Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Master Thesis

An Open-Domain System for Retrieval and Visualization of Comparative Arguments from Text

Matthias Schildwächter
Matrikelnummer: 7003963
MIN-Fakultät
Fachbereich Informatik
Studiengang: Informatik

Erstgutachter: Prof. Dr. Chris Biemann
Zweitgutachter: Dr. Alexander Panchenko

Contents

1. Introduction: An Interface for an Open-Domain Comparative Argumentative Machine (CAM)

What image sensor has less power consumption: CCD or CMOS? Everyone has to choose between different objects from time to time. As in the example from the beginning, this can be a quite specific decision, e.g. for a camera with a certain image sensor, which allows you to use the camera longer without loading, because of its low power consumption. More generally, every informed decision contains one or more comparisons (for example comparing different aspects of the comparison candidates), which then selects one of the compared objects as favorable. Question answering (QA) platforms like `Quora.com`, `Reddit.com`, or `StackExchange.com` contain a large amount of such kind of comparative questions like "How do A and B compare with respect to C", which indicates the human need for comparisons of different kinds. Out there on the web, a user can find a good amount of websites capable of comparisons. For example, many shopping sites nowadays allow users to compare selectable products by their properties or to sort products by criteria which favor one object over another. Other sites like `diffen.com` or `versus.com` are even specialized on comparisons but are still far away from being domain independent.

The goal of this thesis is to satisfy the described human need for object comparisons, by developing a system able to process domain-independent comparative queries and return a preferred object based on a large collection of text documents. To make well-founded design decisions and answer the research question of this work: "*How can an argument-based answer to a domain-independent comparison question be presented?*", three different sources for feature designs are examined. First, recent web-pages able to compare objects (mostly domain dependent) are analyzed and reoccurring features are recorded as comparative input and output pattern. Second, the few scientific works found relevant for argument search and presentation systems are described. As the third source, there are used design guidelines from two books (e.g. [Shneiderman, 2010] and [Johnson, 2014]) to also have a basic view on user interface design in mind. Beyond the newly developed user interface for answering comparative questions, there are two more contributions in this thesis. Firstly, a generic framework for processing comparative question requests (from the interface), extracting relevant data and presenting the answer is proposed (Chapter 3). And secondly, an evaluation of the proposed user interface within a user study, where it is compared to a typical keyword search (Chapter 6), is provided. The evaluation states that a user using the developed system not only can save about 27% time but can also correctly answer more than 15% more comparison questions than with the keyword search.

A demo of the developed comparative question answering system is accessible via the link[1]. Furthermore, the source code is available online[2].

The structure of the thesis is as follows. Related work in form of comparative systems found on the web and previous work on argument search and answer presentation are described in Chapter 2. The backend scheme, its components, and backend feature decisions are described in Chapter 3. In Chapter 4 the initial user interface used in the first study, interface feature decisions and the final user interface are described. The first conducted study and its results are presented in Chapter 5. In Chapter 6 the set-up of the final conducted user study and the evaluation and result discussion are presented. In the end, in Chapter 7 the results are concluded and possible future extensions are described.

[1] http://ltdemos.informatik.uni-hamburg.de/cam/
[2] https://github.com/uhh-lt/cam

2. Background

In the field of argument mining research, the presentation of mined arguments is mostly neglected. To overcome this issue, a wider range of web-based portals able to compare objects is examined with respect to reoccurring output and input pattern. Furthermore, relevant works from the field of argument search, argumentation mining and answer presentation are presented.

2.1. Related Work

In this section, related work in form of comparative systems available on the web or websites that compare objects is described. In total, 25 web pages were examined with respect to reoccurring input and output patterns. Five input and seven output pattern were found. The output patterns sometimes are subdivided into smaller patterns due to similarities. The 25 pages contain a few niche comparison sites, for example, bike insurance and health insurance especially with hospital cover (special insurance for hospital stays), which are strictly domain dependent. In addition, there are pages which are able to compare a wider range of objects like shopping pages, which, however, can still not be considered as domain independent. However, pages like *Diffen*[1] and *Versus*[2] have a good coverage of comparable objects but are still limited due to their data sources, which is based on manually created content like Wikipedia. In addition, manual processing is probably involved, as the possible comparisons are limited. For example, a comparison of *steel* and *iron* is not possible on Versus.com, since both are not available[3].

2.1.1. Comparative Interface Input Patterns

This subsection describes the found user interface input patterns, which occur on pages that compare. The patterns are enumerated with capital letters starting from "A" and names summarizing the content. These letters are subsequently used to refer to the patters.

[1] `https://www.diffen.com` (accessed: 20.06.2018)
[2] `https://versus.com/en` (accessed: 20.06.2018)
[3] Checked on 11.11.2018

A: Selector to Comparison

An example of this first pattern is shown in Figure 2.1.1. The displayed objects provide an "Add to Compare" selector, to chose them for a direct comparison. On the bottom, selected objects are listed. In addition, there it is possible to open the comparison itself. The selection is kept over different search queries, because of that, it is possible to compare all searchable objects (it is even possible to e.g. select a TV and a cell phone for the compare list). The same pattern also can be found in the open source online shop software *Magneto*[4] and in the Wordpress plugin *WooCommerceCompare*[5]. The usage of this pattern in two reusable software packages argues for a wide dissemination.

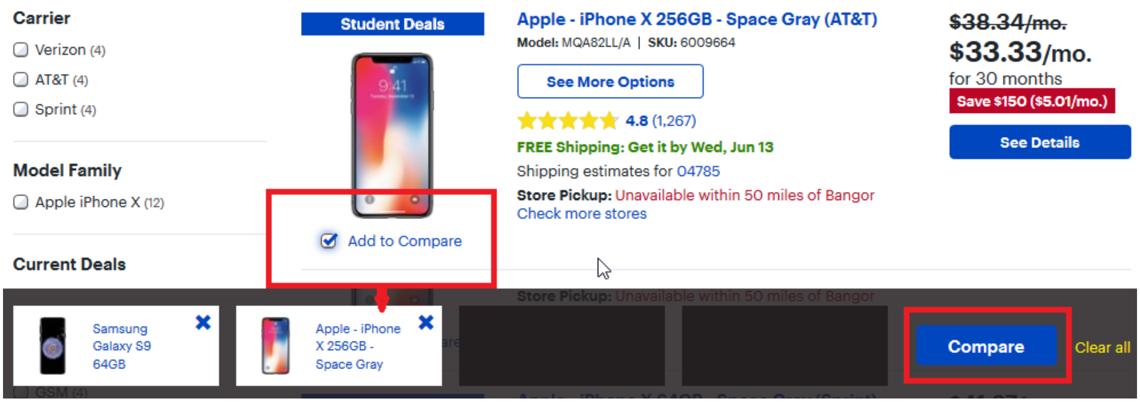


Figure 2.1.1.: The input pattern "A: Selector to Comparison". The screenshot is taken from *BestBuy*[6].

B: Pre Comparison Filter

To use this pattern, the user has to enter some information beforehand to get the right category of objects and to pre-filter them. The pattern, for example, is used at *Bikmo*[7], presented in Figure 2.1.2, here the most important information needed for a bicycle insurance is requested. A similar input pattern can be found at *Tripadvisor*[8], where the user has to enter some information about the desired traveling target to get a sorted list of suggested objects to compare.

C: Specialize Comparison

There is an optional input to make the result more specific. For instance, at *WolframAlpha*[9] (shown in Figure 2.1.3), the query also can be written without "mass", which would

[4] https://docs.magento.com/m1/ce/user_guidemarketing/compare-products.html (accessed: 20.06.2018)
[5] https://de.wordpress.org/plugins/yith-woocommerce-compare (accessed: 20.06.2018)
[6] https://www.bestbuy.com (accessed: 20.06.2018)
[7] https://bikmo.com (accessed: 20.06.2018)
[8] https://www.tripadvisor.com (accessed: 20.06.2018)
[9] http://www.wolframalpha.com (accessed: 20.06.2018)

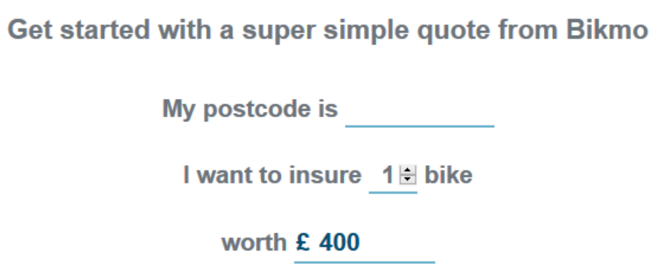


Figure 2.1.2.: The input pattern "B: Pre Comparison Filter" taken from *Bikmo*.

lead to a more general comparison between the two objects. *Check24*[10] also provides this input pattern, for example, the user optionally can enter a postcode to enable a check for availability of DSL bandwidths.

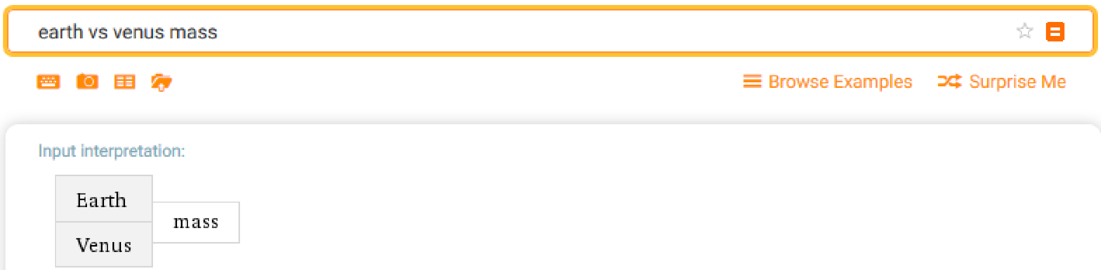


Figure 2.1.3.: The input pattern "C: Specialize Comparison" and "D: What to Compare Fields" taken from *WolframAlpha*.

D: Object Input Fields

The pages using this pattern can have different numbers of input fields to enter objects that should be compared. These fields can use autocomplete features to be certain that entered objects are contained in the data source. The described pattern, for example, can be found in Figure 2.1.3, but also at *StuffCompare*[11], where one input-field per object (maximum 3) is provided.

E: One Field to Best Offer

This pattern provides one input field, which handles one object as input. The system as example searches for this one object in different shops and compares them by price or other criteria. For instance, *Idealo*[12] uses this pattern to directly search for the entered object. However, *Vergleichen.de*[13] also uses this pattern but accesses a variety of comparison pages to display the comparison of multiple pages (like *Idealo*) at once.

[10] `https://check24.de` (accessed: 20.06.2018)
[11] `https://www.stuffcompare.com` (accessed: 20.06.2018)
[12] `https://www.idealo.co.uk` (accessed: 20.06.2018)
[13] `https://www.vergleichen.de` (accessed: 20.06.2018)

2.1.2. Comparative Interface Output Pattern

The following subsection describes the found output pattern of sites that compare.

A: Comparison in Rows

1) The objects are compared based on aspects. “Dimensions” or “Energy Rating” are examples for aspects in the context of televisions. These aspects are shown in an extra line above the displayed comparable object specifications, which are all in one line. For example, the website *John Lewis*[14] uses this output pattern.

2) The aspects to compare object specifications on are displayed in the first column of each line. For instance, the *Maastricht University*[15] uses this pattern to offer comparisons of Bachelor's/Master's programmes.

3) A combination of both: An extra line is used for general aspects and underneath the sub-aspects are presented in the same line as the compared specifications. For instance, in Figure 2.1.4 such format is displayed. In addition to the shown, the website *Kieskeurig*[16] also uses this pattern to compare a variety of different objects (mostly electronic devices).

B: Tiles

1) Shows comparable objects next to each other, only the price can be compared directly. This pattern is used by *Bikmo* to show bicycle insurances in comparison. Basically, three different price categories and their benefits are displayed in a comparable manner.

2) Objects are shown in individual tiles next to each other. The same comparable aspects are shown in each tile. Furthermore, it is possible to receive more information for every single tile. An example of this pattern set-up is shown in Figure 2.1.5.

C: Row Tile Filtering

The objects are shown in lines (long tiles). Each line holds all aspects corresponding to the object (not all aspects necessarily are the same for all objects). Optionally it is possible to reorder the objects by using a sort option based on specific criteria. These criteria fit the comparison domain, for example, “speed” for internet providers and “pay interest” for loans. An example of this pattern is displayed in Figure 2.1.6. The sort option is located

[14] `https://www.johnlewis.com` (accessed: 20.06.2018)
[15] `http://www.comparesbe.nl/compare-bachelor-programmes?p1=1&topic=whatyoulearn` (accessed: 20.06.2018)
[16] `https://www.kieskeurig.nl` (accessed: 20.06.2018)
[17] `https://www.glimp.co.nz` (accessed: 20.06.2018)

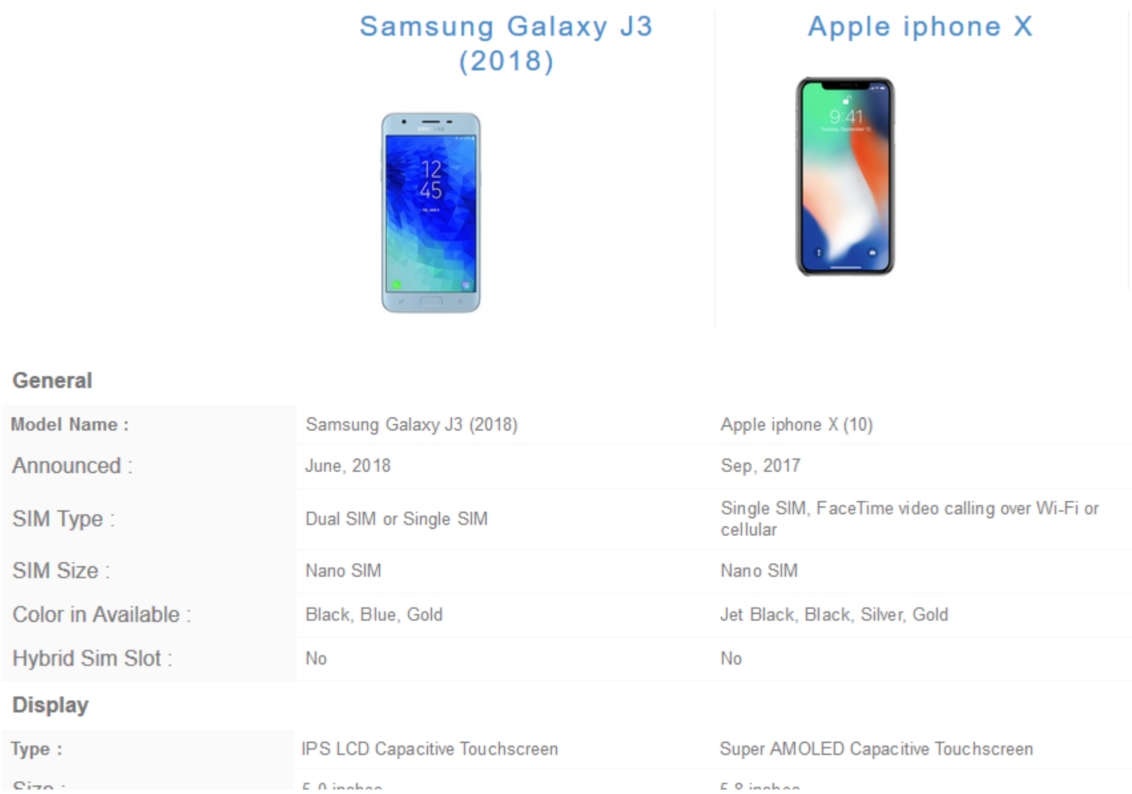


Figure 2.1.4.: The output pattern "A3: Comparison in Rows", taken from *StuffCompare*.

on the top right corner. In addition to the described features of the pattern, there is an option to compare selected objects directly (input pattern *A: Selector to Comparison* and output pattern *A: Comparison in Rows*). *GoCompare*[18] also uses the described pattern, for instance, to compare energy contracts.

D: Two Columns

Dareboost[20] offers a comparison between two websites. In Figure 2.1.7 an excerpt of the presentation is given. The aspects are shown in the middle above the corresponding comparison of this aspect. Either two columns containing the values of the corresponding object or a diagram comparing sub-aspects are shown. None of the other examined websites has a similar format.

E: Visual Representations

Websites using this pattern do not state aspects explicitly, but the specifications can be compared on basis of a visual representation (as shown in Figure 2.1.8). The relation of shown specifications is clear because of the same line. These visual representations, for example, can be bars to represent quality (e.g. the more bars the faster the processor)

[18] `https://www.gocompare.com` (accessed: 20.06.2018)
[19] `https://www.verivox.de` (accessed: 20.06.2018)
[20] `https://www.dareboost.com/en/compare` (accessed: 20.06.2018)

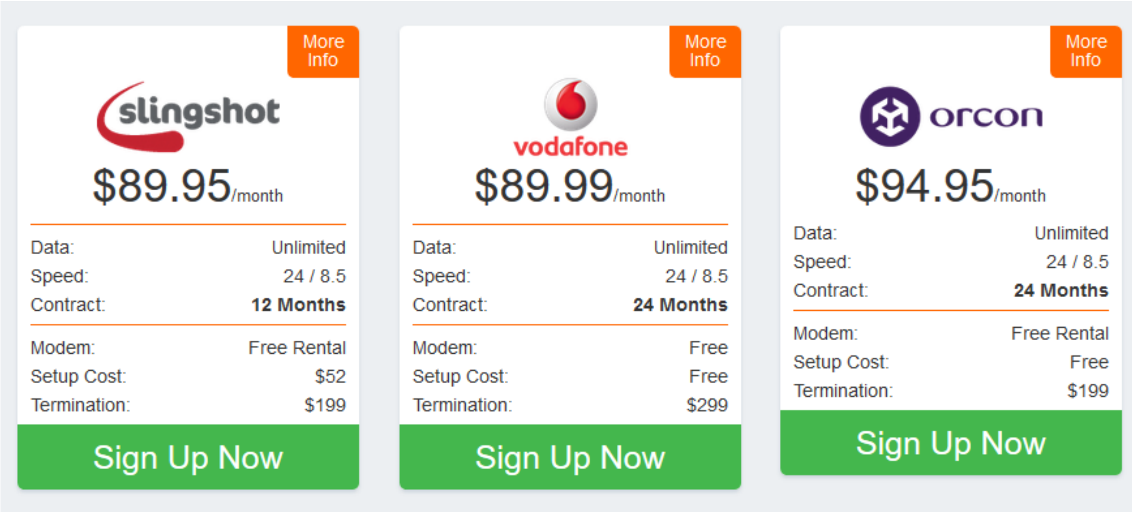


Figure 2.1.5.: The output pattern "B2: Tiles", it was taken from *Glimp*[17].

