

Lifhack Search engine: documentation and evaluation

Project Report in Information Retrieval, summer semester 2019

Lecturer: Jun-Prof. Dr. Martin Potthast

Franziska Görg

Miriam Amin

Table of Contents

1. Introduction	3
2. Documentation.....	4
2.1. Data set	4
2.2. Architecture.....	5
2.2.1. Technology Stack	5
2.2.2. Retrieval Model	6
3. Evaluation.....	7
3.1. Experimental setup.....	7
3.2. Performance Test.....	9
3.3. Results	11
4. Discussion	14
5. Appendix	15
Figure 1: Screenshot of a post containing question, one answer and comments	4
Figure 2: Screenshot of the JSON defining our documents' underlying data structure	5
Figure 3: Judgement Results for LE	11
Figure 4: Judgement results for LSE	12
Figure 5: Comparison of both systems' Average Presicion by topic	13

1. Introduction

Recent developments in the field of facilitating life have led to a renewed interest in Lifehacks. A Lifehack is ‘a technique that can be implemented quickly and is used to make one's physical life more efficient when a more standard approach (as defined by that area's experts) or a product is either unavailable or undesirable. Lifehacks are creative, meaning they use materials that are on hand for uses besides their intended use.’ (Saucier, 2014)

To solve a problem creatively or to reach the goal in an unusual way creates an information need to find hacks matching a present problem. In this context, a more efficient and effective way of life can possibly be achieved. It is our goal to provide the information needed for a concrete and relevant lifehack. It is our motivation to meet this need clearly and efficiently with the means of a web application. Last but not least, we want to rely on state-of-the-art technologies to realize this goal efficiently.

The website Lifehacks Stack Exchange provides a large number of Lifehacks for different purposes. The aim of this report is to develop a search engine to make the documents originating from there searchable. This paper will also investigate the ranking efficiency of our system compared to the search function implemented on the website Lifehacks Stack Exchange.

The report is organized in two major sections. The first part documents the architecture of our system and the structure of our data set. It gives an overview of the technology stack and retrieval model underlying our search engine.

Chapter 3 evaluates the efficiency of our system. It describes the experimental setup and presents the results of the performance tests. Finally, the discussion gives a brief summary and critique of the findings.

2. Documentation

2.1. Data set

The data for our domain originates from the website lifehacks.stackexchange.com. Lifehacks Stack Exchange ‘is a question and answer site for people looking to bypass life's everyday problems with simple tricks’ (“Lifehacks Stack Exchange,” n.d.). It is part of the Stack Exchange question and answer forum network¹, a network of 173 different Question & Answer communities with Stack Overflow as their largest member (Stack Exchange Inc., 2019).

We consider every question and the corresponding answers and comments one document. Figure 1 shows the documents’ original structure as a screenshot from the original website.

