shows the Question Answering component of WikiHow QA. It shows the input mask, as well as the answers. Features that are available on this view are: Question Answering, User-Feedback and Auto-Completion.

Note that there are two answers. The "Short Instructions" box shows the summarized answer, whereas the "Detailed Instructions" box shows the full detailed answer to the question. The "Short Instructions" box also contains a button asking the user to provide a better summary, if the current one is not satisfying as it can be seen in the image below.


Summary not good? Provide a better one!
Write a better summary
Type a better summary here...
29 / 512
Submit

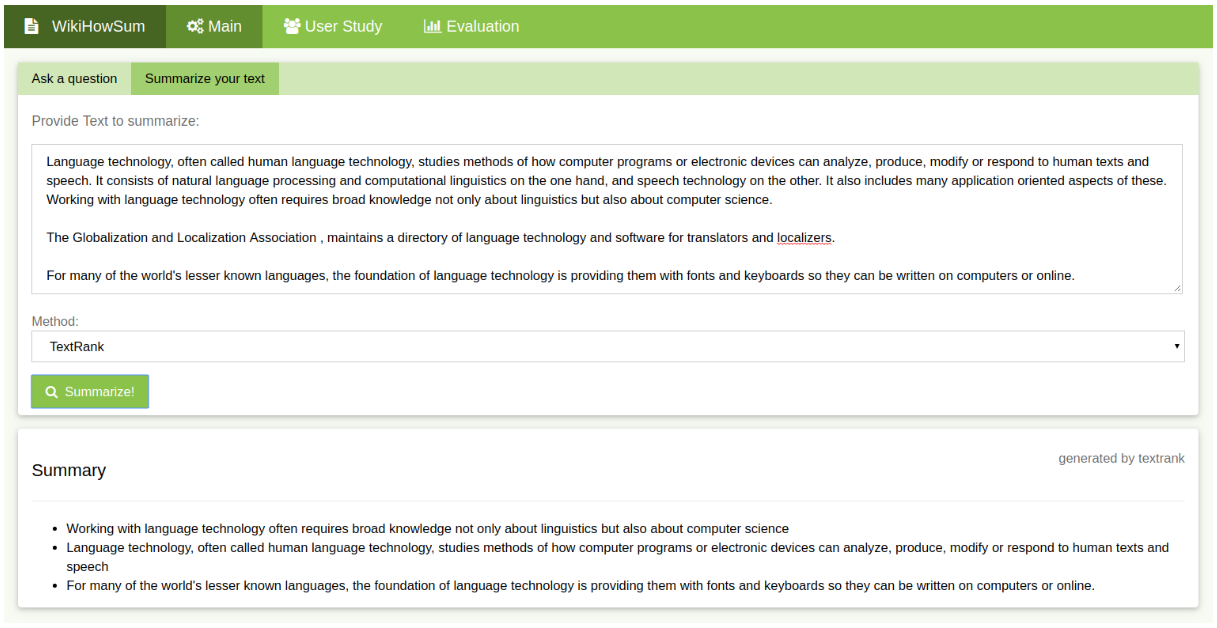
The input mask consists of a simple input field for the question as well as a drop-down menu to select the summarization method. While typing in the input field, the system will suggest possible questions. These suggestions can be selected either with the arrow keys and enter key or simply with a mouse click. See the image below to get an idea of the auto completion.



Type your question: How to be happy

- **How to Be Happy** After Losing a Competition1
- **How to Be Happy** After Love Failure2
- **How to Be Happy** Always
- **How to Be Happy** Go Lucky2
- **How to Be Happy** With Your Body1

Feature 2: Summarize provided text



Provide Text to summarize:

Language technology, often called human language technology, studies methods of how computer programs or electronic devices can analyze, produce, modify or respond to human texts and speech. It consists of natural language processing and computational linguistics on the one hand, and speech technology on the other. It also includes many application oriented aspects of these. Working with language technology often requires broad knowledge not only about linguistics but also about computer science.

The Globalization and Localization Association , maintains a directory of language technology and software for translators and localizers.

For many of the world's lesser known languages, the foundation of language technology is providing them with fonts and keyboards so they can be written on computers or online.