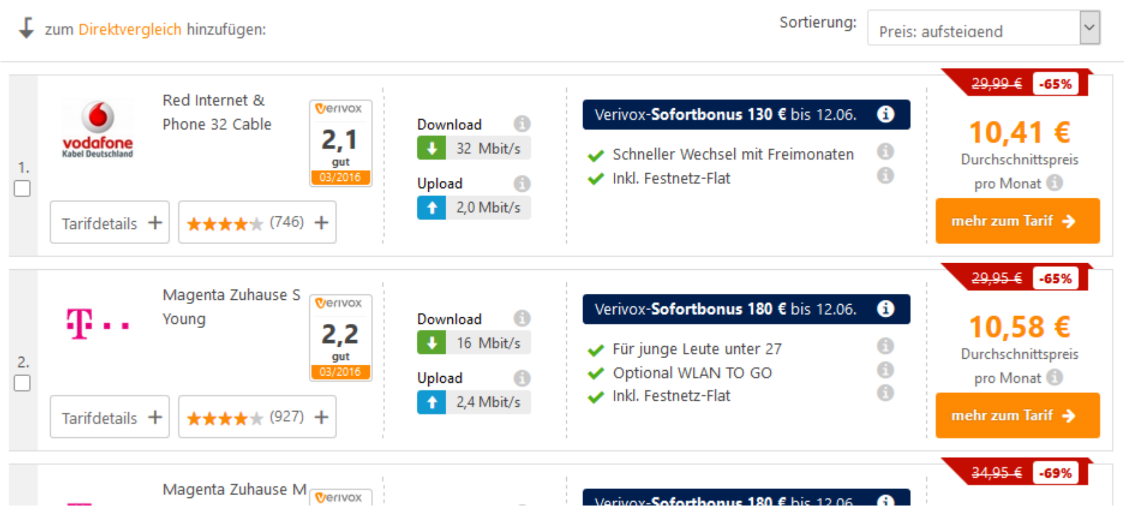


Figure 2.1.6.: The output pattern "C: Row Tile Filtering". Furthermore, a possibility to compare two objects directly (output pattern "A: Comparison in Rows") is given. The screenshot was taken from *Verivox*[19].

used by *Dell*[21] or stars to show user ratings used by *Idealo*.

F: Overview to Detailed

This pattern describes the format used by *Slant*[22]. To begin the comparison, a first overview containing the first three places (based on a calculated score) is shown. The first small overview is followed by a second bigger one, which shows a few of the compared objects, but can be expanded to the full list. On this second overview, a few important aspects and the corresponding values of the listed objects are displayed. An example of this second overview is displayed in Figure 2.1.9. In the end, there is no direct comparison, but a listing of pro and contra arguments of the object, which is displayed in

[21] `https://www.dell.com/en-us` (accessed: 20.06.2018)
[22] `https://www.slant.co` (accessed: 20.06.2018)

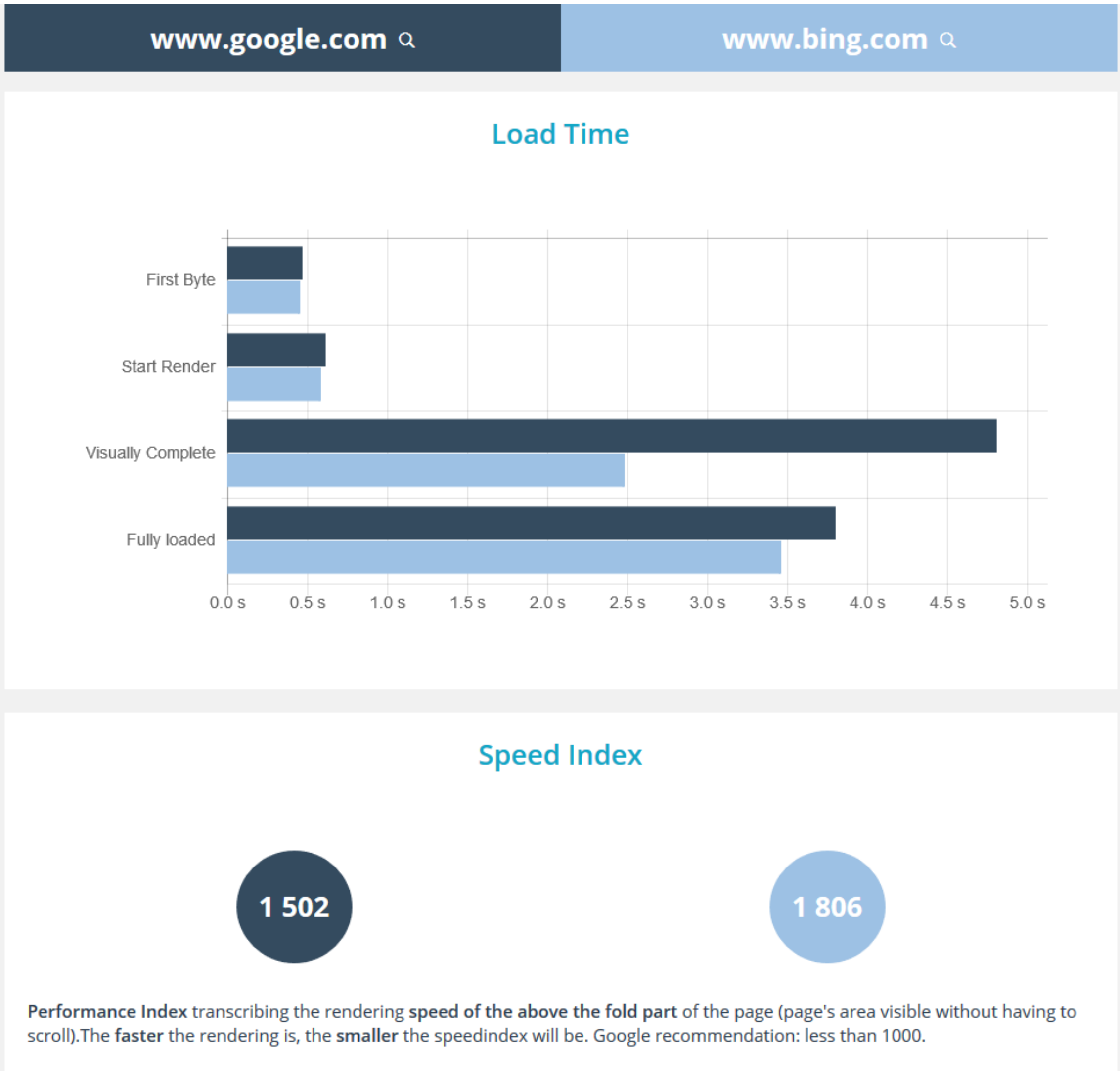


Figure 2.1.7.: The screenshot was taken from *Dareboost* and it shows the output pattern "D: Two Columns".

Figure 2.1.10. The score (top left corner in Figure 2.1.10) is the difference of the number of pro-arguments against the number of contra-arguments. The objects are sorted by this score. Basically, this representation has similarities to output pattern *C: Row Tile Filtering*, where objects are sorted by selected criteria.

G: Aspect Dependent Part Comparisons

A ranking, based on ratings of people, is shown on the top in the comparison presentation of *Versus*. The different colors of the described rankings, later on, are used to distinguish the objects. Directly under the first comparison step, key aspects of the selected objects and the corresponding possessions are shown, as presented in Figure 2.1.11. However, Figure 2.1.12 displays the direct comparison, which is shown underneath on the website. The facts (aspects) are shown on the left with a short description. On the right, the objects are displayed in the user chosen input-order, with applicable aspects displayed per

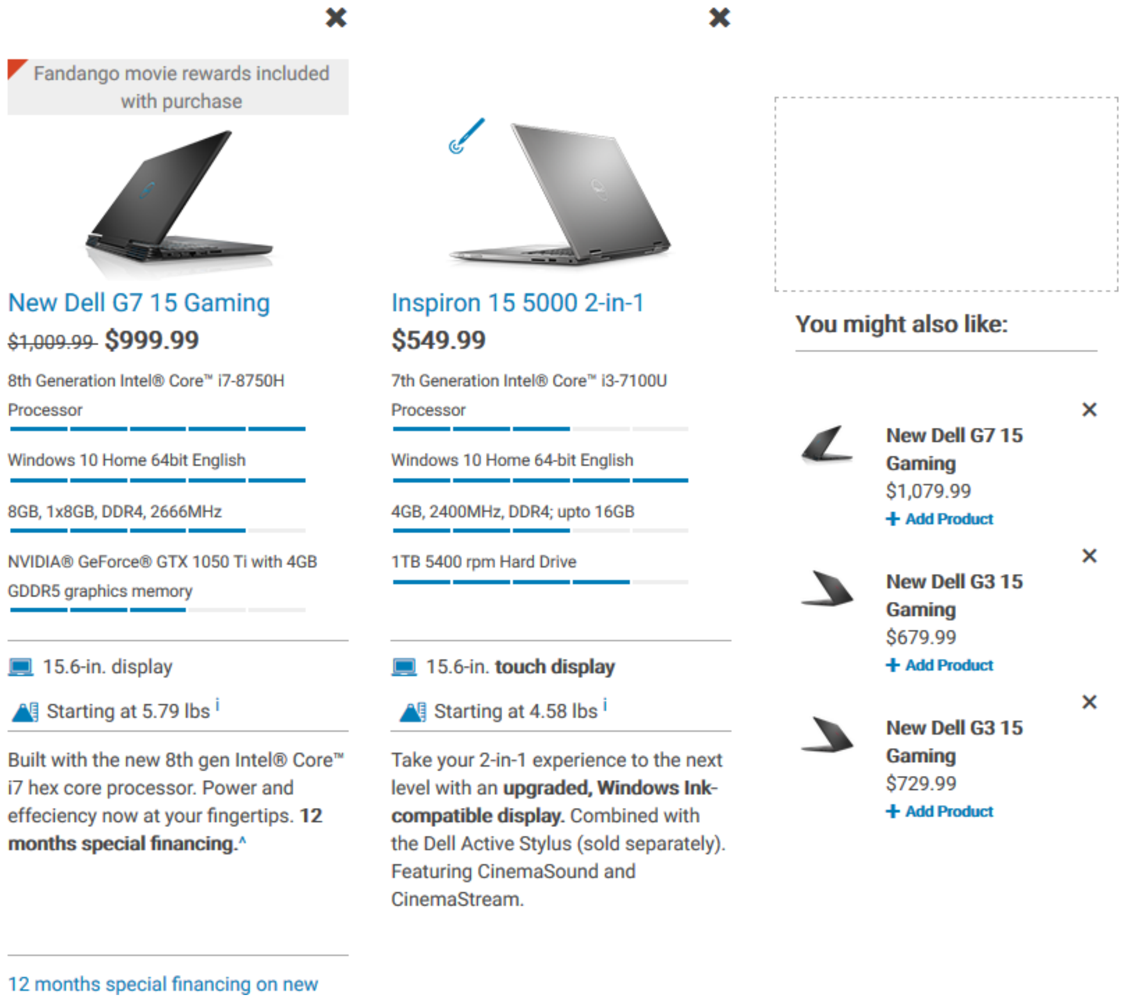


Figure 2.1.8.: The output pattern "E: Visual Representations", taken from *Dell*.

object. In addition, it is shown how many percents of similar objects hold this aspect. The described format is classified as own output pattern, but it holds some similarities to other patterns. For example, the direct comparison with aspects is similar to the above described bigger overview of *Slant*, as in Figure 2.1.9.

2.1.3. Summary

Every examined website has at least one assigned input and output pattern. Table 2.1.1 summarizes the assignments. A few of the patterns are only observed on a single webpage, due to the uniqueness of the presentations. But, even on these presentations similarities to other patterns exist, as described above.

[23] `https://comparebox.pk` (accessed: 20.06.2018)
[24] `https://www.healthpartners.com.au` (accessed: 20.06.2018)
[25] `https://www.moneysavingexpert.com` (accessed: 20.06.2018)
[26] `https://www.trivago.de` (accessed: 20.06.2018)

Website	Input Pattern	Output Pattern
BestBuy	A	A1
John Lewis	A	A1
Magneto	A	A1
Comparebox[23]	D	A2
Maastricht University	D	A2
Diffen	D	A2
HealthPartners[24]	B	A2
WolframAlpha	D + C	A2
WooCommerceCompare	A	A2
Kieskeurig	A	A3
StuffCompare	D	A3
Bikmo	B	B1
Glimp	B	B2
GoCompare	B	C
MoneySavingExpert[25]	B	C
Tripadvisor	B	C
Trivago[26]	B	C
Check24	B + C	C + A2
Verivox	B	C + A2
Dareboost	D	D
Dell	A	E
Idealo	E	E
Vergleichen.de	E	E
Slant	E	F
Versus	D	G

Table 2.1.1.: This table summarizes the assigned patterns to the websites. The table is sorted by output pattern since that is the most interesting column according to the goal of this thesis.

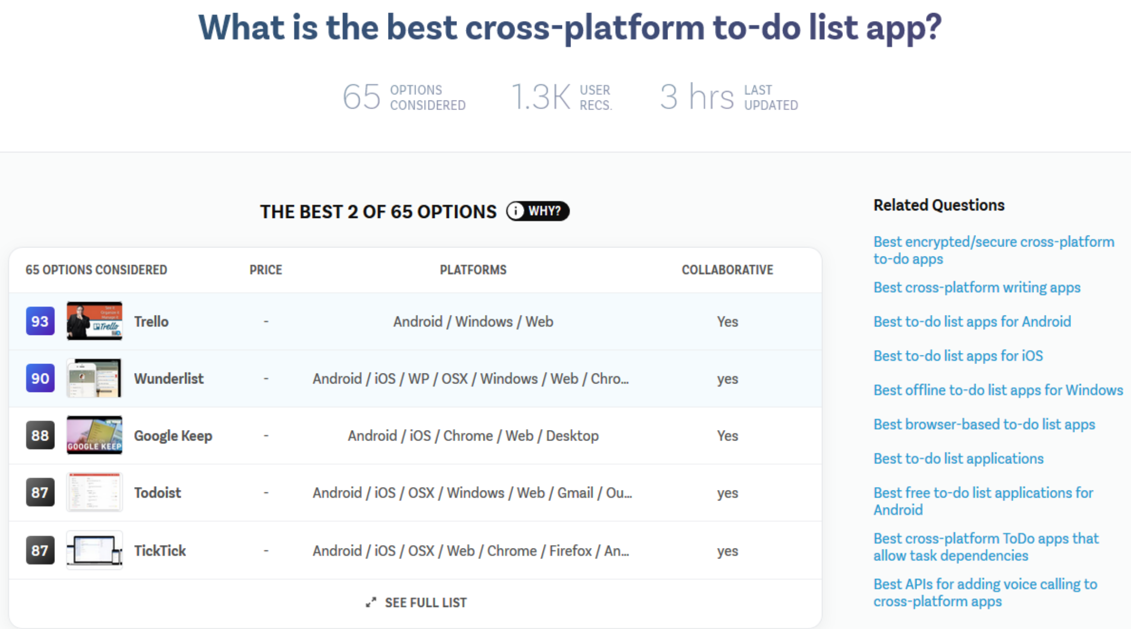


Figure 2.1.9.: Output pattern "F: Overview to Detailed" especially describes the output format of *Slant*, since there was no comparable webpage found, which shared the same format.

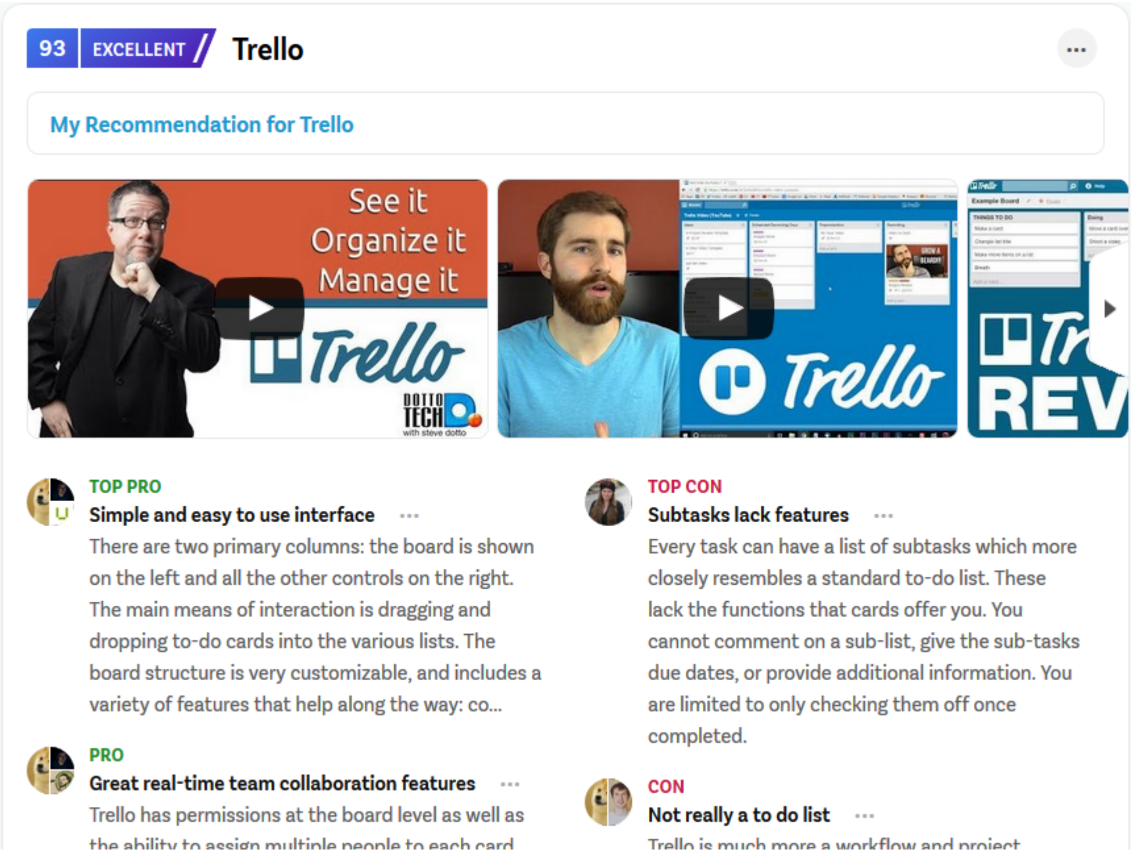


Figure 2.1.10.: Showing the second part of output pattern "F: Overview to Detailed" used by *Slant*. This example exhibits similarities to the one of pattern "C: Row Tile Filtering".