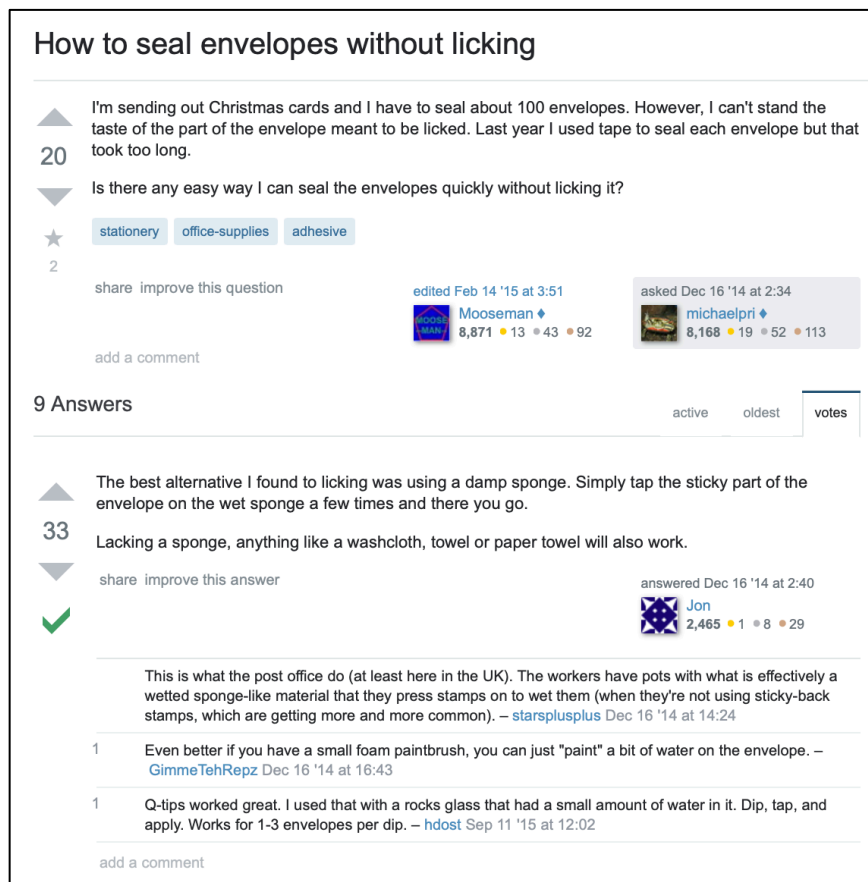
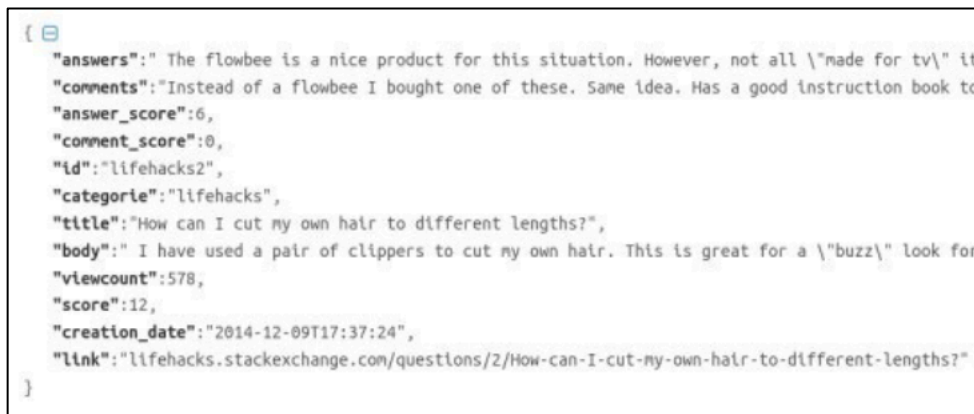


Figure 1: Screenshot of a post containing question, one answer and comments

¹ <https://stackexchange.com/>

We downloaded the data base for Lifehack-Engine from archive.org as zip file, containing 2.375 structured documents with a variety of variables as xml-files. With regard to the information need of the users we defined the following variables as the variables of interest: title, answers, question, comments, score and viewcount. It is noteworthy that in these documents, all answers and comments are stored together in one field.

Part of the data preprocessing implemented in our system is the conversion from XML- to JSON-files. An example of such a file is illustrated in Figure 2.

A screenshot of a JSON document structure displayed in a code editor. The JSON object contains several key-value pairs: "answers" (a string), "comments" (a string), "answer_score" (an integer), "comment_score" (an integer), "id" (a string), "categorie" (a string), "title" (a string), "body" (a string), "viewcount" (an integer), "score" (an integer), "creation_date" (a string), and "link" (a string). The text is syntax-highlighted, with strings in quotes and numbers in plain text. The code is enclosed in curly braces.

```
{
  "answers": "The flowbee is a nice product for this situation. However, not all \"made for tv\" it",
  "comments": "Instead of a flowbee I bought one of these. Same idea. Has a good instruction book to",
  "answer_score": 6,
  "comment_score": 0,
  "id": "lifehacks2",
  "categorie": "lifehacks",
  "title": "How can I cut my own hair to different lengths?",
  "body": "I have used a pair of clippers to cut my own hair. This is great for a \"buzz\" look for",
  "viewcount": 578,
  "score": 12,
  "creation_date": "2014-12-09T17:37:24",
  "link": "lifehacks.stackexchange.com/questions/2/How-can-I-cut-my-own-hair-to-different-lengths?"
}
```