Method: TextRank

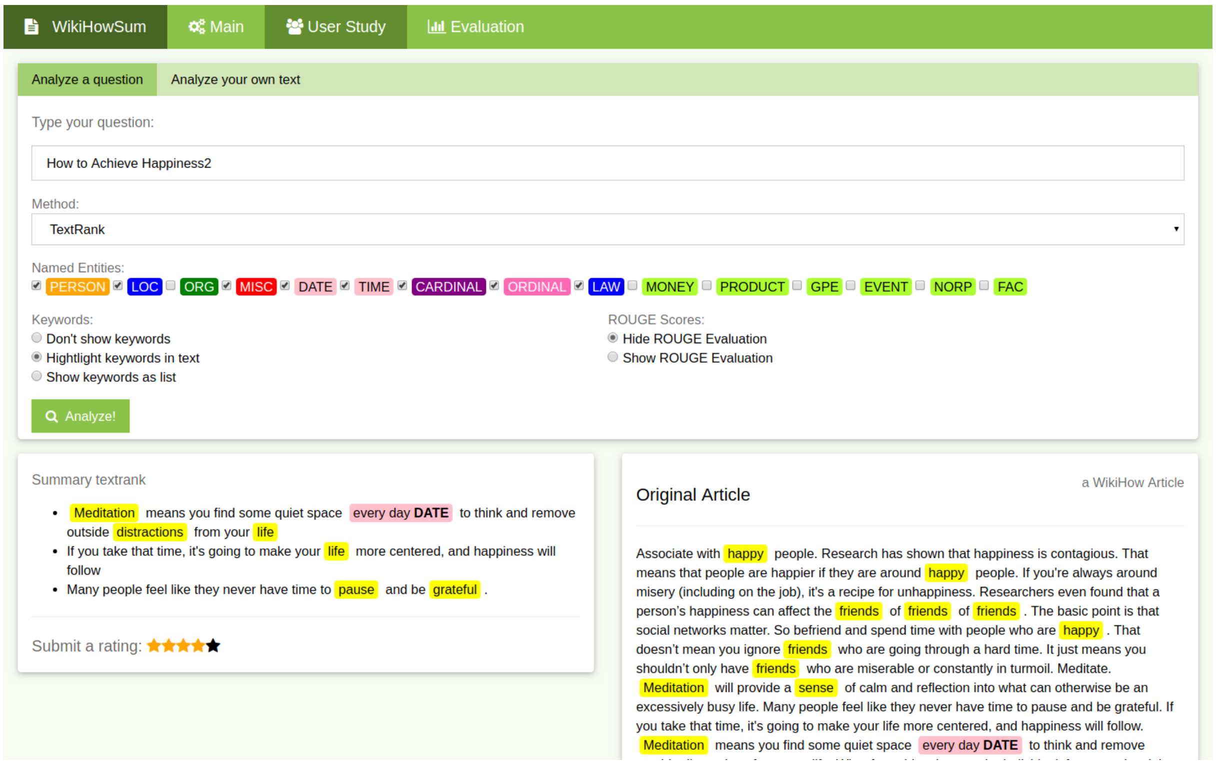
Summary

generated by textrank

- Working with language technology often requires broad knowledge not only about linguistics but also about computer science
- Language technology, often called human language technology, studies methods of how computer programs or electronic devices can analyze, produce, modify or respond to human texts and speech
- For many of the world's lesser known languages, the foundation of language technology is providing them with fonts and keyboards so they can be written on computers or online.

The image above shows the summarization component of WikiHow QA. It shows the input mask with some example text as well as it's summary. Similar as with the Question Answering component, the input mask consists of a simple text input field for the text and a drop-down menu to select the summarization method.

Feature 3: Evaluate summarization techniques



Type your question:

How to Achieve Happiness2

Method:

TextRank

Named Entities:
✓ PERSON ✓ LOC ☐ ORG ✓ MISC ✓ DATE ✓ TIME ✓ CARDINAL ✓ ORDINAL ✓ LAW ☐ MONEY ☐ PRODUCT ☐ GPE ☐ EVENT ☐ NORP ☐ FAC

Keywords:
○ Don't show keywords
● Hightlight keywords in text
○ Show keywords as list

ROUGE Scores:
● Hide ROUGE Evaluation
○ Show ROUGE Evaluation

Analyze!

Summary textrank

- Meditation means you find some quiet space every day DATE to think and remove outside distractions from your life
- If you take that time, it's going to make your life more centered, and happiness will follow
- Many people feel like they never have time to pause and be grateful .