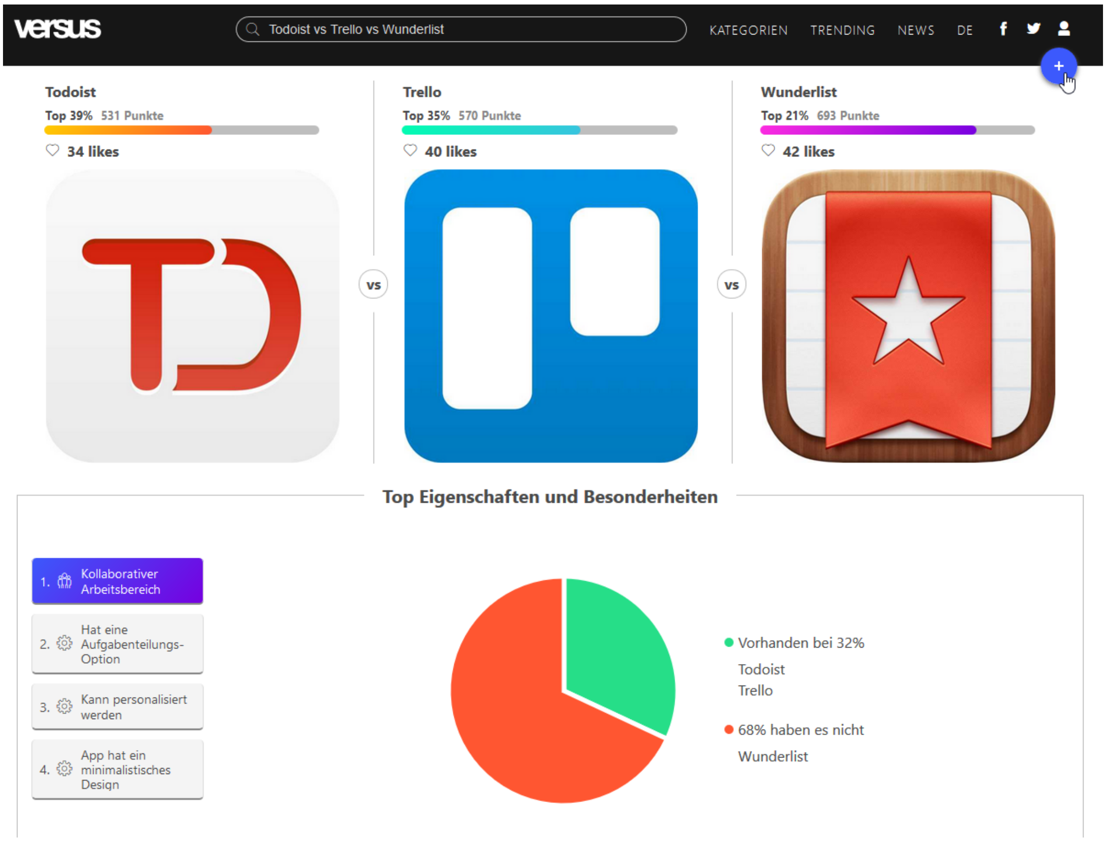


Figure 2.1.11.: The format of *Versus*. Due to its unique representation, *Versus* was classified as independent output pattern "G: Aspect Dependent Part Comparisons".

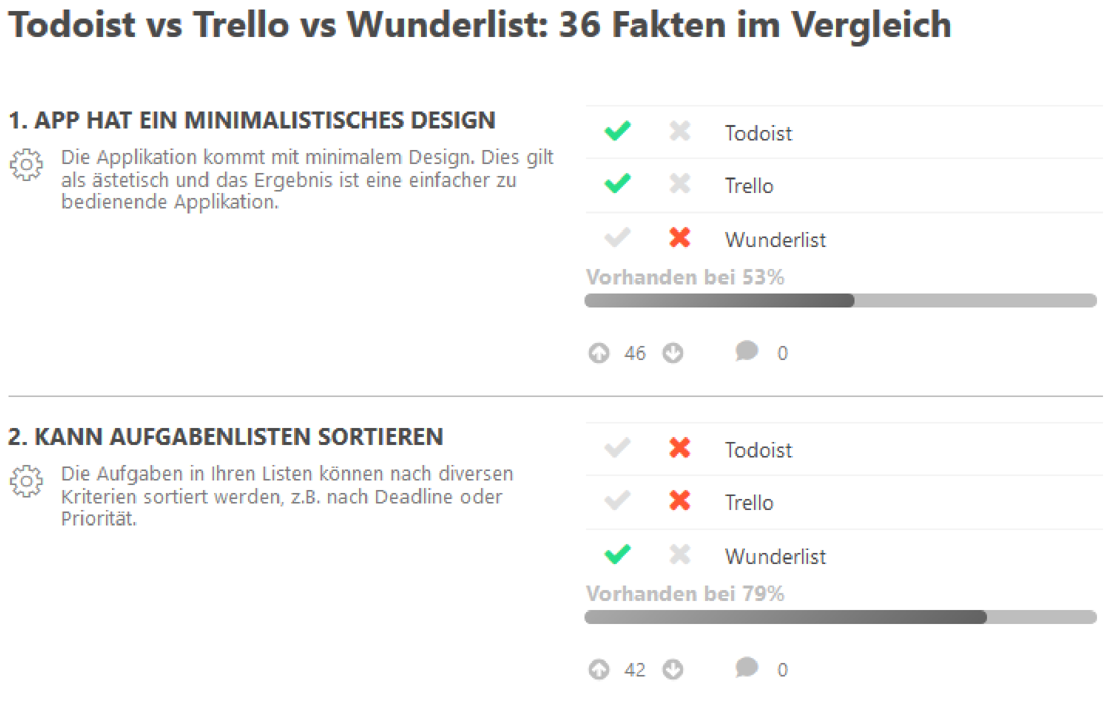


Figure 2.1.12.: The direct comparison of output pattern "G: Aspect Dependent Part Comparisons".

2.2. Argument Search Systems

In the following, publications on argument search and developing argument search systems are reviewed. If a user interface is part of the development, it is shortly described.

A recent approach, similar to that of this work, is presented in [Stab et al., 2018]. They developed a system (called *ArgumenText*) able to query arguments for a user-defined topic. Their system uses an analogous approach as described in [Panchenko et al., 2018] for preparing a data source. To be able to use a less time-consuming argument mining approach that does not need to consider the user-defined topic, they first query relevant documents and afterward apply argument mining to the top-ranked sentences. They use the confidence of their developed argument extraction and stance recognition model to sort the retrieved arguments. To evaluate, they compare the system's output to expert-created argument summaries of an online debate platform (`ProCon.org`). They achieved a high system coverage, but a low precision. The interface developed in [Stab et al., 2018] has a few options to display the queried arguments, it, for example, is possible to filter the arguments by URL or present them in one list.

The goal of [Hua and Wang, 2017] is to find the most relevant argument and the type of the argument in an article supporting an assigned claim. The authors collected and annotated a corpus from *Idebate* (an online debate website) to train a classifier to achieve this goal. Their approach is able to operate on different text types like "blogs", "news" and "scientific". However, the approach is limited to sentential "pro" arguments.

In [Wachsmuth et al., 2017] the authors had the intention to develop an environment for collaborative research on the topic of computational argumentation. First, they built an argument index on basis of mined pre-structured arguments from debate portals. The generic framework for argument search they built relies on that data source. In contrast to many other scientific papers, they also built an interface to present the queried arguments. The default view displays the pro and con arguments separately, opposing each other. The interface is kept just like of a standard search engine, since that is what they wanted to create, an argument search engine (basically a search engine extension). Another approach for such a search engine extension or adaptation in case of argumentation is presented in [Sun et al., 2006]. It is possible to send a pair of search queries, both queries are evaluated by a standard search engine resulting in two page lists. The page lists are combined to one comparative page pair list based on the information pages share and their relevance. The authors introduce two different versions of an interface: A pair-view output, where the pairs are displayed just as in a standard search interface, but with pairs and a cluster-view output where the pairs are clustered by similar topics.

A domain-specific approach, which may nevertheless have some interesting features for the presentation of comparative answers, is described in [Lamy et al., 2017]. This publication develops a user interface for comparing new drugs with existing ones. The authors, for example, used rainbow boxes (designed in [Jean-Baptiste et al., 2016]) and dynamic tables as features for comparison. The developed comparison website was well accepted by the physicians who took part in their user study.

A visual way to compare different products sharing the same aspects was developed in [Riehmann et al., 2012]. In the paper, a multidimensional representation (parallel coordinates) was used to be able to show nearly all aspects to compare on at once. Furthermore, the authors introduced features like a decision bar to store finalized aspects and an attribute repository to cope with multiple dimensions. A visual query is used to find the best fitting object. For example, it is possible to define a range on an axis, to select a value a product should have. Fitting objects are shown on top of the given list. The study to evaluate the system exhibits promising results.

A recent publication on the evaluation of multidimensional representations for decision-making is [Dimara et al., 2018]. The authors compared three different of these representations. The one used in [Riehmann et al., 2012] also is part of the comparison. Overall the authors wanted to explore how to evaluate such representations with respect to their ability to support decisions. They used different tasks and various accuracy metrics.

2.3. Answer Presentation

Although there are many papers that describe argument mining approaches (e.g. [Gupta et al., 2017]; [Lippi and Torroni, 2016]; [Park and Blake, 2012]; [Daxenberger et al., 2017]), the also important topic of presenting the mined arguments or comparative answers is rarely processed in publications. In the following, papers in a more general scope of presenting retrieved answers are reviewed.

For example in [Lin et al., 2003], the authors evaluated four different sizes of presentations for question answering systems, but not for an argumentative answer. It is stated that users need fewer questions to answer a multi-question scenario if more context is given. Furthermore, the authors found out that the more uncertain the source of the answer is, the more context around the exact answer is needed for the user to be certain. An exact answer without any further context (like the sentence containing the answer) was not persuasive enough in most cases. In addition, the expatiated study showed, that users do not care too much about the source of the result.

The influence of snippet length for informational and navigational tasks in a search engine result presentation is analyzed in [Cutrell and Guan, 2007]. As in [Lin et al., 2003], the paragraph-sized snippets lead to the fastest completion times and to the highest accuracy for informational search tasks. The authors state, that searchers looked at a third fewer results in comparison to short snippets. Furthermore, they determined an implicit trust for the ranking of search engines (also known as top-result-bias), which increases the importance of the ranking.

In [Dumais et al., 2001], seven interfaces were created to evaluate two different organizational structures of search result presentations. Some of these interfaces were enriched and grouped with contextual information (category of the results). The authors found out, that the result presentation with additional category information and grouping conveys information significantly faster than a list-based presentation. Furthermore, they discovered that it also speeds up the time to solve tasks when a summary is shown inline as opposed to using a hover-text. In this, but also in [Lin et al., 2003] (described above), the focus-plus-context pattern is mentioned and used. It is further described in [Leung and Apperley, 1994].

A way to enrich generated answers by contextual information is presented in [Perera and Nand, 2015a] and [Perera and Nand, 2015b]. In these papers, the presented answers are not comparative ones, nevertheless, the described methods to enrich presented answers can be of use for this work. The authors describe two different approaches to rank received triples. E.g. token similarity and TF-IDF are used in a bag-of-words approach, whereas e.g. Latent Semantic Analysis is used for a bag-of-concepts approach. The triples shall serve as enrichment for answers. For retrieving these, they use DBPedia[27]. Due to the not too promising results, the authors further investigate using pragmatic aspects, e.g. pragmatic intent and pragmatic inference, to select the triples in [Perera and Nand, 2015b].

Just as the developed representation in [Riehmann et al., 2012], the front-end developed in [Diefenbach et al., 2017] shall serve as a (domain independent) reusable front-end. To obtain this goal, the authors use some general features to display a retrieved answer. For instance, (if available) an image or external links are displayed. However, the developed front-end is only compared (in a feature-based fashion) to some question answering front-ends, but not evaluated in any way.

The paper [Hoque et al., 2017] builds a question answering system based on forum data. In addition to the retrieval of relevant information based on user comments, a user interface is created to support the exploration of the mined data. In comparison to [Lin et al., 2003] and [Dumais et al., 2001], in this visualization, an overview+detail (described in [Cockburn et al., 2009]) instead of a focus+context approach is chosen. Furthermore, different features like filters and colored labels are used to express the relevance of displayed answers and to be able to compare them.

A general approach for improving natural language interfaces is described in [Joshi and Akerkar, 2008]. A rule-based approach is used to cope with semantic symmetry and ambiguous modification. The authors managed to improve the precision of their developed question answering system by dealing with those two language phenomena occurring in English, using their developed approach. In the work, the system further is compared to other question answering systems. It is efficient enough for online scenarios,

[27]DBPedia `https://wiki.dbpedia.org` (accessed: 27.06.2018)

since merely a part-of-speech tagger is required for preprocessing.

2.4. (Comparative) Question Answering

The following publications describe the process of finding the correct answer to a question (including comparative questions).

A system able to process an answer for comparative questions is developed in [Moghaddam and Ester, 2011]. They call it "aspect-based opinion question answering" and take reviews as a dataset, to derive an answer from. Five phases to determine fitting answers are created. For comparative questions, common aspects of the targets with higher rating difference are selected to gather relevant sentences. The rating is based on a method described in [Moghaddam and Ester, 2010], in which reviews are parsed to pairs (<aspect, rating>).

Furthermore, there are some closed domain question answering systems for comparative question answering, which can deliver some useful information to build features.

In [Choi et al., 2011] a question answering system for the domain of business intelligence is developed to answer comparative and evaluative questions. Representations (including XML) that are used between the processing steps are described. The data to query an answer from being saved in a database. The presentation module is created to construct a natural language answer. To obtain that WH-movement based on constituency parse trees, user-defined templates and surface realization is used. The authors claim that the system can also be used for other domains if another data set and predicates are used.

An approach for retrieving information to answer comparison questions under the domain of medicine is described in [Leonhard, 2009]. The author first describes how to query comparison questions from a manually created question answering corpus using regular expressions. The so-created comparison question corpus is later on used to evaluate the developed retrieval approach. The pieces of information to answers were queried with help of the objects to compare, a basis of the comparison (basically an aspect to compare on like "fever") and a publication type label.

3. The Backend of the Comparative Argumentative Machine (CAM)

To avoid the notorious coverage and actuality problems that systems relying on structured data are facing, CAM extracts argumentative structures from web-scale text resources to answer questions asking to compare two objects. The extracted argumentative textual statements should then either support that one of the objects is superior to the other, that they are equal, or that they cannot be compared. A comparison of two objects (A and B) in the CAM-sense is defined as triple ($A, B, \mathbf{C}$): "A κ B with respect to $\mathbf{C}$", where $\kappa \in \{>, <, =, \neq\}$ and $\mathbf{C} = \{c_1, \ldots, c_n\}$ is the set of aspects A and B should be compared on. The focus thus is on mining claims stating that an object A is better or worse than an object B with respect to some aspect $c_k \in \mathbf{C}$ like "*Python=A* is better than *Matlab=B* for *web development=c_k*."

The system's output for one entered aspect can be verbalized as a label that summarizes all statements of corresponding sentences (sentences comparing both objects and containing the aspect):

1. *BETTER*: A is better than B with respect to C
2. *WORSE*: A is worse than B with respect to C
3. $\neq$ ($\neq$, $=$): No statement can be given (or A and B are equal)

The statement of a sentence can either be A is *better* than B, A is *worse* than B or no comparison is available/the objects are equal ($\neq$). Note that, the mapping between classes ($>, <, \neq, =$) to the statement *better* ($A > B$ with respect to (wrt.) $\mathbf{C}$, where $\mathbf{C}$ are contained aspects) and *worse* ($A < B$ wrt. $\mathbf{C}$) is not direct. For instance, the sentence "Python is better than Matlab" (class $>$) supports the same statement as "Matlab is worse than Python" (class $<$), since the order of the object matters.

The scheme of the CAM system for building an answer to a comparison is shown in Figure 3.0.1. It consists of the following general stages that are described in the sections below: (1) retrieval of relevant sentences, (2) sentence preprocessing (3) classification of comparative sentences, (4) ranking of the comparative sentences, (5) extraction of object aspects and (6) presentation of the answer.

The chapter is structured as follows: First, the above scheme of the CAM system is described feature-wise (see Figure 3.0.1). The features produce the different parts of the

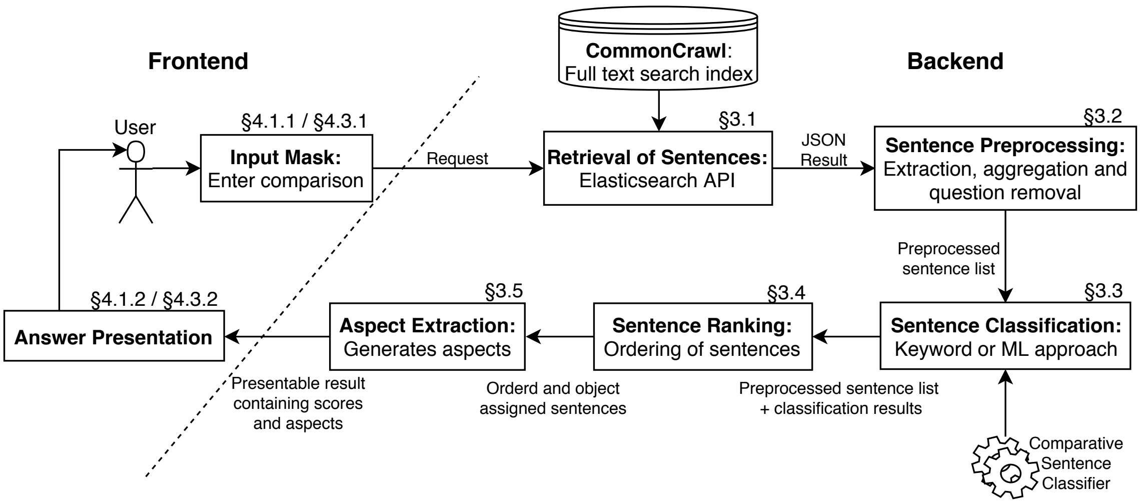


Figure 3.0.1.: The answer processing steps of the CAM version used for the main study. A preprocessing step was added due to the necessary duplicate aggregation step.

answer shown to the user: individual scores corresponding to entered aspects, evidence sentences assigned to the objects and generated aspects. The last section (Section 3.6) of this chapter addresses the main feature decisions and evaluations that lead to the described ones.

3.1. Retrieval of Sentences

The sentence retrieving and score calculation are both executed on a Common Crawl[1] data set. The non-profit organization Common Crawl crawls the web to provide the received data to the public for free. In [Panchenko et al., 2018] this freely available large dataset was used to build a web-scale dependency-parsed corpus. That corpus already was preprocessed, namely only English text was used, (near-) duplicate and HTML tags were removed and the documents were split into sentences. The corpus used in the main study and for the final system still contains duplicates, since there are also valuable information contained. For example, it is assumed that the more documents contain a sentence, the more important the sentence is. Furthermore, that corpus has a document and a sentence id to be able to recreate the document to show the context of a viewed sentence or to reach the original source containing the sentence.

An Elasticsearch index was created for both described corpora to allow the access. The index without duplicates contains 3,288,963,864 sentences, whereas the index without duplicate filtering contains about 14.3 billion sentences. It is also possible to save additional information per sentence. For example, a classification result of the sentence can be saved to the index, to later on speed up the answer preprocessing.

The sentence retrieval of the **default approach** (see Section 3.3.1), is tightly coupled with the classification itself since the used query already retrieves comparative sentences.