Figure 2: Screenshot of the JSON defining our documents' underlying data structure

2.2. Architecture

The main aim of this project is to develop software that allows us to search in the above described data set. Therefore, it is necessary to develop a suitable technology stack and retrieval model. With regards to our data, we call our system Lifehack-Engine. The following sections give an outline of the system's architecture.

2.2.1. Technology Stack

Our Technology Stack consists of three components: a simple webpage based on HTML and CSS as frontend component, the Python framework Flask as backend component and Elasticsearch as search engine. As code base for our system, we forked the existing Github project 'Search it like it's hot' (Lucas Fijen, 2017).

Flask delivers the statically generated build of the frontend and forwards search queries to Elasticsearch. Elasticsearch is a search engine based on Lucene and combines the typical inverted index and the document store in a so-called Elasticsearch index. The index contains the documents and the document representations. We used the associated administration tool Kibana to manage and ensure the completeness of our indexed documents.

2.2.2. Retrieval Model

After preprocessing, the documents are stored in an inverted index and represented by their index terms. The retrieval model used is an optimized version of the Okapi BM25 model (“Similarity module | Elasticsearch Reference [7.3] | Elastic,” n.d.) which weights the indexed documents with the BM25 relevance function. In addition to the term frequency of the query terms, this also takes into account the length of the documents to be weighted and can also be adjusted by two freely selectable parameters.

Elasticsearch’s function `multi_match` allows us to search for words in multiple fields and to assign a weight to the fields. In our case we decided that the title is most relevant, and the comments are less important for the information need of the user. Consequently, we assigned quadruple weight to the title, double weight to the body of the question and halved the weight of the comments. The weight of the answers remains the default value 1.

3. Evaluation

This chapter describes the methods used for the evaluation of the Lifehack-Engine and discusses the results. The main question addressed in this section is whether our system outperforms the search system implemented in the documents' origin, the `lifhack.stackexchange.com` website. We chose ad-hoc retrieval as retrieval task since it suits our domain and data set better than an interactive retrieval task. Another main advantage is the convenient measurability of retrieval performance.

This section is structured into three parts. We will begin by laying out the experimental setup that underlies our evaluation. Doing this, we will report how we developed the topics representing information needs and which struggles we faced with regards to our document collection. The second section is concerned with the performance tests we applied and the results we obtained. Finally, the last subsection gives a summary and discusses the findings.

3.1. Experimental setup

Following the Cranfield evaluation paradigm (Webber, 2009), we conducted our laboratory experiments using a document collection, a set of information needs represented as topics and a set of relevance judgments.

A methodologic difficulty lies in the fact that we are comparing two systems with two different setups and components. Our Lifehack-Engine (LE) has the architecture and retrieval model as described in the preceding section. Retrieval model and general setup of the search function of Lifehack Stack Exchange (LSE) is not known to us. One important obvious difference between the two systems is, that LSE exclusively retrieves and displays documents that are considered as relevant, whereas LE finds the 10 documents with the highest relevance score for a given query, even if they might not reach the threshold for being considered relevant. Despite the differences, a comparison

of the two systems with regards to their retrieval performance is possible. However, the differences prohibit us from drawing conclusions concerning optimizable parameters.

As text collection for the experiment we use our entire document collection. Since it is a relatively small corpus there was no need to draw a test sample.