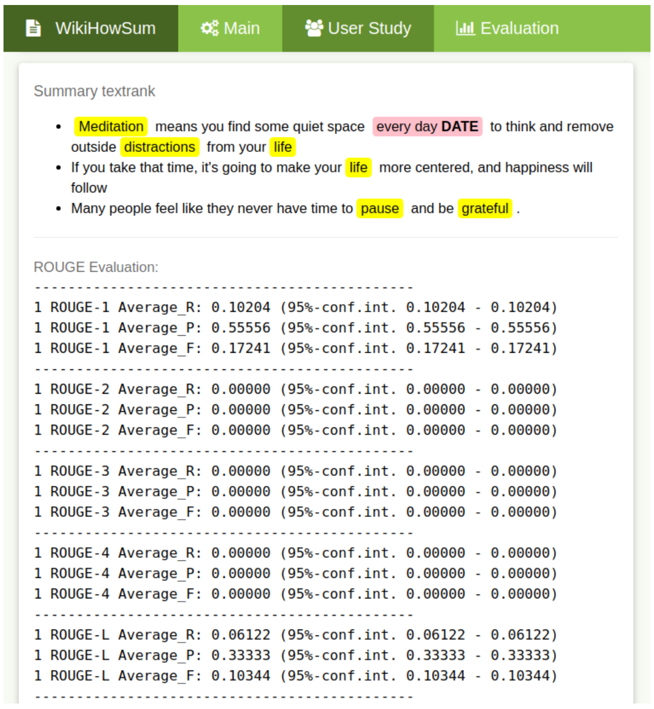
Submit a rating: ★★★★★

Original Article

a WikiHow Article

Associate with happy people. Research has shown that happiness is contagious. That means that people are happier if they are around happy people. If you're always around misery (including on the job), it's a recipe for unhappiness. Researchers even found that a person's happiness can affect the friends of friends of friends . The basic point is that social networks matter. So befriend and spend time with people who are happy . That doesn't mean you ignore friends who are going through a hard time. It just means you shouldn't only have friends who are miserable or constantly in turmoil. Meditate. Meditation will provide a sense of calm and reflection into what can otherwise be an excessively busy life. Many people feel like they never have time to pause and be grateful. If you take that time, it's going to make your life more centered, and happiness will follow. Meditation means you find some quiet space every day DATE to think and remove

The User Interface of the "User Studies" component of WikiHow QA can be seen in the image above. The output is organized differently compared to the other components. One or more summaries are shown on the left side and the full article is shown on the right side. In contrast to previous components, this component has a very complex input mask. Again, there is an input field for the question as well as a drop-down menu to select the summarization method. Moreover, there are various checkboxes to select different types of Named Entities that should be visualized in the summary and in the full article. Also, it is possible to highlight keywords in the texts as in the image above or just display them as a list. Moreover, the ROUGE evaluation score can be displayed as seen below.