[1] `https://commoncrawl.org` (accessed: 09.06.2018)

To do so, sentences containing the tuple (both objects) and one or more marker sequences, are retrieved. The marker sequences are build from a list of comparative words (or word sequences) like *better*, *quicker* or *less secure* in combination with *than*, *alternative to* or *then* to get for example “better AND than” or “quicker alternative to”. The *AND* is used to allow the marker parts to be somewhere in the sentence, not necessarily next to each other. The build markers are disjunctively linked (see Listing 3.1) to get all sentences containing both objects and at least one marker. If the “Faster search” option of the interface is selected for this approach, the maximum number of sentences (size) retrieved is limited to 500 (instead of 10000).

```
http://ltdemos.informatik.uni-hamburg.de/depcc-index/depcc/_search?q=
    ↪ text:"OBJECT_A" AND "OBJECT_B" AND ("superior to" OR "inferior
    ↪ to" OR superior OR inferior OR ("better" AND than) OR ("easier"
    ↪ AND than) OR [...] OR ("better alternative to") OR ("easier
    ↪ alternative to") OR [...]  OR ("better then") OR ("easier then")
    ↪  OR [...] OR ("finer then"))&from=0&size=500
```

Listing 3.1: The query used to retrieve comparative sentences from the Elasticsearch index to classify them further. That query is only used for the marker approach. OBJECT_A and OBJECT_B are placeholders for the first and second object.

To query sentences to use the **approach of machine learning** (described in Subsection 3.3.2), two queries are used. The first (see Listing 3.2 at the top) retrieves sentences containing both entered objects and at least one of the entered aspects to compare the objects on (triple). If no aspect is entered by the user, that first query is skipped. The second (see Listing 3.2 at the bottom) retrieves fall-back sentences containing only both objects to build a more general solution. For both queries at most, the 10,000 most recent (according to the Elasticsearch score[2](*sentence_score*)) sentences are queried to cap the processing time. The number of queried fall-back sentences can be reduced to 500 by selecting the “Faster Search” option of the interface, it allows the user to get a faster solution without changing the outcome for entered aspects.

```
http://[...]/depcc/_search?q=text:/depcc/_search?q=text:"OBJECT_A" AND
    ↪ "OBJECT_B" AND ("ASPECT_1 OR [...] OR ASPECT_n")&from=0&size
    ↪ =10000

http://[...]/depcc/_search?q=text:/depcc/_search?q=text:"OBJECT_A" AND
    ↪ "OBJECT_B"&from=0&size=500
```

Listing 3.2: Both queries are used to retrieve sentences from the Elasticsearch index to classify them in the next step. The query on the top retrieves sentences containing the objects and at least one of the entered aspects. The one at the bottom is used to retrieve fall-back sentences. OBJECT_A and OBJECT_B are placeholders for the first and second object. ASPECT_1 to ASPECT_n are the user entered aspects.

[2] https://www.elastic.co/guide/en/elasticsearch/guide/current/scoring-theory.html (accessed: 19.09.2018)

The structure of the resulting JSON retrieved from Elasticsearch is the same for both approaches. A total number of hits, the maximum Elasticsearch score (used later on for scoring functions) and hits consisting of the sentence-dependent information like the Elasticsearch score, the text, the sentence id, and the document id, are contained.

3.2. Sentence Preprocessing

For both approaches, the queried sentences are extracted from the JSON result. In addition, questions (sentences containing a question mark) are removed, since they will not help the user to compare objects. However, for the final system used in the second study, the larger corpus is used and therefore in addition duplicates need to be aggregated. For aggregation, the id's of all documents containing the same sentence (exact duplicate) are assigned to it to make them accessible in the answer presentation.

After retrieving the sentences as described in Section 3.1 and applying the above-described preprocessing steps, a list of sentence objects is the result. At this stage, such object contains the sentence itself, the Elasticsearch score and id pairs (the document id (source) assigned to the sentence id). In later stages, the object gets enriched with further information. The sentences presented in Figure 3.2.1 are taken from the sentence objects corresponding to the query "*AM* compared to *FM* with respect to *frequency*" retrieved with the approach of machine learning. The sentences are referenced in later stages to clarify the functionality contributed by each feature.

```
1    AM/FM Frequency .
2    "The result is frequency modulation, FM."
3    AM/FM Tuning: rotary knob adjusts AM and FM frequency .
4    "Tuner: Frequency Band: AM, FM."
...
255  Like your thoughts are on AM frequency and I'm only getting FM.
256  "The AM won't come in, despite being lower frequency than the FM."
257  "you must specify the AM/FM frequency step used in your area. """
...
297  "These changes to digital radio do not affect FM, AM or online BBC radio services."
298  "Thus, FM is at a much, much higher frequency than AM, with the lowest frequency on
      the FM dial 55 times as great as the highest on the AM dial."
299  "Frequency is sent as the KHz frequency (550 to 1600) for AM, and the MHz frequency
      (8800 to 10800) for FM."
...
955  Tuning Range Broadcast (AM): 510 - 1620 kHz Short Wave: 5.9 - 16 MHz Frequency
      Modulated (FM): 88 - 108 MHz .
956  The problem is that AM radio requires a much bulkier antenna than FM due to the
      lower frequency.
957  "THE SYSTEM OPERATES ON AMPLITUDE MODULATION (AM) FREQUENCY MODULATION (FM),
      CONTINUOUS WAVE (CW), AND SINGLE SIDEBAND (SSB) SIGNALS."
...
```

Figure 3.2.1.: Sentences queried for the triple: "*AM* compared to *FM* with respect to *frequency*" (ordered by Elasticsearch score).

3.3. Approaches for Sentence Classification

In this step, the sentences are assigned to the first (A) or the second entered object (B), or they are discarded, based on the taken statement of the sentence. They can either be *better*, *worse* or $\neq$ as described at the beginning of this chapter. The statement of given sentences is determined using one of the underneath described sentence classification approaches. In the interface, the user can select one of them to use.

3.3.1. Default Approach: Query with Markers

After the above described sentence retrieval of comparative sentences with markers, the sentences are preprocessed and ranked. If a sentence contains one of the entered aspects its rank is increased. The use of the markers shrinks the smaller corpus to 16,161,110 and the larger corpus to 45,408,377 sentences since only sentences containing at least one of the described markers are queried. The fact, that the list of marker sequences does not capture all possible, means that some sentences, which could be used to compare the objects, are not found. The approach described in Subsection 3.3.2 is able to deliver better results since the majority of comparing sentences are found and used.

3.3.2. Approach of Machine Learning: Classifier

A classifier developed in [Franzek et al., 2018] is used to distinguish between four classes: the first object from the user input is better / equal / worse than the second one ($>, =, <$), or no comparison is found ($\neq$). The classifier mainly uses the text between both objects to identify the polarity. An aspect like "price" is not taken into account for this step. To train and evaluate different classifiers and feature sets, in [Franzek et al., 2018] a dataset of 7199 comparative sentences containing three domains was build and used. *XGBoost* [Chen and Guestrin, 2016] with 1000 estimators was selected, out of thirteen evaluated, as classification method due to the high F1 score. *Gradient boosting* is used as a boosting method and *decision trees* as learners for the *XGBoost* classification model. Furthermore, a variety of different feature sets were evaluated and compared in [Franzek et al., 2018]. The best two feature sets (*Bag-of-Words* and *InferSent*[Conneau et al., 2017], a method to create sentence embeddings), according to F1, are taken for the CAM interface. Both feature types achieved a high F1 score of 0.92 for $\neq$, good F1 of 0.74 for $>$ but pretty bad F1 of 0.39 (InferSent) and 0.46 (BoW) for $<$. The main problem was in handling negations for which a heuristic (see Subsection 3.4.2) was added to the CAM system. Since InferSent-based classification was too slow for real-time requirements of the system, only the about ten times faster BoW-based classifier was used in the conducted studies.

The classifier assigns a classification confidence between zero and one for all polarities except for equal ($>, <, \neq$). The equal polarity is represented by evenly high confidences for $<$ and $>$. If no comparison is found ($\neq$), the sentence is discarded. In Table 3.3.1 the assigned confidences per label are shown for the example sentences of Figure 3.2.1. When assigning the sentences to the objects, the winning confidence is added to the sentence object. The mapping between assigned polarity and winning object depends on the order of the objects in the sentences (as described at the beginning of the chapter). For example, the classifier set $>$ as polarity for sentence 256, but the sentence is assigned to object B (AM) of the example since it occurs first. The classification confidence is used in the next step, the ranking of sentences, to boost highly certain sentences.

Sentence	$>$	$\neq$	$<$	max
1	0.03	**0.95**	0.03	$\neq$
2	0.03	**0.95**	0.03	$\neq$
3	0.03	**0.95**	0.03	$\neq$
255	0.02	**0.92**	0.05	$\neq$
256	**0.8**	0.07	0.13	$>$
257	0.03	**0.95**	0.02	$\neq$
297	0.03	**0.95**	0.02	$\neq$
298	**0.86**	0.07	0.07	$>$
299	0.07	**0.83**	0.1	$\neq$
955	0.03	**0.84**	0.13	$\neq$
956	**0.85**	0.11	0.04	$>$
957	0.03	**0.95**	0.02	$\neq$

Table 3.3.1.: The confidences (and labels) for the sentence examples shown in Figure 3.2.1.

3.4. Sentence Ranking

The function to calculate a score for sentences processed by the **default approach** is based on the Elasticsearch score received when querying the sentences (as described in Subsection 3.6.1). Furthermore, if the regarded sentence contains a user entered aspect, the aspect weight set by the user is used as a multiplier for the score. The last building block of the score is the number of markers a sentence contains, it is also used as a multiplier. In the end, all sentences are ordered by the calculated score.

For the **approach of machine learning** the classification confidence, the Elasticsearch score and the aspect weights (if an aspect is contained) are used to build sentence scores to rank the sentences. Tn the last step, the statement of each classified candidate sentence (*better*, *worse*, or *none*) was used to assign the sentences to the objects. Therefore, at this stage, one ranked list of sentences for each object (A and B) that therefore either support the statement A is *better* than B with respect to **C** or A is *worse* than B with respect to **C** (C represents entered aspects) can be built. The sentence rankings are based on the following score for the i-th sentence:

$$s_i = \begin{cases} \alpha + e_i + e_{max}, & \text{if confidence } > \gamma \\ \beta(\alpha + e_i), & \text{otherwise} \end{cases} \tag{3.1}$$

where e_i is the Elasticsearch score of i-th sentence, e_{max} is the maximum Elasticsearch score, $\beta = 0.1$ and $\alpha = w_{C_k} e_{max}$ if a user-specified aspect C_k is present in the i-th sentence and is zero otherwise ($\alpha = 0$). The α is the aspect boost, where w_{a_k} is the weight of aspect specified in the user interface. Based on observations of the first study and manual examination (see Subsection 3.6.1) the sentence score was selected mainly consisting of the Elasticsearch score. The Elasticsearch score ranks shorter sentences containing the searched keywords (objects) highest. The high score leads to a top positioning of concise sentences since it is later on used to order the sentences. For example, sentence 256 has a higher Elasticsearch score than sentence 298 (see Figure 3.2.1) and therefore will be ranked higher (unless only sentence 298 gets boosted), even though the classification confidence of sentence 298 is higher. The classification confidence is only used to boost sentences, but not to rank them: γ is determined by using the gradually decreasing threshold described in Subsection 3.4.1 with a sentence-threshold (x) of five. Therefore, γ depends on the number of high-quality sentences (high confidence). All sentences with a confidence above γ obtain a boost of e_{max} to their score, which in the end leads to a position at the top of the ranked list. Tests executed on the dataset of the first study showed the highest accuracy for this threshold (see Subsection 3.6.2).

The single sentence scores are summed up in different categories to calculate independent scores for answer presentation. The categories are selected with respect to the aspect(s) contained in each regarded sentence:

$$category(c) = \begin{cases} \text{“General Comparison”}, & c = 0 \\ C_k, & c = 1 \\ \text{“Multiple Aspects”}, & c > 1 \end{cases} \tag{3.2}$$

where c is the number of contained aspects and C_k is the only contained aspect. The total score supporting the statement $A > B$ with respect to $\mathbf{C}$ (as in Figure 4.3.2) is the normalized sum over all sentences ($S_{1 \ldots n}$) (regardless which category) supporting the statement:

$$\sum_{i\text{: supports } A>B \text{ with respect to } \mathbf{C}} s_i \Big/ \sum_i s_i \tag{3.3}$$

In the end, the negation dissolving heuristic, described in Subsection 3.4.2, is used to move due to negation incorrectly assigned sentences.

3.4.1. Gradually Decreasing Threshold

First, the sentences above each step are counted. Second, the threshold according to this count (represented as $c(y)$) is determined using the following formula:

$$threshold(x) = \begin{cases} 0.8, & x < c(0.8) \\ 0.7, & c(0.8) < x < c(0.7) \\ 0.6, & c(0.7) < x < c(0.6) \\ 0.5, & c(0.6) < x < c(0.5) \\ 0, & otherwise \end{cases} \tag{3.4}$$

where x is a number of sentences that should at least be presented to the user, basically another threshold. That new sentence-threshold is now used to determine the confidence threshold: if the set number is five, there should be more than five sentences with a confidence higher than e.g. 0.8 to take 0.8 as the threshold.

3.4.2. Negation Dissolving Heuristic

The heuristic uses contrary comparatives like “bigger” $\rightarrow$ “smaller” to move semantic equivalent sentences to the same object. Sentences are only considered if they exhibit the following pattern: “y (...) A (...) y (...) C_k (...) y (...) B (...) y”, where y is a positive comparative adjective (only one has to occur somewhere), A and B are the objects and C_k is one of the entered aspects (they are processed one after the other). For example, the following sentence exhibits the described pattern:

- “**FM** uses *higher* **frequency** than **AM**”

After such sentence was found, among those assigned to object A, the heuristic looks through the sentences assigned to the other object to find possible negations of the form “z (...) B (...) z (...) C_k (...) z (...) A (...) z”. z is a contrary comparative adjective (if y is “bigger” then z, for example, is “smaller”). The other variables are just as above. If such negated sentence is found, it is moved to the list of object A. The same procedure is executed again starting with sentences of object B. For instance, the following sentences exhibit the described pattern for negated sentences:

- “The **AM** won’t come in, despite being *lower* **frequency** than the **FM**.”
- “The **AM** won’t come in, despite being *lower* **frequency** than the **FM**".”
- “In the US, the **AM** bands are an order of magnitude *lower* in **frequency** than the **FM** bands (AM tops out at 1.705 MHz, FM starts around 88MHz).”
- “So **AM** radio, which operates at *lower* **frequency** than **FM** radio would need an even longer aerial than AM radio would.”

The moved sentences also contain sentence 256 shown in Figure 3.2.1. It was assigned to object *B* (AM), even though there are sentences taking the same statement (but negated) assigned to object *A* (FM).

The performance of the heuristic was manually evaluated on the dataset of the first study. The moved sentences and the impact on the aspect dependent score were examined for this purpose. For the classifier mainly using BoW as a feature, the average gold deviation decreased from 0.37 to 0.36, which is a slight performance increase. In addition, the standard deviation decreased from 0.3 to 0.29. Other measurements like precision or recall kept equal. The sentences moved from one object to the other were manually examined and classified as correctly moved, wrongly moved or pettily moved. For BoW five sentences were wrongly moved, seven were pettily moved (e.g. "In addition, the copper oxide also has good thermal conductivity, and lower price than noble metals, e.g. gold, silver, etc." was moved from *copper* to *gold* although the aspect was *conductivity*.) and 18 were correctly moved (e.g. "It is claimed that the Wii U processor carries a clock speed of 1.24 GHz - less than half the speed of the PS3 and Xbox 360." was moved from *Wii U* to *PS3* for the aspect *processor*.)

For the Infersent feature set, the total accuracy increased 12% and a similar amount of sentences was moved (6 falsely, 1 pettily and 22 correctly moved sentences). To summarize: the heuristic on average brought the scores nearer to the best possible (gold) score and therefore has a benefit for the application.

3.5. Aspect Extraction

For both conducted studies a basic aspect extraction method is used in the system: A Part-of-Speech tagger is used to determine all nouns of the independent sentence lists of the two entered objects. These nouns are further filtered so that they do not contain any stop-words, markers, numbers and some other words like *come*, *much* or *good*, which are considered uninteresting. The frequency of each word is counted per object sentence list. Words contained in both lists are considered as aspect candidates. To decide to which object a found aspect should be assigned, the ratios of candidate frequencies between both lists are calculated. The ten aspect candidates with the highest ratio per object are taken as aspects.

However, the final system contains a more elaborated aspect extraction methodology: There are three different methods CAM uses to extract these aspects. For each of those, for each sentence, the words are classified via a part of speech tagger to find out which of them are nouns, adjectives and so on.

The first method scans each sentence for comparative adjectives and adverbs. Those that do not have any information value by itself are then filtered out – as these aspects are supposed to give reasons as to *why* an object is better than the other, words like *better* are not useful here.

The second method also uses comparative adjectives and adverbs, however, it does not just collect those alone but scans for structures like *easier for (something)* or *faster to (something)*. As the example shows, this can lead to aspects like *quicker to develop code* or *better for scientific computing*. Aspects collected using this method are usually more useful than those from the first method, however, they are also more sparse.

The third method is independent of comparative adjectives and adverbs and instead focuses on other sentence structures such as *because (something)*, *since it has (something)* or *as we have (something)*. If a sentence contains a structure like that, all nouns that follow afterward are collected as aspects.

All aspects are collected for the object that wins the corresponding sentence. When this process is finished each object's aspects are ranked by dividing their frequencies of occurrence for that object by their frequencies of occurrence for the other object if they appear for both objects.

3.6. Backend Feature Decisions

The backend features are compared manually or with respect to results for the triples of the first study given in Table 5.2.1. To obtain such comparison a script was used to feed the triples to the backend (just as a user input). The backend does its usual processing and delivers a score for the given aspect. For each of the triples, a difference to a *gold score* can be calculated. The *gold score* basically is the optimal score the system can give: For *BETTER* as gold label 100% is the corresponding *gold score*, for *WORSE* it is 0% and for *NONE* it is 50%. This difference is hereafter referred to as "gold deviation". The average gold deviation over all evaluation triples is one measurement to compare the quality of backend features.

A low gold deviation means that the aspect dependent score gives a clear advice where the relevant sentences can be found. Furthermore, in addition to the gold deviation there were calculated precision, recall, accuracy, and F1. To classify a processed score as correct or incorrect to calculate the enumerated metrics, the ranges in Figure 3.6.1 were used.