The definition of topics to model information needs caused some problems due to the characteristics of the documents, the domain and source. As described in the previous section, our document collection originates from the website lifehacks.stackexchange.com. Being part of the Stack Exchange network, Lifehack Stack Exchange shares the network's community rules and guidelines. These comprise the prevention of multiple questions on the same topic through the possibility to mark posts as duplicates. Consequently, and contrarily to conventional web search, almost every document serves a unique information need. For the definition of topics, this means that if they are too specific, in most cases there is either no or exactly one relevant document. To ensure a challenging retrieval task, we decided to test the systems with a mix of less (Example 1) and more specific (Example 2) topics. In total, 20 topics were created by the two authors of this report. The full list of topics including query, description and narrative is enclosed in the appendix.

```
<topic number="18" type="single">
  <query>
    food preparation
  </query>
  <description>
    What are tricks to make food preparation
    easier, faster or more enjoyable?
  </description>
  <narrative>
    Relevant documents can include all kind of
    tricks helping to save time, effort or
    resources during the process of food
    preparation. Relevant documents can include
    all kinds of food and meals.
  </narrative>
</topic>
```

Example 1: Broad information need


```
<topic number="12" type="single">
  <query>
    put toothpaste back in tube
  </query>
  <description>
    avoiding to waste toothpaste
  </description>
  <narrative>
    Relevant documents must include methods and
    instructions how to get the extra
    toothpaste back in the tube. They must also
    include ideas of storing toothpaste aside
    from its common packaging.
  </narrative>
</topic>
```

Example 2: Specific information need

The set of binary relevance judgements to assess the ground truth was also provided by the two authors of this report. For the purpose of comparing the two systems, both relevance judges assessed the first ten results of both hit lists for every topic. Every document in the list was rated as either relevant or irrelevant. We consider documents relevant when they either contain a question that is relevant to our topic or when they contain an answer that gives an answer to our query.

3.2. Performance Test

In order to compare both systems' performance, we calculate the Average Precision for each topic and each system. By summing up the systems' Average Precision scores and dividing the sum by the number of topics, we obtain the Mean Average Precision. We decide MAP to be the relevant metrics for the comparison of our systems.

In order to perform the calculations, we were forced to harmonize the hit lists. As mentioned before, LSE restricts the hit list to relevant results, whereas LE retrieves the 10 results with the highest relevance score. To counter this, we fill the ranks of LSE's hit list for topics with less than 10 hits with the rating 'irrelevant'. For example, for topic 2, LSE retrieves one result. Rank 1 is assigned the rating 'relevant' and rank 2–10 are filled with 'irrelevant'. This modification of the hit list is methodologically unproblematic since Average Precision is calculated with the precision@k scores of

relevant documents. Consequently, adding irrelevant documents at the end of the hit list without relevant documents succeeding will not influence the MAP.

Another characteristic of LSE is that relevant documents sometimes appear twice in the hit list, once marked as Question, the other time marked as Answer. Contrary to our system, Questions and Answer are two different document types in LSE that both can be retrieved. In LE, Answers and Questions are both part of one document (see section 2.1 for data structure) that entails one entire question, answers and comments forum thread. Consequently, even if a question as well as an answer of one document is relevant to our query, the document will only appear once in LE’s hit list. We decided to only assess the first appearance of a question or answer related to the same thread in LSE as relevant since counting one thread twice would manipulate the evaluation to the detriment of LE.

As already mentioned, to measure the ranking effectiveness we calculate the Average Precision as the mean of the Average Precision of both judge’s relevance judgements for every topic and every system. We determine the Average Precision by adding the Precision@k at ranks with relevant documents and dividing the sum by the expected number of relevant documents. We estimate the expected number of relevant documents for a given topic as the maximum of relevant documents retrieved by one of the systems. This heuristic is inaccurate in cases where a relevant document has not been found by either system. Nonetheless, this approach ensures that the more effective system scores a higher value than the less effective system for a given topic.