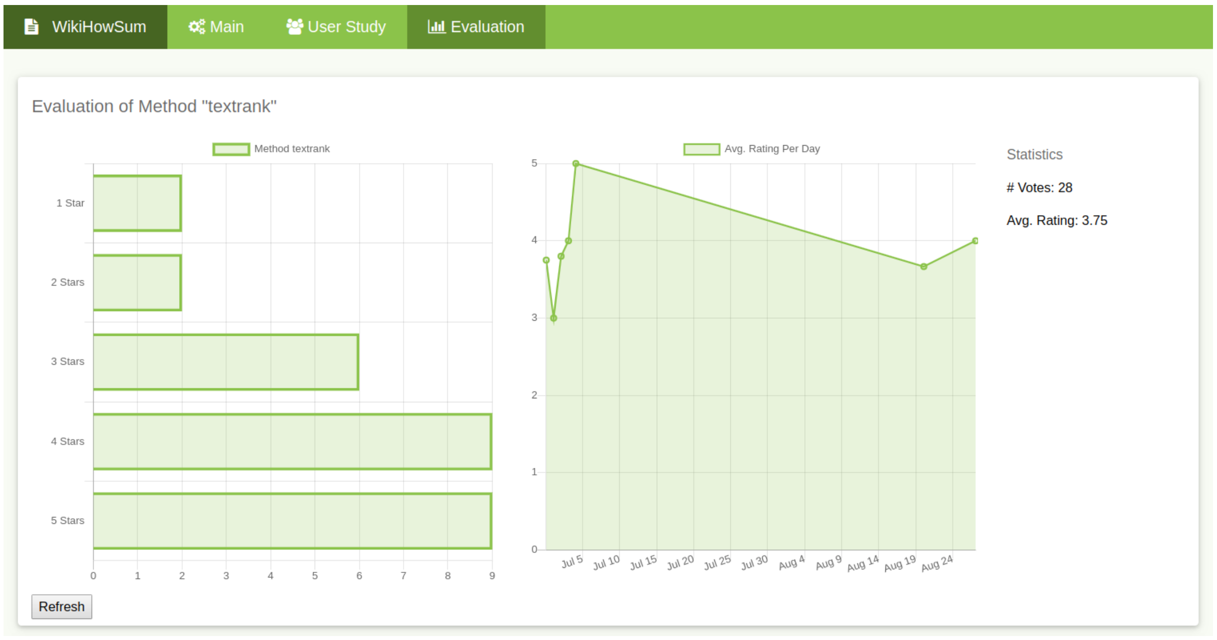
Summary textrank

- Meditation means you find some quiet space every day DATE to think and remove outside distractions from your life
- If you take that time, it's going to make your life more centered, and happiness will follow
- Many people feel like they never have time to pause and be grateful .

ROUGE Evaluation:

```
---------------------------------------------
1 ROUGE-1 Average_R: 0.10204 (95%-conf.int. 0.10204 - 0.10204)
1 ROUGE-1 Average_P: 0.55556 (95%-conf.int. 0.55556 - 0.55556)
1 ROUGE-1 Average_F: 0.17241 (95%-conf.int. 0.17241 - 0.17241)
---------------------------------------------
1 ROUGE-2 Average_R: 0.00000 (95%-conf.int. 0.00000 - 0.00000)
1 ROUGE-2 Average_P: 0.00000 (95%-conf.int. 0.00000 - 0.00000)
1 ROUGE-2 Average_F: 0.00000 (95%-conf.int. 0.00000 - 0.00000)
---------------------------------------------
1 ROUGE-3 Average_R: 0.00000 (95%-conf.int. 0.00000 - 0.00000)
1 ROUGE-3 Average_P: 0.00000 (95%-conf.int. 0.00000 - 0.00000)
1 ROUGE-3 Average_F: 0.00000 (95%-conf.int. 0.00000 - 0.00000)
---------------------------------------------
1 ROUGE-4 Average_R: 0.00000 (95%-conf.int. 0.00000 - 0.00000)
1 ROUGE-4 Average_P: 0.00000 (95%-conf.int. 0.00000 - 0.00000)
1 ROUGE-4 Average_F: 0.00000 (95%-conf.int. 0.00000 - 0.00000)
---------------------------------------------
1 ROUGE-L Average_R: 0.06122 (95%-conf.int. 0.06122 - 0.06122)
1 ROUGE-L Average_P: 0.33333 (95%-conf.int. 0.33333 - 0.33333)
1 ROUGE-L Average_F: 0.10344 (95%-conf.int. 0.10344 - 0.10344)
---------------------------------------------
```