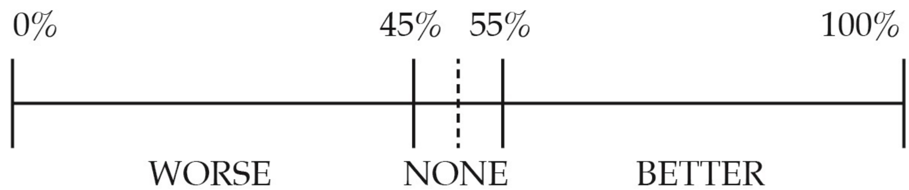


Figure 3.6.1.: Ranges to assign a label to calculated scores for comparing them with an existing gold label. E.gif the score is smaller than 45% the corresponding label is *WORSE*. The range for *NONE* is not as large as for the other labels since a 5% advantage for one object is an obvious indicator for its superiority.

The marker approach as described in Subsection 3.3.1, is chosen as the baseline system. It uses a basic score function presented in Listing 3.3, taking into account the number of occurred markers (*marker_count*), the Elasticsearch score, the aspect weight (*weight*) (normally chosen by the user) and the maximum Elasticsearch score (max_sentence_score).

```
(sentence_score / max_sentence_score) * (weight + marker_count)
```

Listing 3.3: Score function used in the baseline system. The (maximum) Elasticsearch score, the aspect weight and the number of contained markers are used.

With the Elasticsearch index (depcc) described in Section 3.1, the baseline system reached an average gold deviation of 0.39 with a standard deviation of 0.39, when only taking into account the triples with results. In total there were only 18 of 34 triples with a result, 10 were correct and 8 incorrect. Quality measurements for the 18 given results are presented in Table 3.6.1.

	BETTER	WORSE	NONE
precision	0.67	0.57	0
recall	0.25	0.89	0
f1 score	0.36	0.69	0
accuracy	0.61	0.61	0.89
total accuracy			0.56

Table 3.6.1.: Measurements for the baseline system according to the evaluation set of the first study and the scores for the entered aspect. Only the 18 of 34 triples with results are taken into account.

3.6.1. Sentence Scoring and Ranking

The system used for the first study had a scoring function without an influence of the classification confidence assigned by the classifier (described in Subsection 3.3.2) for each label (see Listing 3.4).

```
sentence_score * aspect_weight
```

Listing 3.4: Basic score function using the Elasticsearch score (sentence_score) and the user entered aspect weight.

Instead, the confidence was used to sort the sentences, which brought longer sentences to the top, whereas in Kibana only the shortest and most concise sentences are placed top (The ranking function of Kibana places sentences with a higher number of query words with respect to the sentence length to the top). This observation of the first study showed that the Elasticsearch score should be considered for sorting instead of only using classification confidence. Nevertheless, since the sorting of the sentences does not influence the scores, the scoring function of the first study already performed better than the baseline system with an average gold deviation of 0.38 with a standard deviation of 0.3. In addition, the system found for 33 of 34 triples results where 20 were correct and 13 incorrect according to the ranges given in Figure 3.6.1. Furthermore, almost all values for the BETTER and NONE classification increased (see Table 3.6.2).

	BETTER	WORSE	NONE
precision	0.61	0.73	0.25
recall	0.73	0.5	0.5
f1 score	0.66	0.59	0.33
accuracy	0.67	0.67	0.88
total accuracy			0.61

Table 3.6.2.: Measurements for the scoring function used in the first study, considering the 33 triples with results.

The classifier using the *InferSent* feature set reached a slightly lower (better) gold deviation with a higher standard deviation of 0.32. However, only for 31 triples, results were found, where 17 were correctly set.

As described, it is useful to consider the Elasticsearch score for the sentence ranking. However, taking the classification confidence into account is also important, since it determines the correctness of sentences with respect to the labels. Because of that, other scoring functions were evaluated containing both Elasticsearch score and classification confidence. The result of the development is one scoring function including both and being nearly as good as the one above. It is presented in 3.5.

```
(sentence_score + classification_confidence * max_sentence_score)
* aspect_weight
```

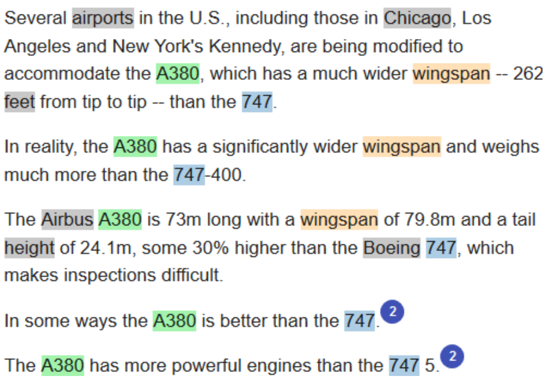
Listing 3.5: More advanced score function taking into account the classification confidence produced by the classifier. Furthermore, again the (maximum) Elasticsearch score and the user entered aspect weight.

The average gold deviation and its standard deviation just stayed the same, but there is one more triple incorrectly determined. However, on most of the manually examined triples, the sorting of sentences improved in comparison to the basic score function presented in Listing 3.4, just as in the example in the figures 3.6.2 and 3.6.3. The ranking corresponding to the basic score function, (Figure 3.6.2) has a long sentence on the top, whereas the ranking corresponding to the new score function (Figure 3.6.3) presents very concise sentences right at the top.

However, using the last described score function still includes the classification confidence for ranking. The final approach described in Section 3.4 only uses the confidence to boost the score of certain sentences (using the maximum Elasticsearch score), the score function used is the basic score function described at the beginning of this section.

3.6.2. Confidence Threshold

For the sentences, the label with this highest classification confidence of the classifier (described in Subsection 3.3.2) is used. As the worst case, two labels can get 0.33 as confidence and the last one 0.34, which then would select the last one as result, even though there are nearly equal confidences for each label. Because of that, another approach to



Several airports in the U.S., including those in Chicago, Los Angeles and New York's Kennedy, are being modified to accommodate the A380, which has a much wider wingspan -- 262 feet from tip to tip -- than the 747.

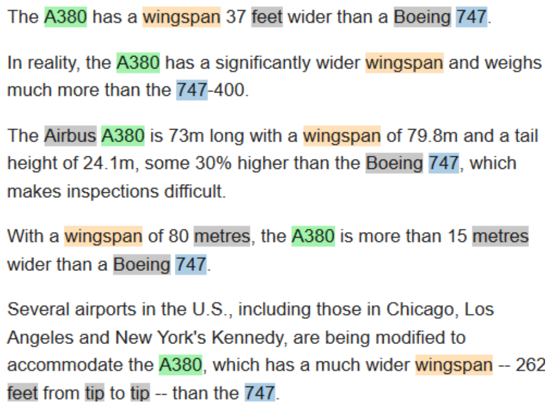
In reality, the A380 has a significantly wider wingspan and weighs much more than the 747-400.

The Airbus A380 is 73m long with a wingspan of 79.8m and a tail height of 24.1m, some 30% higher than the Boeing 747, which makes inspections difficult.

In some ways the A380 is better than the 747. 2

The A380 has more powerful engines than the 747 5. 2

Figure 3.6.2.: The first five sentences of the comparison triple *A380* compared to *747* with respect to *wingspan*, for the object *A380* using the first described scoring function (sentence_score * aspect_weight) and the classification confidence to sort.



The A380 has a wingspan 37 feet wider than a Boeing 747.

In reality, the A380 has a significantly wider wingspan and weighs much more than the 747-400.

The Airbus A380 is 73m long with a wingspan of 79.8m and a tail height of 24.1m, some 30% higher than the Boeing 747, which makes inspections difficult.

With a wingspan of 80 metres, the A380 is more than 15 metres wider than a Boeing 747.

Several airports in the U.S., including those in Chicago, Los Angeles and New York's Kennedy, are being modified to accommodate the A380, which has a much wider wingspan -- 262 feet from tip to tip -- than the 747.

Figure 3.6.3.: The first five sentences of the comparison triple *A380* compared to *747* with respect to *wingspan*, for the object *A380* using the last described scoring function to also sort.

use the valuable information of the classification confidence is to filter out all sentences below a determined confidence threshold. To select a threshold that does not filter out too much, but also is not too low so that uncertain sentences are ranked top, above-described methodology is used again to compare different thresholds based on aspect dependent scores.

The values described are only from using the 34 triples of the first study, which probably are too few to make clear statements, what threshold would fit best. Nevertheless, tendencies can be observed. As expected the gold deviation decreases if the threshold increases, since more uncertain sentences are dismissed. The lowest gold deviation is reached for 0.9 as the threshold, as can be seen in Figure 3.6.4. However, the drawback of that very high threshold of 0.9 is the decrease of triples with a result, as can be seen in Figure 3.6.5. For 0.9, only 20 triples with result are left. However, for those 20 triples, the highest accuracy (70%) is reached. In addition, with that threshold, the system achieves higher values for all measurements (e.g. F1) for *BETTER* and *NONE* and comparable values for *WORSE* (see Table 3.6.3).

	BETTER	WORSE	NONE
precision	0.83	0.5	1
recall	0.77	0.67	1
f1 score	0.8	0.57	1
accuracy	0.75	0.7	0.95
total accuracy			70

Table 3.6.3.: Measurements for the threshold 0.9 the above described basic rank function (sentence_score * aspect_weight).

To overcome the described drawback of loosing too many sentences for triples and showing no results to the user, the **gradually decreasing threshold** described in Subsec-

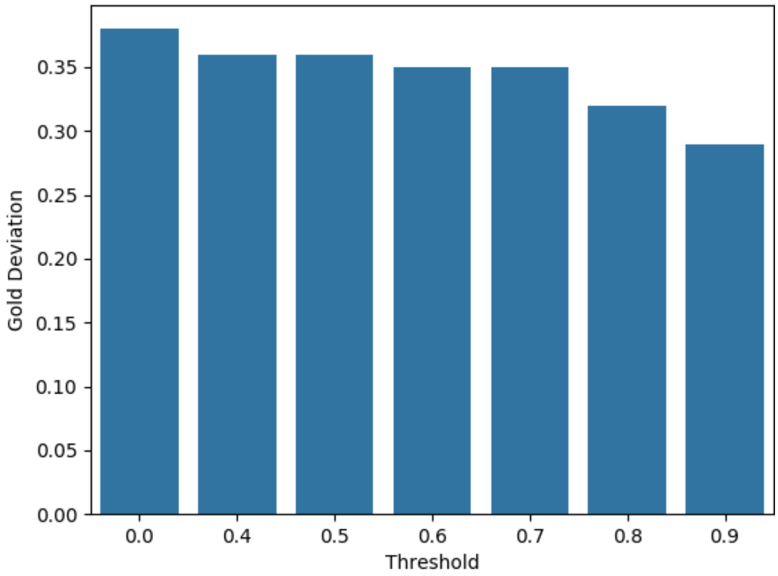


Figure 3.6.4.: The average deviation from the gold score with respect to different confidence thresholds. The bar at 0 as threshold basically is the result of the above described basic rank function (sentence_score * aspect_weight).

tion 3.4.1 was introduced and evaluated.

As expected, the CAM can deliver for almost all triples results using a sentence-threshold (how many sentences should at least be used for the result) above one. For five there is reached the highest accuracy with 21 correct against 12 incorrect classified labels (see Figure 3.6.7). Furthermore, for the same sentence-threshold, the best (lowest) gold deviation is reached (see Figure 3.6.6).

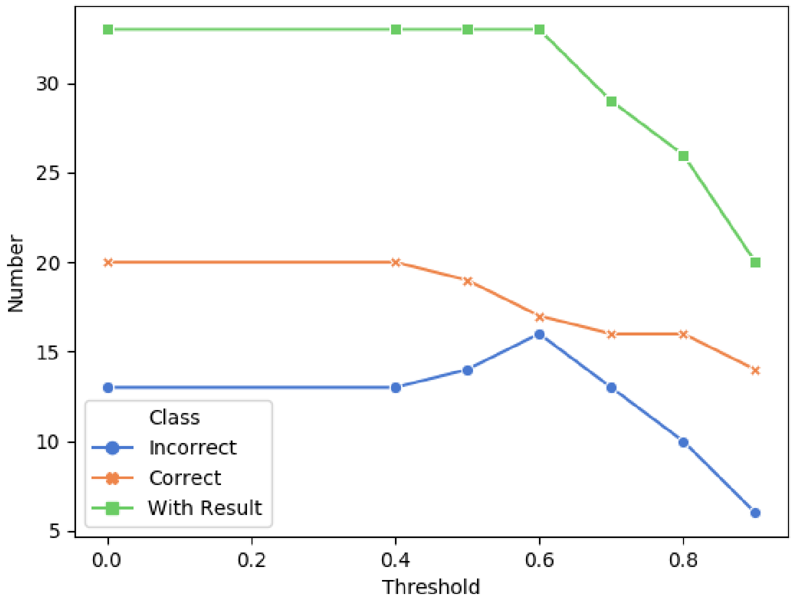


Figure 3.6.5.: The number of correctly and incorrectly classified triples and the total number of classified triples (out of 34), all depending on the threshold.

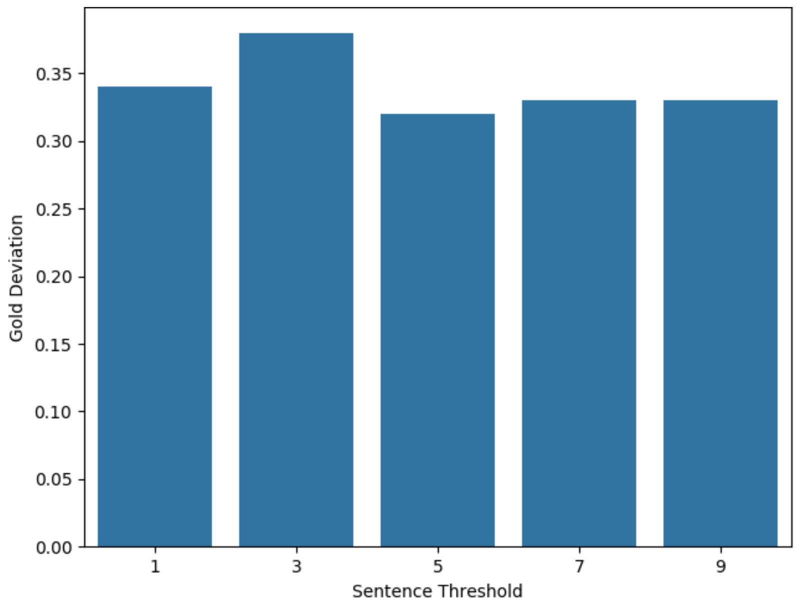


Figure 3.6.6.: The average deviation from the gold score with respect to different sentence-thresholds. The thresholds again were evaluated using the above described score function (sentence_score * aspect_weight).

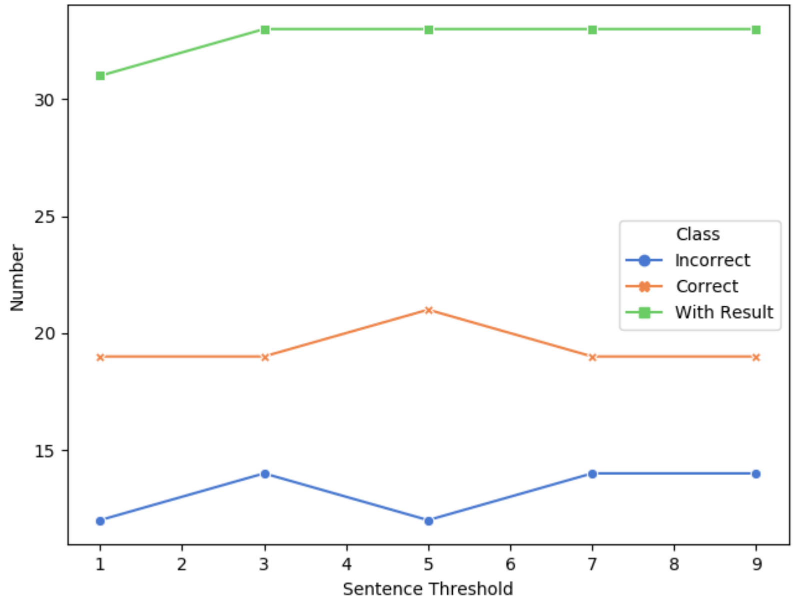


Figure 3.6.7.: The number of correctly and incorrectly classified triples and the total number of classified triples (out of 34), all depending on the sentence-threshold.

3.7. Summary

To show how the backend features, described in this chapter, work together to build up the result presented to the user, the annotated screenshot (see Figure 3.7.1) is used:

(1) Shows the aspect dependent score. All sentences scores containing one aspect are summed up, to build this score. It is described in Section 3.4.

(2) Shows the aspect independent score. All sentence scores containing no aspect (fallback sentences) are summed up, to build this score. It is also described in Section 3.4.

(3) Shows the extracted aspects. The extraction of aspect is described in Section 3.5. It is the last step since the object assignments of sentences have to be clear for this step.

(4) Is sentence 256 of the example sentences of Figure 3.2.1. In Step 3 (Section 3.3) the sentence was assigned to object B (see Table 3.3.1), but it was moved to object A by the negation dissolving heuristic in Step 4 (see Subsection 3.4.2), due to negated sentences with the same meaning in the sentence list of object A.

(5) Shows the result of the in Step 2 (Section 3.2) performed aggregation of exact duplicates.

(6) Is sentence 298, of the example sentences (see Figure 3.2.1), which was assigned to object A directly (see Table 3.3.1). It was ranked high, because of its high Elasticsearch score and even got boosted by the scoring in Section 3.4, due to the classification confidence above 0.8 (0.86), which is the highest step of the gradually decreasing threshold (see Subsection 3.4.1).

(7) Shows sentence 956, of the example sentences of Figure 3.2.1, which was missed by the heuristic, due to the limited capabilities of the chosen pattern matching approach (see Subsection 3.4.2).

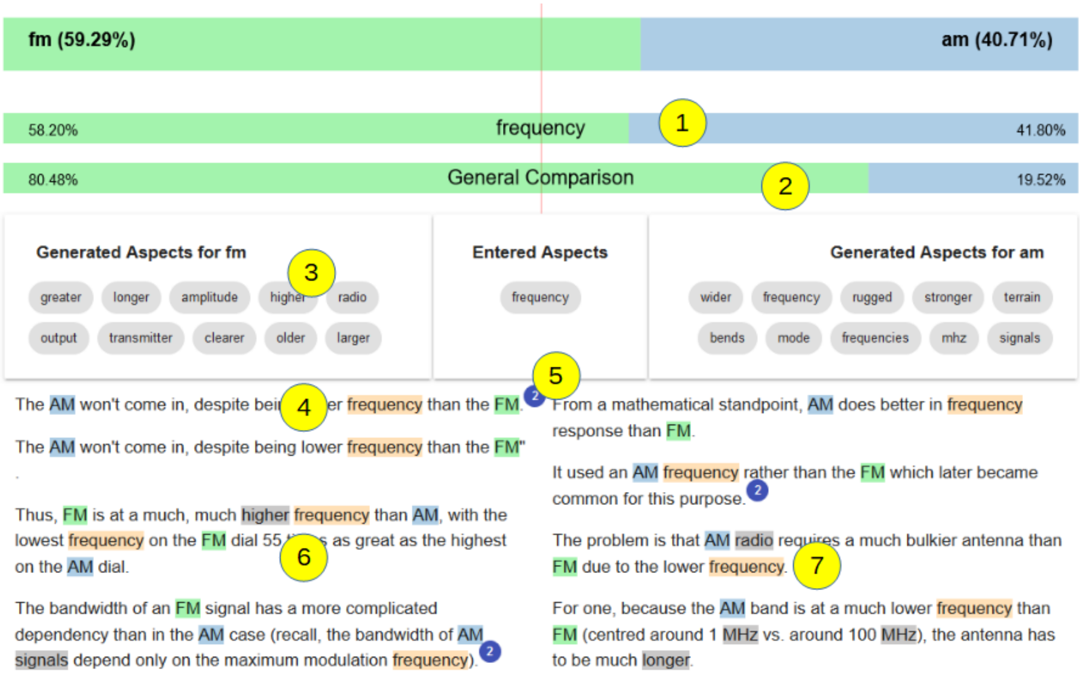


Figure 3.7.1.: Frontend screenshot with annotations to map the shown results to the described backend features.