It is worth mentioning that both assessors gave the same judgement for every topic in every system without exception. Consequently, the judges’ assessment shall not be differentiated in the evaluation. Also, the estimation of the assessor reliability by kappa statistics is therefore obsolete.

3.3. Results

For both systems, we calculated the Average Precision for each topic using the maximum number of relevant docs per topic over both systems as the expected number of relevant docs. Figure 3 and Figure 4 present the results obtained from the relevance judgment and the Average Precision calculated from the ratings. Our system (Figure 3) scored a Mean Average Precision of 0.69, while LSE reached a MAP of 0.62.

Topic	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10	Expected # of rel. docs	Average Precision
0	1	X	O	X	X	X	X	O	O	O	5	0.81
1	2	O	O	O	O	O	O	O	O	O	1	0.00
2	3	X	O	O	O	O	O	O	O	O	1	1.00
3	4	X	X	O	O	O	O	O	O	O	2	1.00
4	5	X	O	O	O	O	O	O	O	O	3	0.33
5	6	O	X	O	X	O	O	O	O	O	2	0.50
6	7	O	O	X	X	O	O	O	O	O	2	0.42
7	8	X	X	O	O	O	O	O	O	O	2	1.00
8	9	X	O	X	O	O	O	O	O	O	2	0.83
9	10	X	O	O	O	O	O	O	O	O	1	1.00
10	11	O	O	O	X	O	X	O	O	X	3	0.29
11	12	X	O	O	O	O	O	O	O	O	1	1.00
12	13	X	O	O	O	O	O	O	O	O	1	1.00
13	14	X	X	O	O	O	O	O	O	O	5	0.40
14	15	X	X	X	O	O	O	X	X	O	8	0.52
15	16	X	X	X	X	O	X	O	O	X	6	0.92
16	17	X	O	X	X	O	O	O	X	O	5	0.58
17	18	X	O	O	O	O	X	O	X	O	3	0.57
18	19	X	X	O	X	O	X	O	X	X	7	0.77
19	20	X	X	X	X	O	O	O	O	X	5	0.91
MAP: 0.6925												

Figure 3: Judgement Results for LE

Topic	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10	Expected # of rel. docs	Average Precision
0	1	X	X	X	X	X	O	O	O	O	5	1.00
1	2	X	O	O	O	O	O	O	O	O	1	1.00
2	3	X	O	O	O	O	O	O	O	O	1	1.00
3	4	O	O	O	O	O	O	O	O	O	2	0.00
4	5	X	O	X	O	O	O	O	X	O	3	0.67
5	6	X	O	O	O	O	O	O	O	O	2	0.50
6	7	X	X	O	O	O	O	O	O	O	2	1.00
7	8	X	O	O	O	O	O	O	O	O	2	0.50
8	9	O	O	O	O	O	O	O	O	O	2	0.00
9	10	X	O	O	O	O	O	O	O	O	1	1.00
10	11	X	O	O	O	O	O	O	O	O	3	0.33
11	12	X	O	O	O	O	O	O	O	O	1	1.00
12	13	X	O	O	O	O	O	O	O	O	1	1.00
13	14	X	X	X	O	O	O	X	O	X	5	0.81
14	15	X	X	X	X	X	X	O	X	O	8	0.96
15	16	O	O	O	O	O	O	O	O	O	6	0.00
16	17	X	X	O	O	X	O	O	O	X	5	0.71
17	18	O	O	O	O	O	O	O	O	O	3	0.00
18	19	X	X	X	X	X	O	O	O	O	7	0.71
19	20	X	O	O	O	O	O	O	O	O	5	0.20
MAP: 0.6195												

Figure 4: Judgement results for LSE

To get more insights into the strengths and weaknesses of LE compared to LSE, we compared the Average Precisions by topic and took a closer look into the topics with count of expected relevant documents greater than five and large deviation between the values (>0.4) (Figure 5).