Feature 4: Statistics



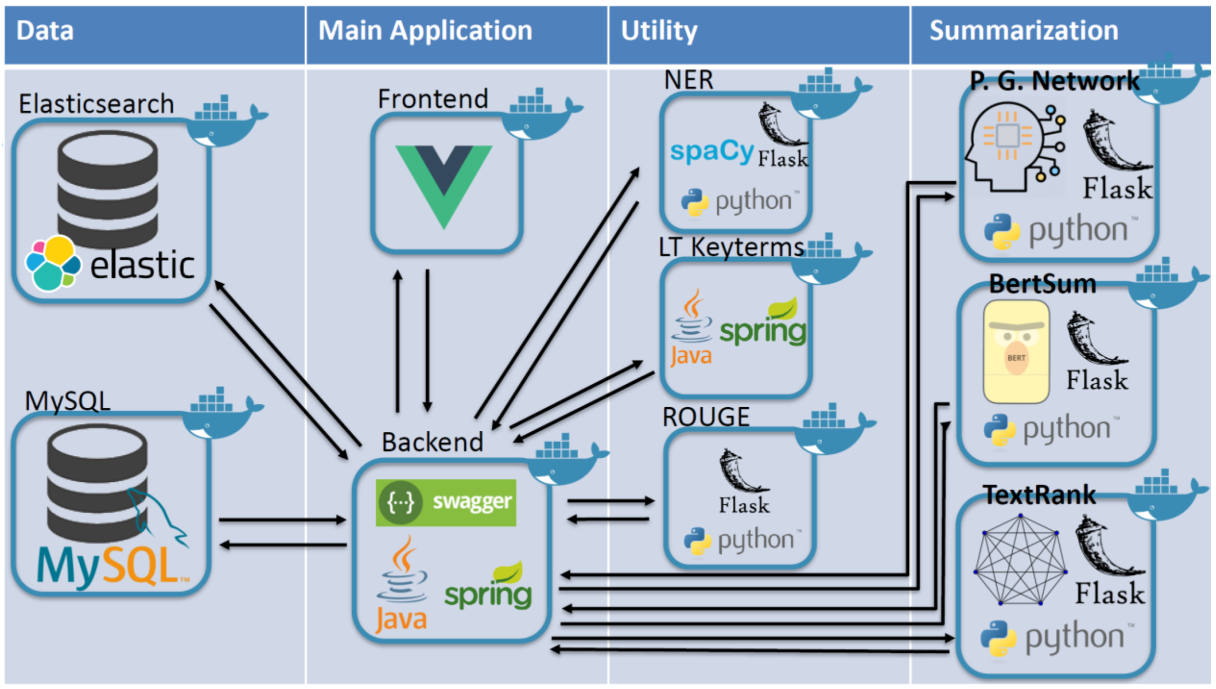
The statistics component of WikiHow QA visualizes collected ratings. Both, the total ratings graph as well as the average ratings per day graph for a certain summarization technique are shown in the image above. Any user can submit a star rating in the “User Studies” component, please see the image of “Feature 3: Evaluate summarization techniques”.

On the left side, the total ratings graph shows how many and which star ratings a certain summarization technique received.

On the right side, the average ratings per day graph visualizes the change of the ratings over time for a certain summarization technique.

Next to that, basic statistics like the number of total votes are displayed. By scrolling down this page, the user can view the visualizations for all the other summarization techniques.

Application Architecture



The WikiHow QA system is at its core an ensemble of various microservices, multiple databases, a backend that combines all of them and a single frontend that utilizes all the features provided by the various microservices. Each of these small subsystems lives in its own docker container. As the image above suggests, the WikiHow QA system can be roughly grouped into 4 categories: Data, main application, utility and summarization.

The two services responsible for holding all the relevant data are Elasticsearch and MySQL. Elasticsearch holds an index for each WikiHow article and also a special index that allows auto-completion for question titles. The MySQL database stores the summaries as well as the star ratings that are provided by the users.

The three summarization services are stateless microservices that just provide one single functionality: Summarizing a given text. The functionality can be accessed with a simple API call. Each summarization method runs in its own environment as they have very different requirements (for example different TensorFlow versions).

All three utility services are also stateless microservices that each provide a single functionality. The NER component utilizes a spaCy NER model to find Named Entities given a certain text. The LT Keyterms service utilizes a large background corpus to find keywords given a text. Lastly, the ROUGE microservice, calculates the ROUGE evaluation score given a gold and system summary. All three microservices are used in the "User-Studies" feature of WikiHow QA's user interface.

The Main Application consists of two services. The Backend component communicates with every single component and mainly acts as a broker between the frontend and every other service. This is achieved by exposing one single unified API to the frontend. The Frontend component, therefore, only needs to communicate with the Backend to make use of every available service. With access to all services, the frontend can realize all features explained in the previous chapter.

Technology (API, Databases, Frontend)

Technology

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

To build and develop the WikiHow QA system various technologies were used.