4. The Frontend of the Comparative Argumentative Machine (CAM)

In this chapter, first the initial user interface and in the end the final user interface is described. In between, the feature decisions taken are presented and explained. The basic structure of the interface is kept over the development. It consists of two main elements: a comparative question input and an answer presentation component. The principal goal of the input component is to provide a user interface to submit a comparative question request in a form of a triple as defined in Chapter 3. The output component returns a decision-making support summarizing data retrieved from a large text collection.

4.1. The Initial User Interface

In the following subsections, the user interface, evaluated in the first user study, is described. It has basic features, which also can be found in some examined interfaces in Section 2.1.

4.1.1. User Input

The user interface presented in Figure 4.1.1 shows a kept simple form, for entering user input. There are two input fields to enter two objects, which should be compared. Furthermore, there is an arbitrary number of input fields for aspects, on which the two objects should be compared. The described setup is following a combination of input pattern *C: Specialize Comparison* and *D: Object Input Fields* presented in Section 2.1. As in *D*, an arbitrary number of input fields is given to the user. Autocompletion has not yet been introduced but would make a valuable feature. In addition, like in *C*, the user has the possibility to further specify the comparison by adding an arbitrary number of aspects. Furthermore, a weight can be set to the entered aspects by using a slider. The weight influences the score of sentences belonging to that aspect, this increase of sentence scores leads to a bigger impact on the overall score. The different sentence classification models, described in Section 3.3, are options of the drop-down menu. Selecting another answer retrieving model changes the processing, but not the answer presentation. As described in Section 3.1, the "Faster Search" option decreases the number of queried sentences from a maximum of 10,000 to 500. For the machine learning approaches, only the number of sentences not containing the aspects is decreased. On the bottom, the current step of the answer processing is given while processing.

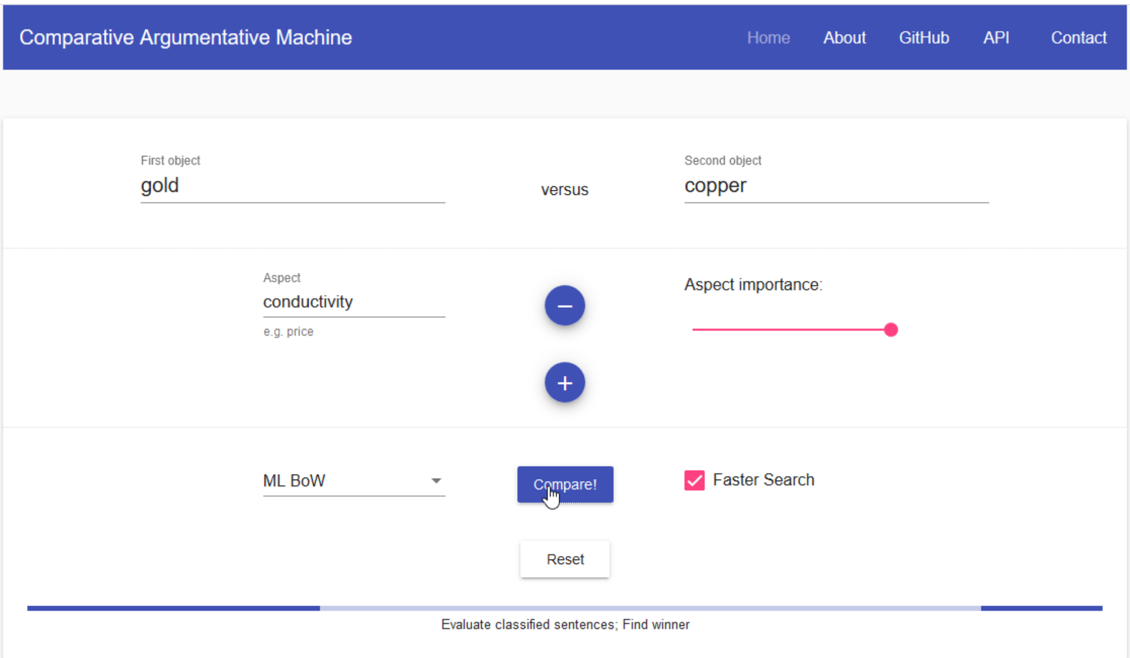


Figure 4.1.1.: The user interface for entering a desired comparison triple. In this example *gold* is compared to *copper* with respect to *conductivity*.

4.1.2. Answer Presentation

The answer presentation of the system, presented in Figure 4.1.2, shows clearly separated areas for both objects. One of the objects is declared as winner, based on the scores of found supporting sentences. On the bottom, these supporting sentences are shown. In the middle, generated aspects are presented, these can indicate on which aspects the corresponding object is better than the other. The compared objects are highlighted in red or green, based on the standing. Furthermore, entered aspects are highlighted in light blue, whereas generated aspects are highlighted in grey. The answer presentation combines features of output pattern *F: Overview to Detailed* and *G: Aspect Dependent Part Comparison* (presented in Section 2.1). As shown in Figure 2.1.10, in *F* pro and con arguments are presented column-wise in this pattern. The CAM answer presentation shows the support sentences, which can be viewed as pro and con arguments corresponding to the assigned objects, in a column-wise manner, too. In addition, the presentation shows a ranking of objects based on the sentence rankings just like in *G* where an individual ranking for the compared objects is given. Furthermore, just as in *G* the given colors for the ranking are used to identify the individual objects in the answer presentation of CAM.

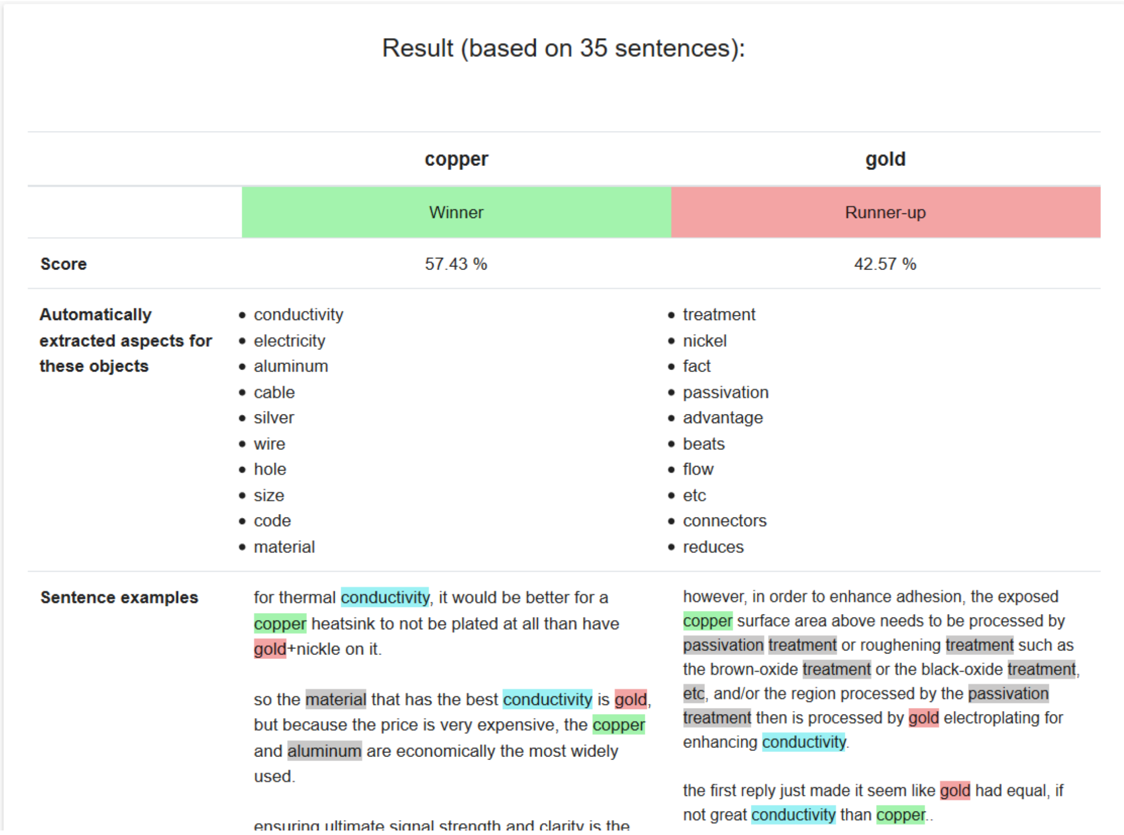


Figure 4.1.2.: The result presentation for the entered input triple: *copper* vs. *gold* with respect to *conductivity*.

4.2. Frontend Features Decisions

In this section, the developed frontend features are compared and selected argumentatively, based on interface design guidelines (e.g. "provide unbiased data", "make the system familiar" or "help people find alternatives"), more detailed below, found in [Shneiderman, 2010] and [Johnson, 2014]. The comparison based on guidelines is chosen since it is not possible to calculate a numeric value to directly compare features.

4.2.1. Score Presentation

The initial score presentation (see the upper part of Figure 4.1.2) was able to present the total score percentages of the compared objects, which gave a first impression what object wins the comparison. Different colors were already used to emphasize the winning object.

The score presentation feature of the final interface (see Figure 4.2.1) takes into account shortcomings detected in the preparation of the first study and the study itself. For example, because only a total score was shown, it could be that object A (e.g. *Gold*) is better in some aspects, but the total score shows object B (e.g. *Copper*) as the winner. The new presentation, therefore, introduced a more partitioned score, where every entered aspect has its own bar to present the score ratio. The coloring of the losing object also changed to a more neutral color, because of the same reason (an object could also win in some aspects, but be inferior for the total score). Furthermore, charts are used to make the distribution of scores more visual and therefore easier to grasp.

Taking into account design guidelines and rules, the new score presentation is more helpful reaching the goals of the users, since it explicitly displays the result of the comparison with respect to the entered aspect(s). To **understand the goals** of the users is an important point when designing user interfaces as described in [Johnson, 2014, p. 12]. A guideline to support human decision-making is to **provide unbiased data** (see [Johnson, 2014, p. 176]), which is better supported by the score presentation used in the final study, since it is more granular and therefore can be better understood. In [Shneiderman, 2010, p. 76-77] five guidelines for organizing the display are described. For the score presentation of the system, the **efficient information assimilation by the user** is most interesting. To do so, the presentation of data in a graphical form and to present digital values only when necessary is suggested. The score presentation used for the final study takes into account the suggestions and therefore simplifies the capturing of the results. Design rules to reduce the amount of attention a user needs to operate the system are given on [Johnson, 2014, p. 146]. To **make the system familiar** (one of the rules) the output pattern D (see Section 2.1) where a variety of bar charts is used was taken up.

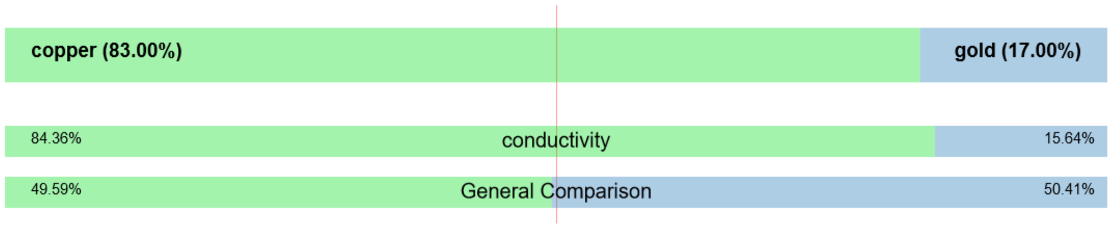


Figure 4.2.1.: The score presentation used by the final system.

4.2.2. Sentence Presentation

Two variants for the presentation of single sentences are given in Figure 4.2.2. On the left, the sentences are underlined when hovering with the cursor (to show the user it is clickable). On the right, a button shows that there is a source, that can be viewed. The left variant was selected over the right because it allows keeping the same sentence presentation even if no source is available (the hover event would disappear as only change). To use the same sentence presentation supports the **strive for consistency** rule, which is one of the eight golden rules described in [Shneiderman, 2010, p. 88-89]. Furthermore, the presentation shown on the left is similar to the presentation of links, which makes it follow a design rule described on [Johnson, 2014, p. 146] (**make the system familiar**). The final interface uses the chosen variant on the left to access the context as presented in Subsection 4.2.3.

To group the sentences by the objects and by contained aspects also two different

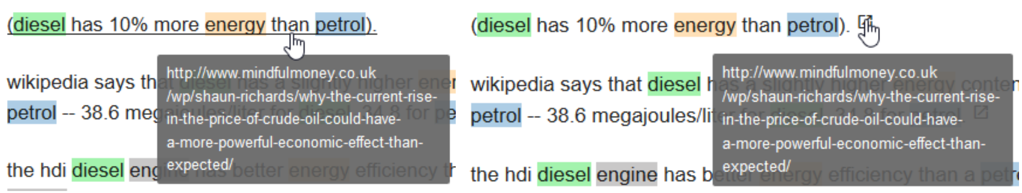


Figure 4.2.2.: The two compared approaches for the inclusion of sentence sources. The marking is kept the same for both.

variants were developed and compared based on used guidelines and general advantages/disadvantages.

The approach of Figure 4.2.3 focused on the grouping by aspect. The advantage of this approach is that the user is able to see how much evidence (number of sentences) is given per aspect. In addition, without expanding one group the user can get an overview of the distribution of sentences. Furthermore, it is possible to click the aspects to add them to the entered aspects to use them on another comparison run.

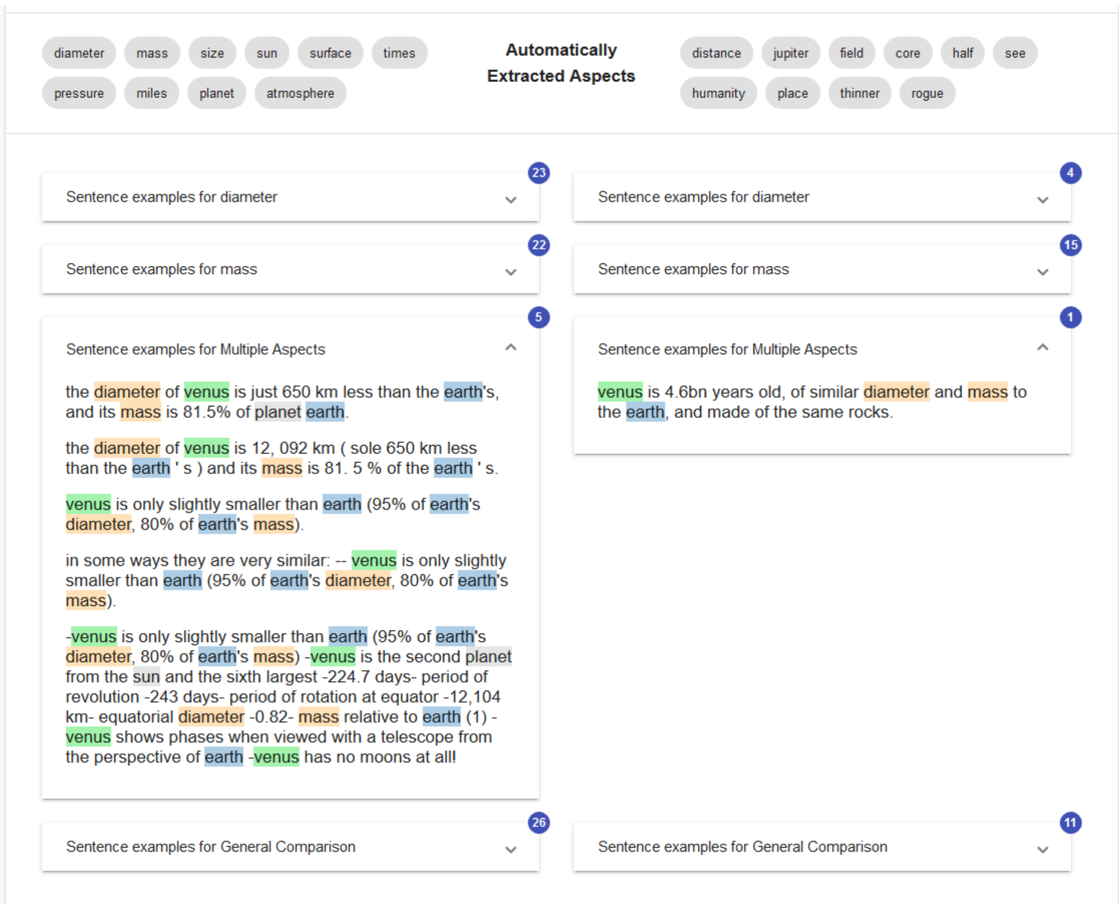


Figure 4.2.3.: The entered aspects determine groups of sentences. Sentences with more than one aspect are placed in an extra group called "Multiple Aspects". All groups can be expanded individually to allow the user to read them without any distraction of other sentences.

However, the approach presented in Figure 4.2.4 is used for the final CAM version. For the user, it is possible to reach the same grouping of sentences as in Figure 4.2.3 by selecting the entered aspects as filter options. Furthermore, the user immediately can see result sentences without the need for orientation and reading the titles of groups. The gestalt principle **proximity** described in [Johnson, 2014, p. 13ff.] is used to show the affiliation of sentences to objects instead of expandable elements. Furthermore, the filter approach enables the user to find alternatives immediately, which supports a guideline contained in the decision-making system guidelines described in [Johnson, 2014, p. 176] (**help people find alternatives**).