For topic 14 (‘clean laptop’) and topic 15 (‘clean shoes’) we assume that our weighting scheme is responsible for the fact that LSE outperforms LE. Some of the relevant documents LSE retrieved that LE could not find were marked as answers, not questions. We assume that the high weight we assign to title and body scores documents irrelevant to the topic higher than relevant documents with the corresponding keywords in answers and comments. Consequently, these relevant documents disappear from the top 10 hit list.

In topic 16 (‘no wrinkles without iron’) and topic 20 (‘noisy neighbors’) LE obtained significantly better Average Precisions. From the hitlists we can see that LE’s relevant search results for the most part do not contain the query terms, but synonyms. Regarding topic 16, all relevant documents contain the word ‘creases’ which is a synonym of ‘wrinkles’ and the correct term to use for lines and ridges in clothes. LSE on the other hand was not able to retrieve a single document due to the query term’s language misuse. In case of ‘noisy neighbors’, LE also consider the word ‘loud’, which represents more of a common usage.

Topic	Expected # of rel. docs	Average Precision LE	Average Precision LSE	Difference
1	5	0.81	1.00	-0.19
2	1	0.00	1.00	-1.00
3	1	1.00	1.00	0.00
4	2	1.00	0.00	1.00
5	3	0.33	0.67	-0.34
6	2	0.50	0.50	0.00
7	2	0.42	1.00	-0.58
8	2	1.00	0.50	0.50
9	2	0.83	0.00	0.83
10	1	1.00	1.00	0.00
11	3	0.29	0.33	-0.04
12	1	1.00	1.00	0.00
13	1	1.00	1.00	0.00
14	5	0.40	0.81	-0.41
15	8	0.52	0.96	-0.44
16	6	0.92	0.00	0.92
17	5	0.58	0.71	-0.13
18	3	0.57	0.00	0.57
19	7	0.77	0.71	0.06
20	5	0.91	0.20	0.71

Figure 5: Comparison of both systems' Average Precision by topic

Overall, these results indicate that our system reaches a higher ranking efficiency than LSE. Especially LE’s query enhancement seems to contribute considerably to the better MAP.

4. Discussion

The present report aimed at documenting the Lifehack-Engine and evaluating its retrieval efficiency. Our main question was whether our system is able to outperform LSE's retrieval efficiency measured by the Mean Average Precision.

This report has shown that our system was able to reach a higher efficiency than the documents' original source. We discovered that the major cause for this was our system's query enhancement by query terms' synonyms. The results also indicate that the chosen weighting scheme was not optimal. In cases, where documents are relevant because of the answers' and comments' content, our system was not able to assess this appropriately. For further endeavors, we propose to find a more balanced weight assignment.

A fact that we disregarded in the evaluation concerns the asynchronicity of the datasets of the two systems. Since the data set we used comes from a third source, it is not 100 percent up to date. As of today (13/8/2019), the website Lifehacks Stack Exchange counts 2.511 question threads, whereas we are indexing 2.375 documents. This could cause LSE to find more relevant documents for some topics because they are simply not in our record. However, since the absolute difference is only 136 documents, we estimate the effect to be rather small and therefore negligible. Nonetheless, it is always possible to replace the zip file with a recent version of the website archive.org.

5. References

- Lifehacks Stack Exchange [WWW Document], n.d. . Lifehacks Stack Exch. URL <https://lifehacks.stackexchange.com/> (accessed 8.13.19).
- Lucas Fijen, 2017. A Search Engine for Stack Exchange based on ElasticSearch [WWW Document]. URL https://github.com/lucasfijen/Search_it_like_its_hot (accessed 8.13.19).
- Saucier, Z., 2014. What is a lifehack? [WWW Document]. Lifehacks Meta Stack Exch. URL <https://lifehacks.meta.stackexchange.com/questions/8/what-is-a-lifehack> (accessed 8.13.19).
- Similarity module | Elasticsearch Reference [7.3] | Elastic [WWW Document], n.d. URL <https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html> (accessed 8.13.19).
- Stack Exchange Inc., 2019. About - Stack Exchange [WWW Document]. URL <https://stackexchange.com/about> (accessed 8.8.19).
- Webber, W., 2009. When did the Cranfield tests become the “Cranfield paradigm”? « Evaluating E-Discovery. URL <http://blog.codalism.com/index.php/when-did-the-cranfield-tests-become-the-cranfield-paradigm/> (accessed 8.8.19).