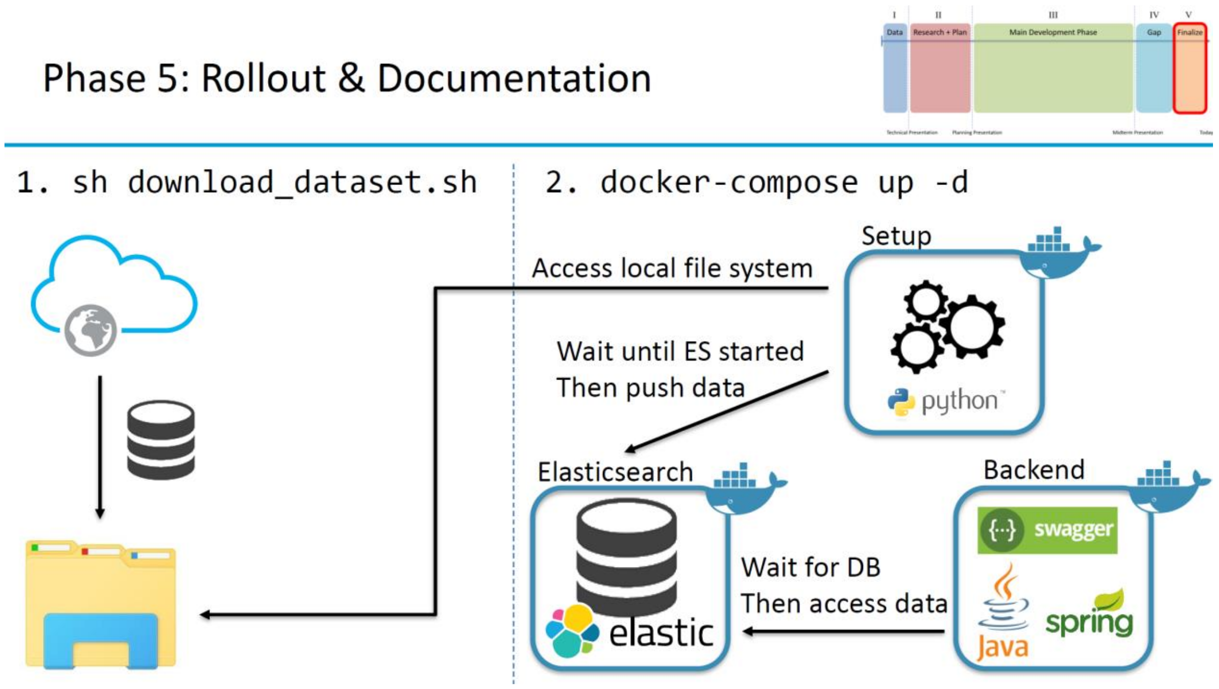
The three main programming languages used in this project are JavaScript, Python and Java. The frontend is completely written with JavaScript and the Vue.js framework. Most microservices were written with Python. Lastly, the Backend that combines all microservices was written with Java.

Each microservices has its own functionality that needs to be exposed and accessible. For that reason, the Python framework Flask was used to create a public REST API in most cases. The Backend component utilized the Java Spring framework in combination with the Swagger framework to create the REST API. Swagger provides a great automatically generated documentation of the developed REST API, which made the frontend development and communication with the Backend component much easier.

The data of the WikiHow QA system is stored using Elasticsearch and MySQL. Elasticsearch was chosen because of its powerful indexing and fast access to unstructured information like full text articles. Therefore, Elasticsearch stores all WikiHow articles. MySQL was chosen because of its full integration with the Java Spring framework used by the Backend component. This way, it was easy to store user generated feedback in the database.

Lastly, every single component utilizes Docker to make the whole WikiHow QA system as independent as possible and as easy to install as possible.

Installation / Deployment



Even though the WikiHow QA system is a very complex ensemble of many different subsystems, it is fairly easy to install & setup.

To understand why it is hard to install such a large system, it is first necessary to understand the dependencies between the various components. The Frontend is completely dependent on the Backend. This means that without a running Backend component, no features of the Frontend are working. The Backend component, however, is dependent on the Elasticsearch component, which holds all WikiHow articles.

One important criterion for the deployment was to be able to startup all docker containers at once and then the system is running. Even, if it is the first startup! Therefore, the startup order of the single components had to be enforced. Also, a service that sets up and imports all WikiHow Articles into Elasticsearch, but only runs on the first startup, was necessary.

To install the WikiHow QA system, two steps are required. First, the WikiHow Articles must be downloaded. Next, all Docker containers must be started. Now, the Frontend starts when the Backend is available and the Backend starts when the Elasticsearch index is available. Since on first startup the Elasticsearch index is not available, the Setup component runs. This component waits until the Elasticsearch component is running, accesses the previously downloaded WikiHow Articles and imports them using a Python script. Once this is done, the whole application is running.