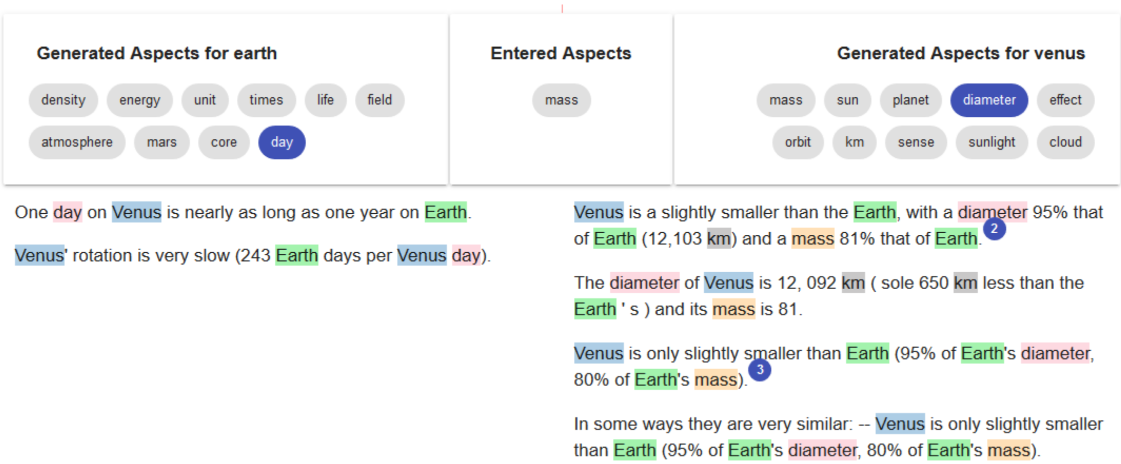


Figure 4.2.4.: In this approach, the user is able to filter sentences by clicking aspects, if more aspects are clicked the sentences containing all selected are presented. Both columns are separately filterable, only the *Entered Aspects* filter both columns.

The selection of emphasizing colors for the objects and aspects took into account the guidelines for using color presented in [Johnson, 2014, p. 45-46]. However, some kind of anti-pattern called **text on noisy background** (described in [Johnson, 2014, 77-78]) describes the bad influence on readers' performance when text is placed above a noisy background. The need to use distinguishable colors suggested by the guidelines, but not disturbing the reader lead to a trade-off selecting of lighter versions of the most distinctive colors (red, green, yellow and blue).

4.2.3. Source- and Context-Presentation

In Figure 4.2.5 the selected way context is added to sentences is shown. Another approach was to place a button beneath every sentence, which would open the context presentation, as can be seen in Figure 4.2.2 on the right. The feature selection for this part is described above.

For both developed context presentation features the gestalt principle **Figure/Ground** described in [Johnson, 2014, p. 21ff.] is used to show the context to help the user keep oriented. Figure 4.2.6 shows one them. It has the advantage, that at the beginning all sources, the sentence occurs in, are listed as an overview. The context presentation itself is shown after a source is selected. It looks just like for a sentence with only one document as context.

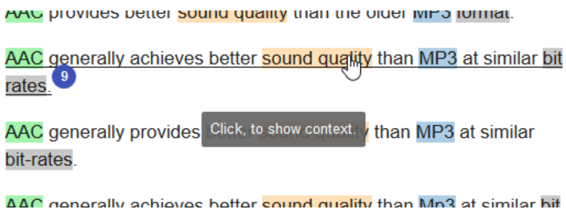


Figure 4.2.5.: The sentence presentation selected for the usage in the final version of CAM. Exact duplicates are aggregated and summarized by the shown number. The coloring of the objects refers to the coloring presented in Figure 4.2.1.

Nevertheless, the approach in Figure 4.2.7 is used for the final system, because it allows the user to select different sources more easily. Furthermore, the same window can be presented to the user regardless, if the sentence is taken from one or multiple documents, which increases the consistency, which is desirable according to the eight golden rules ([Shneiderman, 2010, p. 88-89]).

As suggested by [Cutrell and Guan, 2007] and [Lin et al., 2003] a rather long default snippet length is selected: Three sentences before and after the clicked sentence. Furthermore, the whole document can be displayed to allow the user to stay on CAM as far as the original source is not needed. If the original source is clicked, it is loaded in another tab to keep CAM open.

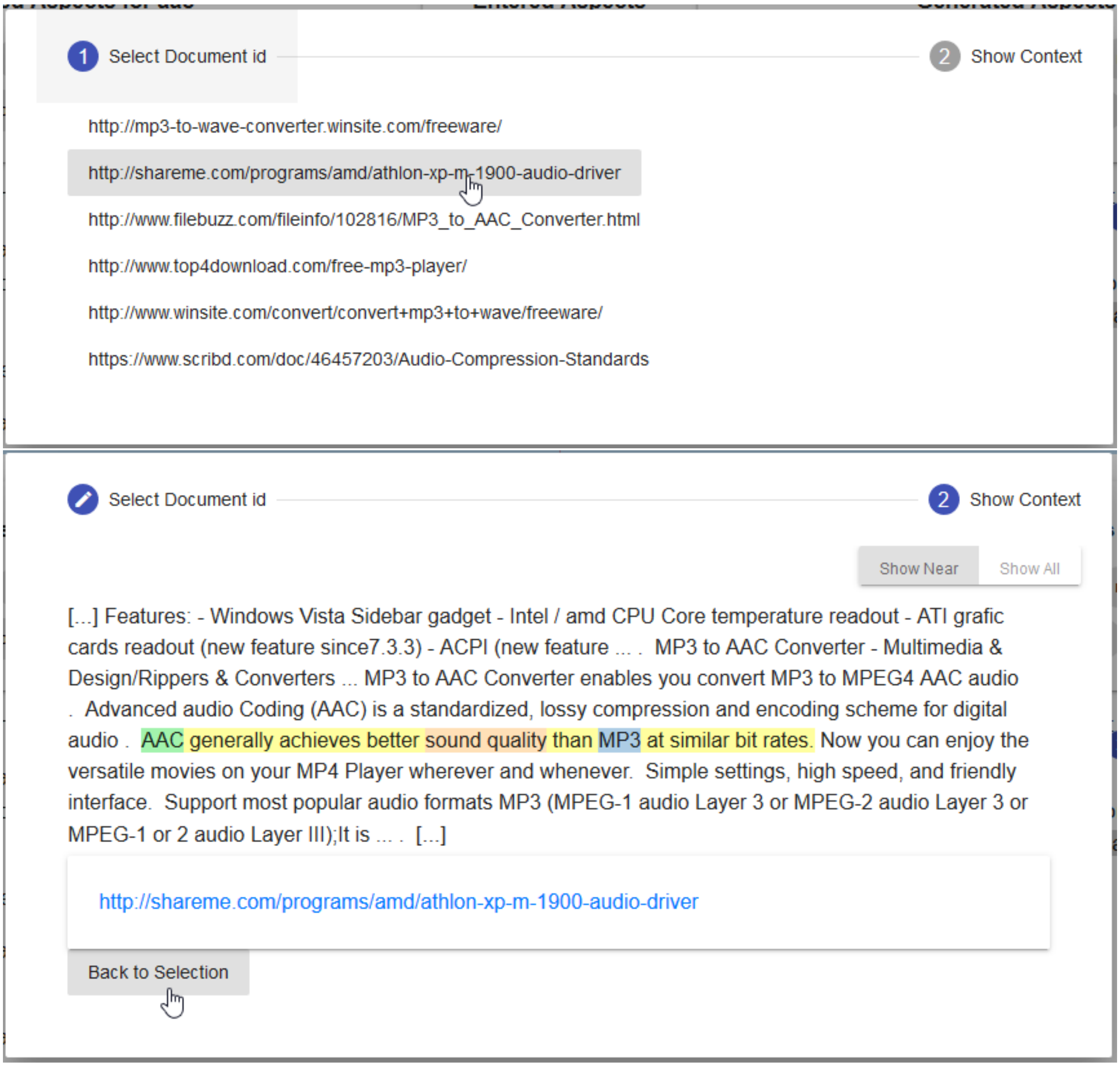


Figure 4.2.6.: On the top, document sources of one sentence occurring multiple times are displayed. On the bottom (shown when a source is clicked) the context corresponding to the document is presented. It is possible to show the whole document by clicking "Show All" and to open the document by clicking the link.

Figure 4.2.7.: The near context of the selected document is presented. It is possible to show the whole document by clicking "Show All". Furthermore, other document sources can be selected within the drop-down options. The source can be opened by clicking the button right to the drop-down.

4.3. The Final User Interface

This section describes the final user interface as used as in the second user study described in Chapter 6. It basically summarizes the design decisions of the previous section in the following two subsections.

4.3.1. User Input

The user interface to enter comparisons, presented in Figure 4.3.1, is divided into three parts. On the top, the user enters comparison target objects. In the middle, the interface allows to add an arbitrary number of aspects and weight them from one to five. The set weight boosts the scores of the sentences containing the assigned aspect and therefore the position of the sentences in the presentation. Furthermore, the weight increases the share of the total score for that aspect. On the bottom, the three different models to classify accordingly retrieved sentences, as described in Chapter 3, can be selected (*Default*, *Bag-of-Words*, and *Infersent*). The *Faster Search* option limits the number of queried fall-back sentences to 500, to speed up the answer processing.

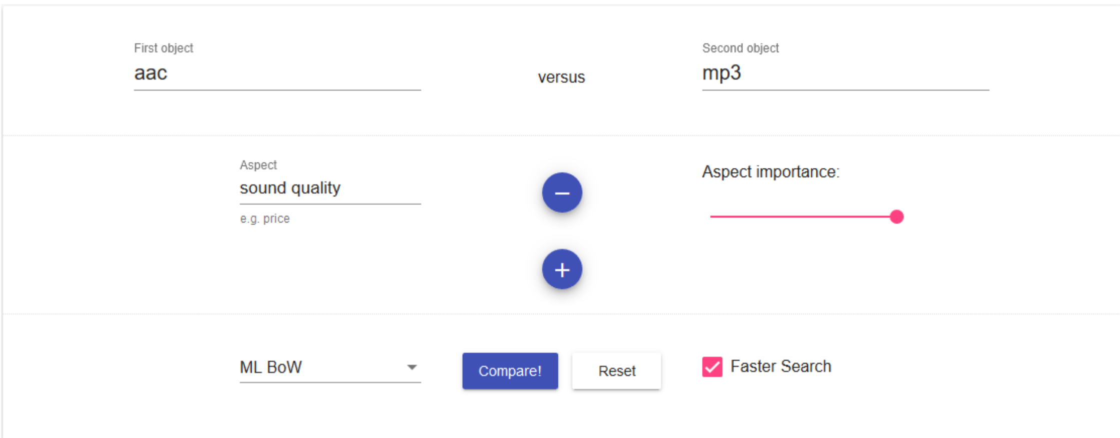


Figure 4.3.1.: The input mask of CAM used in the final study.

4.3.2. Answer Presentation

The presentation of comparative answers is shown in Figure 4.3.2. On the top, different scores are given. The overall score distribution bar allows the user to grasp a general answer for the entered comparison, on that score all sentences (including the fall-back sentences) are considered. Underneath, aspect-specific scores are shown. At the bottom, the *General Comparison*, which just includes scores of sentences not containing any entered aspects (fall-back sentences), is presented. Basically, the overall score summarizes all other presented. Further, generated and entered aspects are presented in a clickable manner to allow the user to filter displayed sentences. The filter words are combined as a disjunction. User entered aspects filter all sentences, whereas generated aspects only filter the corresponding column. The generated aspects were extracted from all assigned sentences in Step 5 of the scheme presented in Chapter 3. The objects in displayed sentences are highlighted with the same colors used for score presentation to support the assignment, aspects are also colorized to ease distinction. By clicking a sentence, its context can be viewed — first, a sentence window of three sentences before and after of the clicked one, with the possibility of expanding to the original document.

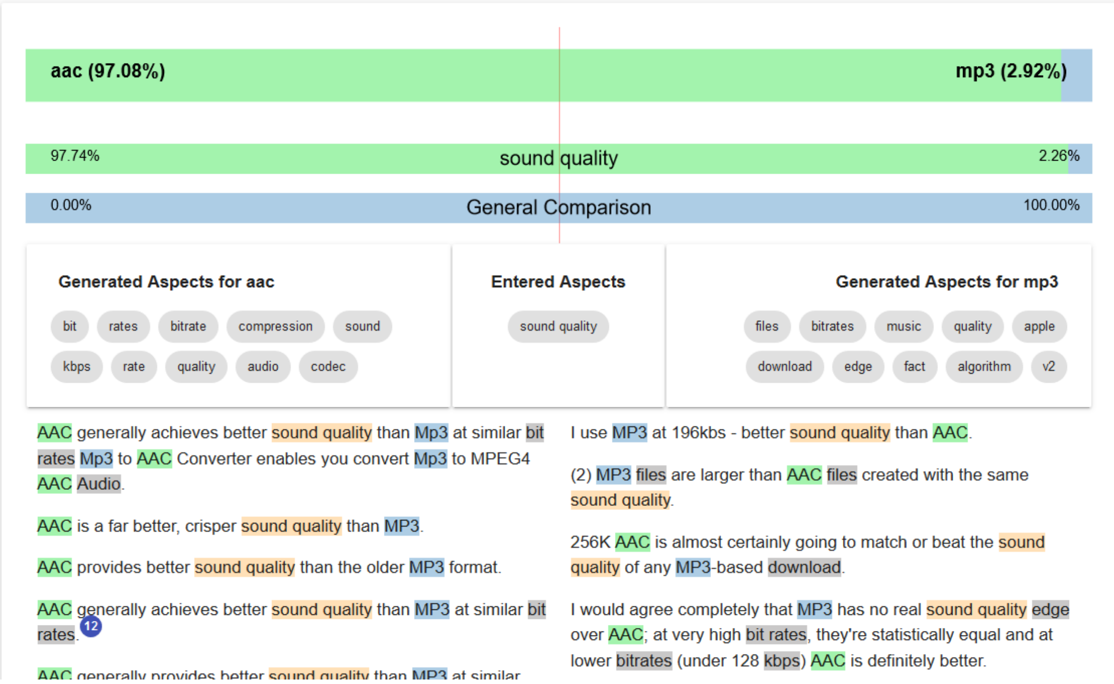


Figure 4.3.2.: The CAM result presentation used in the final study.

5. The First User Study

The first study served as a starting point to develop a system, that can deliver an easy to capture answer on a comparative question with respect to a given aspect. The prototypes quality was measured against a baseline system. Kibana[1] served as this baseline system. Kibana is a tool to process a keyword search in an Elasticsearch instance. For the first study only queries of the form "A AND B AND C_k" were relevant. Such queries only find sentences containing the objects (A and B) and the aspect (C_k). If the object or the aspect had more than one word, the participants had to put quotation marks around that sequence.

Both systems worked on the same smaller corpus, which is described in Section 3.1.

5.1. Backend Setup

The backend of this first version used the smaller of the corpora (described in Section 3.1) that does not contain any duplicates. Therefore, no sentence preprocessing as described in Section 3.2 was needed (except for the extraction of sentences from JSON). The approaches for sentence classification described in Section 3.3 were already used for this study system, but without the negation dissolving heuristic, which was a consequence on the study results. Another consequence was the inclusion of the classification confidence for scoring and not only for sorting the assigned sentences like in this study. The first of the methodologies, described in Section 3.5, was used in the first study.

5.2. Evaluation Dataset

To see how CAM performs in comparison to Kibana as baseline system, the evaluation triples in Table 5.2.1 were used as input. The dataset meets some rules to make the study more expressive.

The triples are clear-cut so that the participants can understand the scenario, which is meant by the triple. For example "*Eclipse* compared to *Netbeans* with respect to *plugins*" is ambiguous, since there can be found sentences comparing the number, but also the quality of plugins. Good triples (taken from Table 5.2.1) for example are "*Earth* compared to *Venus* with respect to *mass*" or "*a380* compared to *747* with respect to *wingspan*", since numbers specifying the clear winner exists. In addition, it was tried to select triples that

[1]Kibana `https://www.elastic.co/de/products/kibana` (accessed: 11.06.2018)

are not too easy or part of general knowledge, since the time to determine the answer can be expected to decrease if the result is known. Furthermore, the triples were taken not too general, e.g. "Adidas compared to Puma with respect to price" is true for some products, but not for all, which can lead to confusion while capturing the answer.

If the selected triples are too trivial (only delivering a few hits in Kibana), there would be nothing to measure. Because of that, only triples with more than twenty hits in Kibana were selected. The triples of Table 5.2.1 are sorted by hits in Kibana, the first entry has the most hits.

The labels are based on the assumption that more or higher is better. This assumption was set to be able to simplify the aspects to generate more hits in Kibana. To obtain the labels Google search[2], but also WolframAlpha was used to find aspects for comparison.

[2]Google search engine `https://www.Google.com/` (accessed: 07.06.2018)

index	Object A	Object B	Aspect	Label
1	c	python	performance	BETTER
2	petrol	diesel	energy	WORSE
3	gold	copper	conductivity	WORSE
4	aac	mp3	sound quality	BETTER
5	earth	venus	mass	BETTER
6	steel	aluminum	harder	BETTER
7	swimming	running	calories	WORSE
8	milk	soda	calories	BETTER
9	usa	australia	size	BETTER
10	gold	platinum	density	WORSE
11	java	python	crossplatform	NONE
12	steel	aluminum	elastic	WORSE
13	python	go	performance	WORSE
14	lion	tiger	weight	WORSE
15	ben hur	titanic	oscars	NONE
16	MLB	NBA	salaries	WORSE
17	coal	oil	energy content	WORSE
18	pennsylvania	michigan	weather	BETTER
19	concrete	metal	durability	BETTER
20	metal	concrete	ramps	WORSE
21	maple	ash	harder	BETTER
22	iron	copper	melting point	BETTER
23	rowing	running	calories	WORSE
24	galaxy s4	iphone 5	performance	BETTER
25	germany	italy	life expectancy	WORSE
26	windows 7	windows 8	boot time	BETTER
27	porcelain	ceramic	more durable	BETTER
28	plywood	lumber	strength	WORSE
29	walnut	oak	harder	WORSE
30	poland	portugal	average income	WORSE
31	a380	747	wingspan	BETTER
32	model s	i3	range	BETTER
33	hamburg	venice	bridges	BETTER
34	wii u	ps3	processor	WORSE

Table 5.2.1.: Evaluation triples used for the first study. The triples can be formulated as a sentence using the following pattern: ***Object_A*** *compared to* ***Object_B*** *with respect to* ***Aspect***.

5.3. Study Setup

The goal of the study was to measure the quality of the CAM prototype in comparison to an off-the-shelf search engine as the baseline. To be able to use the same dataset as CAM for the comparison system, Kibana was used as a substitute. In the study, the participants were constrained to alternately use CAM and Kibana to make the learning process as equal as possible for both systems. The task was to input the different triples of Table 5.2.1 and determine the label to it. The label could be one of those described at the beginning of Chapter 3 (*BETTER*, *WORSE* or $\neq$).

Two different metrics were taken to compare the quality of the systems. First, the speed of using the system was taken. More accurate, the time determining an answer and also the preparation time (the time from starting to type until the answer is loaded and shown) were taken manually. Second, the correctness of given classifications was determined (correct if exactly the label of the evaluation data was determined, incorrect otherwise).

This study was taken out one-on-one, where the participant had to read and input the triple and capture the answer to determine the label. The participants were said to look for enough evidence (about three sentences), to be certain about the winner. The instructor manually measured the time consumption and wrote down the determined labels.