6. Appendix

Topic list

Topics for Lifehack-Engine Performance Test

```
<topic number="01" type="single">
  <query>
    remove dog hairs
  </query>
  <description>
    How can dog hairs be removed easily and without special
    equipment?
  </description>
  <narrative>
    Relevant documents must include clear instructions on how to get
    rid of dog hairs on fabric like clothing, carpets, couches, car
    seats etc. They should include a list of the tools, chemicals if
    necessary and how to use them. These items should be part of a
    normal household.
  </narrative>
</topic>

<topic number="02" type="single">
  <query>
    homemade insecticide
  </query>
  <description>
    What homemade recipe can be used to kill aphids and other bugs
    harmful for plants?
  </description>
  <narrative>
    Relevant documents must include instructions and ingredients
    needed to produce something that can be applied on plants in order
    to protect them against insects.
  </narrative>
</topic>

<topic number="03" type="single">
  <query>
    clean cap
  </query>
  <description>
    How to clean a cap?
  </description>
  <narrative>
    Relevant documents must include instructions and chemicals for
    cleaning a cap. They should inform you, if using a washing machine
    would damage the cap or not.
  </narrative>
</topic>

<topic number="04" type="single">
  <query>
    food against headache
  </query>
  <description>
    Which food or drink can reduce headache?
  </description>
  <narrative>
    Relevant documents must include tips to reduce headache. These
    tips can be either something to eat or to drink and should also
    include medicine.
  </narrative>
</topic>
```



```
<topic number="05" type="single">
  <query>
    save water
  </query>
  <description>
    How can I save water in my everyday life without severely
    restricting myself?
  </description>
  <narrative>
    Relevant documents can include all kind of tips that help saving
    water. Documents include recommendations for water saving actions
    in the private home, at work or in public. They can also include
    fixes for common water wasting circumstances like dripping water
    taps.
  </narrative>
</topic>
```

```
<topic number="06" type="single">
  <query>
    remove stone of olive
  </query>
  <description>
    How to remove stone of olive efficiently
  </description>
  <narrative>
    Relevant documents must include techniques and tools for removing
    the stone of olives efficiently. They must also include simple
    standard tools, which everyone owns as well as an olive pitter.
  </narrative>
</topic>
```

```
<topic number="07" type="single">
  <query>
    boil an egg
  </query>
  <description>
    methods how to boil an egg
  </description>
  <narrative>
    Relevant documents must include methods for hard-boiled and soft-
    boiled eggs. They must also point out the tools which can be used,
    like cooking water, a microwave or an oven.
  </narrative>
</topic>
```

```
<topic number="08" type="single">
  <query>
    cut onions without tearing up
  </query>
  <description>
    ways to cut an onion without tears
  </description>
  <narrative>
    Relevant documents must include ways how to cut the onions and
    tools which prevent you from tearing up, like glasses or a sharp
    knife.
  </narrative>
</topic>
```

```
<topic number="09" type="single">
  <query>
    how to defrost quick
  </query>
  <description>
    reduce the defrost time for meat and vegetables
  </description>
  <narrative>
    Relevant documents must include techniques and necessary tools to
    defrost meat and vegetables quickly, like hot water, an aluminum
    plate or a defrost mode on the oven.
  </narrative>
</topic>
```

```
<topic number="10" type="single">
  <query>
    peeling garlic
  </query>
  <description>
    how to peel garlic efficiently and fast.
  </description>
  <narrative>
    Relevant documents must include tips and techniques which make
    peeling garlic easier. They must also include tools which can be
    helpful for peeling and a concrete instruction.
  </narrative>
</topic>
```

```
<topic number="11" type="single">
  <query>
    get rid of acne
  </query>
  <description>
    how to remove pimples and heal acne
  </description>
  <narrative>
    Relevant documents must include methods, cremes, washing routines,
    medicine and chemicals which can be helpful to reduce pimples and
    heal acne.
  </narrative>
</topic>
```

```
<topic number="12" type="single">
  <query>
    put toothpaste back in tube
  </query>
  <description>
    avoiding to waste toothpaste
  </description>
  <narrative>
    Relevant documents must include methods and instructions how to
    get the extra toothpaste back in the tube, like using a syringe.
    They must also include ideas of storing toothpaste aside from its
    common packaging.
  </narrative>
</topic>
```

```

<topic number="13" type="single">
  <query>
    cook in electric kettle
  </query>
  <description>
    What (besides water) can be cooked in an electric kettle
  </description>
  <narrative>
    Relevant documents must include food that can be prepared using
    only an electric kettle and things to take into account when
    preparing the food.
  </narrative>
</topic>

<topic number="14" type="single">
  <query>
    clean laptop
  </query>
  <description>
    How can a laptop (keyboard and screen) be cleaned efficiently
  </description>
  <narrative>
    Relevant documents must include instructions and means to clean a
    complete laptop or specific parts (e.g. screen, keyboard,
    touchpad) of it.
  </narrative>
</topic>

<topic number="15" type="single">
  <query>
    clean shoes
  </query>
  <description>
    How can shoes of different colours and materials be cleaned?
  </description>
  <narrative>
    Relevant documents must include instructions, tips and material
    needed to clean shoes of different materials (e.g. leather or
    fabric) and colours.
  </narrative>
</topic>

<topic number="16" type="single">
  <query>
    no wrinkles without iron
  </query>
  <description>
    How can one make a shirt wrinkle-free without ironing it?
  </description>
  <narrative>
    Relevant documents must include instructions and tips how to
    remove creases from a shirt without ironing it. They can also
    include tips on how to prevent shirts from becoming crumpled in
    the first place.
  </narrative>
</topic>

```

```

<topic number="17" type="single">
  <query>
    store fruits
  </query>
  <description>
    How can fruits and vegetables be stored so they last long?
  </description>
  <narrative>
    Relevant documents must include tips on preprocessing or storage
    on fruits and vegetables sin order to expand their shelf life.
    Relevant documents can also include other tips that makes storing
    fruits more pleasant or more space efficient.
  </narrative>
</topic>

<topic number="18" type="single">
  <query>
    food preparation
  </query>
  <description>
    What are tricks to make food preparation easier, faster or more
    enjoyable?
  </description>
  <narrative>
    Relevant documents can include all kind of tricks helping to save
    time, effort or resources during the process of food preparation.
    Relevant documents can include all kinds of food and meals at
    every step of preparation.
  </narrative>
</topic>

<topic number="19" type="single">
  <query>
    organize shelf
  </query>
  <description>
    How can I organize my shelf or cupboard in a way to save space,
    make it look more tidy or prevent things from being dusty
  </description>
  <narrative>
    Relevant documents can include all kinds of tricks that help to
    organize shelves and cupboards more efficiently. Relevant
    documents can include tricks different objects like books,
    clothing, cd's, accessories, etc.
  </narrative>
</topic>

<topic number="20" type="single">
  <query>
    noisy neighbors
  </query>
  <description>
    What can help me to decrease the noise from the neighbors at my
    place?
  </description>
  <narrative>
    Relevant documents must include tips to ease the disturbance by
    the noise of the neighbors? Documents can include tips that deal
    stopping the noise from the source as well as ideal to seal the
    apartment or room. Source of noise can also be the neighbor's
    dogs, children or a noisy street.
  </narrative>
</topic>

```