5.4. Participants

Four participants were chosen to attend the study. This number allows a good overview of the quality, but is not too generalizable, which is okay as starting point. All participants had a bachelor degree and were between 18 and 24 years old. Three of the participants were male and one was female. They received a monetary compensation for their spent time.

5.5. Discussion of the Results

The main results of this study are presented in Figure 5.5.1. The results per triple are sorted by the ratio between the used systems. The figure shows which system performed better on which triple (see Figure 5.5.1 description). The numbers of the horizontal axis correspond to the triple numbers of the evaluation data shown in Table 5.2.1.

Six of the total 34 triples were taken out, due to the fact that every participant, no matter what system was used, answered them wrong. Nevertheless, those six triples were analyzed: the participants needed on average 14% less time to capture the answer with CAM. This result can mean that it is easier in CAM to realize that the presented answer is bad, than in Kibana.

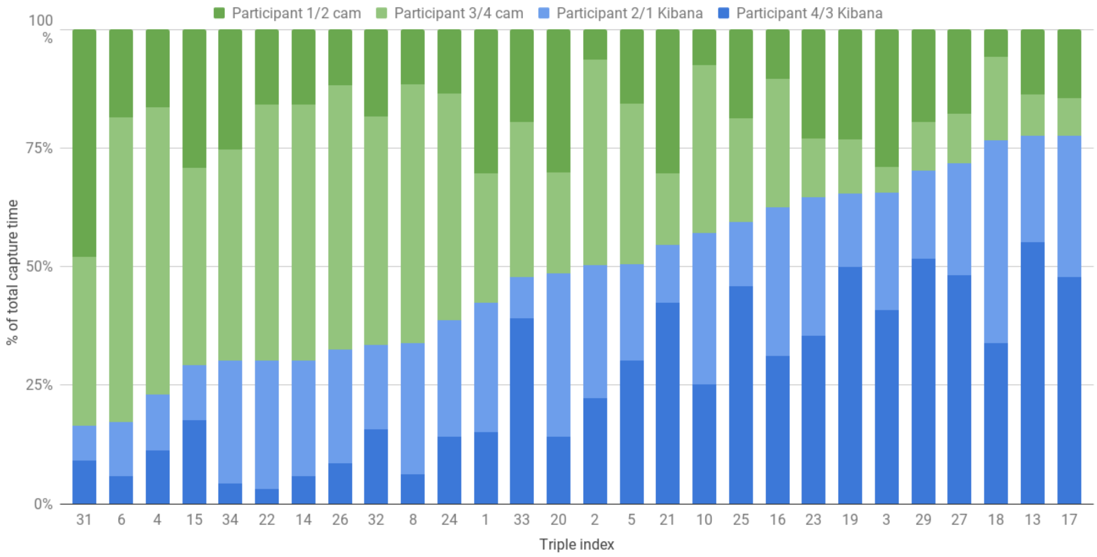


Figure 5.5.1.: The ratio of total determination times of CAM and Kibana. The green parts of the bars are the proportion of total times from CAM and the blue parts of the bars are the proportion of total times from Kibana. If one system is above the 0.5 line, it means, that the combined time both participants needed to determine the answer is longer than the time the others needed with the other system.

The remaining 28 triples, with at least one participant delivering the correct comparison result, were used to make a statement about the quality of CAM compared to the baseline system Kibana. Comparing the combined determining times of both participants using one system, 14 triples were captured faster on CAM, whereas 14 were captured faster on Kibana. In Figure 5.5.1, it can be seen that the triples (31, 6, . . . , 20) are won by Kibana and the triples (2, 5, . . . , 17) are won by CAM. Summing up the times consumed determining the answer with the systems, Kibana shows an 8% advantage over CAM. This can have several reasons, for example, one participant said, that it is more enjoyable to use CAM due to the highlighting of objects and aspects. That participant even needed 27% less time determining the answers with Kibana. However, the same participant only had an accuracy of about 43% in Kibana, but about 93% in CAM, which is noticeable. In total, in Kibana 50% more incorrect labels than in CAM were determined. CAM had an accuracy of about 82% in total, whereas Kibana had an accuracy of about 73% in total, as can be seen in Figure 5.5.3.

To summarize the presented results, the participants needed more time determining the presented answer with CAM, but in terms of accuracy, the use of CAM has a big advantage over the baseline system. Since, for the most comparing tasks, it is more preferable to get a correct answer for a bit more time needed, CAM can be preferred over Kibana. In addition, the median for the determination times is about 25% lower for CAM, as can be seen in Figure 5.5.2.

In addition to the measured numbers, there were some edge cases, where the participants needed drastically more time on one system compared to the other. For the

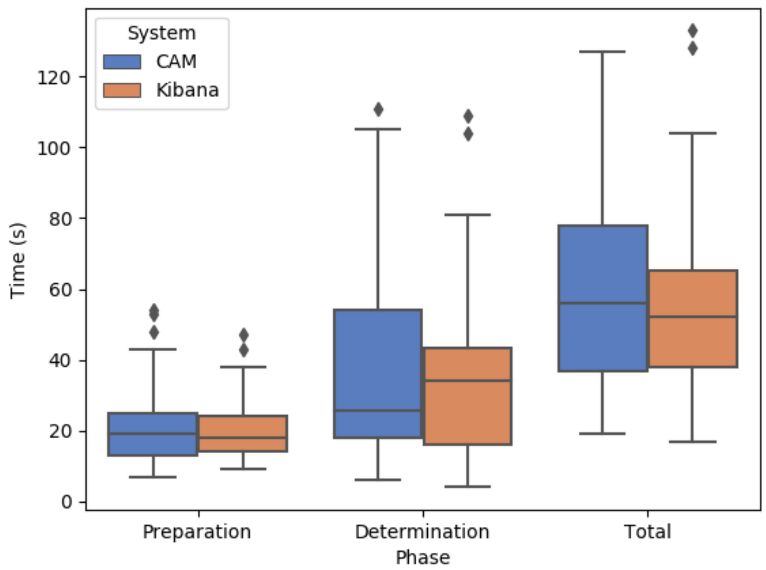


Figure 5.5.2.: Summarized time measurements for both systems.

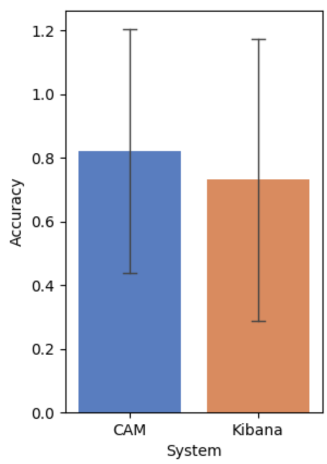
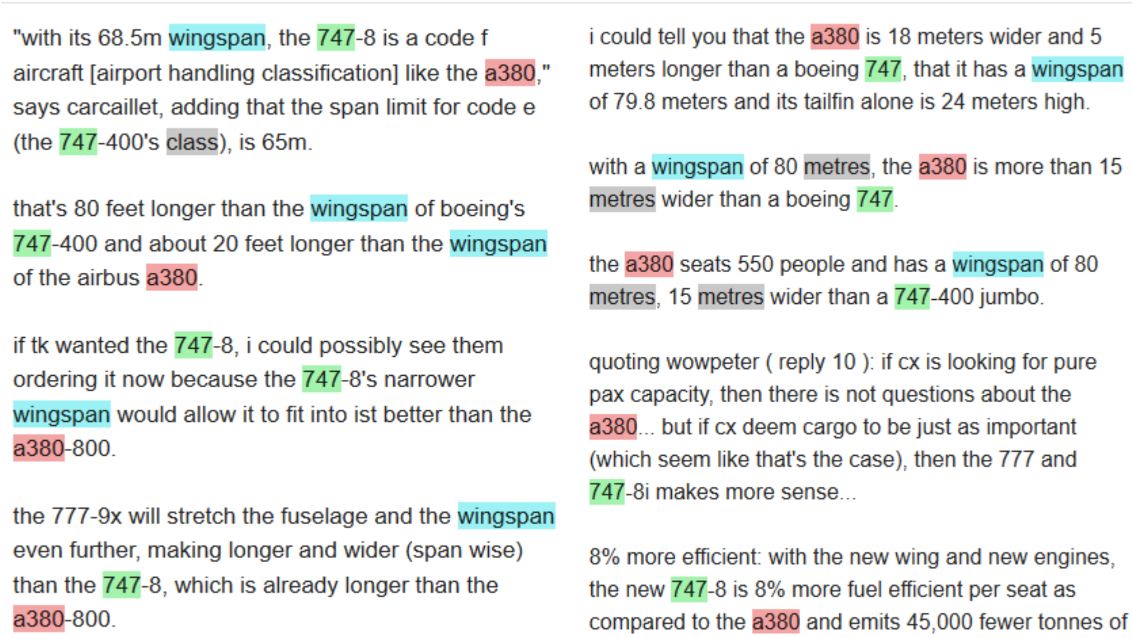


Figure 5.5.3.: Accuracies for the used systems. The error bars present the standard deviation.

triple with index 31 *(a380, 747, wingspan)* the participants, which used Kibana, on average needed 10 seconds to determine the correct answer, whereas on CAM on average about 51 seconds were required to obtain the correct answer. Looking at the presented answers in Figure 5.5.4 and Figure 5.5.5, it gets clear why this behavior appears. In Kibana, the first three sentences all precisely corresponded to the queried comparison. In CAM the first sentences contained the aspect, but they did not directly compare the objects with respect to it. Furthermore, the participants most probably first read the sentences on the left side, since it was the proclaimed winner of the comparison. Since only the second sentence on the left and the second and third sentence on the right contained the right answer, the participant had to read seven instead of three sentences to reach the same amount of certainty for the given answer. For the triple with index 6 *(steel, aluminum, harder)* the ratio (72 seconds on average for CAM against 15 seconds on average for Kibana) and also the problem was similar. The top-ranked sentences are not assigned to the needed answer as clear as in Kibana. The same holds for the triple with index 4 *(aac, mp3, sound quality)*. A possible solution to this problem, which belongs to the ranking of sentences and finding the winner, is to make a combination score function out of the confidence of the classifier and the ElasticSearch relevance score.

Looking at the other side, where the participants needed drastically more time using Kibana, another pattern appears. For the triple with number 17 *(coal, oil, energy content)*, the CAM users needed on average 26 seconds whereas the Kibana users needed on average 89 seconds. On this example, CAM presented only a few sentences that lead to incorrect answers and on Kibana the participants had to read way more sentences, but one determined the correct label. So the CAM users spend 71% less time reading but got the doubled amount of errors. Since the used corpus is the same for both systems, the relevant sentences can also show up in CAM, if the ranking function is adapted accordingly


"with its 68.5m wingspan, the 747-8 is a code f aircraft [airport handling classification] like the a380," says carcaillet, adding that the span limit for code e (the 747-400's class), is 65m.

that's 80 feet longer than the wingspan of boeing's 747-400 and about 20 feet longer than the wingspan of the airbus a380.

if tk wanted the 747-8, i could possibly see them ordering it now because the 747-8's narrower wingspan would allow it to fit into ist better than the a380-800.

the 777-9x will stretch the fuselage and the wingspan even further, making longer and wider (span wise) than the 747-8, which is already longer than the a380-800.

i could tell you that the a380 is 18 meters wider and 5 meters longer than a boeing 747, that it has a wingspan of 79.8 meters and its tailfin alone is 24 meters high.

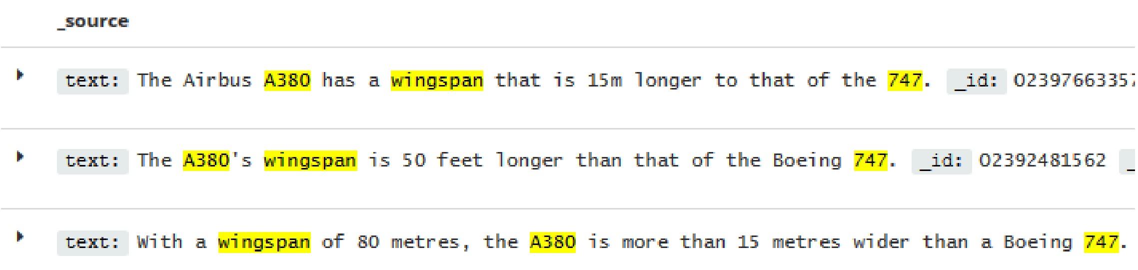
with a wingspan of 80 metres, the a380 is more than 15 metres wider than a boeing 747.

the a380 seats 550 people and has a wingspan of 80 metres, 15 metres wider than a 747-400 jumbo.

quoting wowpeter (reply 10): if cx is looking for pure pax capacity, then there is not questions about the a380... but if cx deem cargo to be just as important (which seem like that's the case), then the 777 and 747-8i makes more sense...

8% more efficient: with the new wing and new engines, the new 747-8 is 8% more fuel efficient per seat as compared to the a380 and emits 45,000 fewer tonnes of

Figure 5.5.4.: Excerpt answer to the 31st triple (*a380* vs. *747* with respect to *wingspan*) presented by CAM. On the left, the sentences belong to the object 747, on the right they belong to the object a380.


_source

text: The Airbus A380 has a wingspan that is 15m longer to that of the 747. _id: 02397663357

text: The A380's wingspan is 50 feet longer than that of the Boeing 747. _id: 02392481562 _

text: With a wingspan of 80 metres, the A380 is more than 15 metres wider than a Boeing 747.

Figure 5.5.5.: The top three sentences presented an answer to the query ("*a380 AND 747 AND wingspan*") corresponding to the 31st triple (*a380* vs. *747* with respect to *wingspan*) by Kibana.

as described above. The triple with index 13 *(python, go, performance)* was an example where CAM outperformed Kibana. Again the users spend 71% less time capturing the answer in CAM, but on this triple, only in Kibana, a participant determined a wrong label. The ranking worked better for CAM on this triple so that the user only had to read the top four sentences to get three sentences indicating the right answer. In Kibana many sentences were shown, which are not related to the topic at all, for example: "If you like Monty Python and can't get to the live performance, go buy the CD.". Similar holds for the triple with index 18 *(pennsylvania, michigan, weather)*.

The preparation time was also measured, as described in Subsection 5.3. In total, it was 8% shorter in Kibana than with CAM. CAM needs more time preprocessing the result sentences than Kibana. That preprocessing time for using the classifier can be reduced if the classified label already is part of the ElasticSearch index[3].

[3] ElasticSearch Index API `https://www.elastic.co/guide/en/elasticsearch/reference/`

As discussed above, a performance increase can be reached with an adoption of the sentence ranking function. Another thing, which appeared relevant was the fact, that in Kibana the result is directly visible when loaded, whereas in CAM the user has to scroll down to see the answer. Because of that, the capture time can further be decreased by automatically scrolling down, when the answer is ready. Furthermore, to neutralize the issue of showing the winner of the comparison with respect to the aspect on the looser side, due to the total score, it is necessary to split up the score to aspect dependent fine granular scores. The finer-grained scores allow the user to exactly obtain the influence of the single entered aspects and to decide better on which side to read. To get an overview of what information are contained in the corresponding sentences for one aspect it is useful to present a summarization feature. For example, the approach of Coocviewer from [Rauscher et al., 2013] could be used to get a fast insight.

As mentioned above to decrease the preparation time of CAM, it is suitable to already use the classifier on ElasticSearch index creation. In addition, to speed up the process of entering objects to compare, an autocomplete feature can be integrated to show suggested options, while the user starts typing (typeahead feature). For example, if the user already typed in the first object, the field for the second object can suggest matching objects for comparison (“Earth”, for example, could lead to “Jupiter”, “Mars” and so on, as a suggestion).

`current/docs-index_.html` (accessed: 09.06.2018)

6. The Second User Study

In order to measure the quality of the final developed system (described in Chapter 3 and Section 4.3), another user study was conducted. The goal of this study again was an evaluation of CAM in comparison to a basic keyword search.

6.1. Evaluation Dataset

The evaluation dataset consisted of 34 comparative questions (triples), with an index and a gold label (see Table 6.1.1) to calculate the system accuracies. The dataset of the first study was not used again, because it was used to tune different backend parameters and features. Most of the new triples were found by using the Google query "*"better than" site:quora.com*". Furthermore, given comparisons from pages like *Diffen*[1] and *Difference Between*[2] were used. To address a shortcoming of the first study, where it was assumed that more or higher is always better and to face ambiguities, comments were added to each triple (see Table 6.1.2). These comments also help clarify subjective divergences, for example, a person on diet would probably say food with fewer calories is better, whereas a person who wants to gain muscle will probably prefer food with more calories. Just as in the first study, it was manually double-checked that the underlying corpus of CAM and the keyword-based search (the 14.3 billion Common Crawl sentences) allows to answer the comparison and only included triples with at least twenty hits in the keyword-based search. At least twenty hits on the keyword search were taken, to make an effect from aggregation by CAM measurable.

[1] `https://www.diffen.com(accessed:31.08.2018)`

[2] `http://www.differencebetween.net/category/technology/software-technology(accessed: 31.08.2018)`

index	Object A	Object B	Aspect	Label
1	mp3	wma	compression	WORSE
2	cable	dsl	speed	BETTER
3	vhs	betamax	picture quality	WORSE
4	ruby	php	performance	WORSE
5	nickel	copper	melting point	BETTER
6	earth	uranus	mass	WORSE
7	rfid	nfc	range	BETTER
8	wav	mp3	sound quality	BETTER
9	fat32	ntfs	security	WORSE
10	ccd	cmos	power	WORSE
11	ntsc	pal	resolution	WORSE
12	lead	silver	density	BETTER
13	copper	bronze	harder	WORSE
14	granite	marble	durable	BETTER
15	hdmi	dvi	quality	NONE
16	pakistan	india	poverty	BETTER
17	yale	harvard	endowment	WORSE
18	mexico	argentina	area	WORSE
19	japan	china	air pollution	BETTER
20	soda	orange juice	calories	BETTER
21	steel	titanium	melting point	WORSE
22	raven	crow	size	BETTER
23	London	Paris	Population	BETTER
24	running	cycling	calories	BETTER
25	induction	gas	boil	BETTER
26	android	ios	app quality	WORSE
27	erlang	java	performance	WORSE
28	turkey	chicken	protein	BETTER
29	plywood	osb	cost	WORSE
30	fm	am	frequency	BETTER
31	glucose	fructose	sweetness	WORSE
32	lightroom	photoshop	price	BETTER
33	concrete	asphalt	cost	WORSE
34	nylon	polyester	elastic	BETTER

Table 6.1.1.: Evaluation triples used for the main-study. The triples can be formulated as a sentence using the following pattern: ***Object A*** *compared to* ***Object B*** *with respect to* ***Aspect***.

index	comment (In this context, we assume that the...)
1	...format with the bigger compression is better.
2	...connection which offers more speed is better.
3	...data medium with better picture quality is better.
4	...language providing better performance on execution is better.
5	...metal with a higher melting point is better.
6	...planet with more mass is better.
7	...technology with higher range is better.
8	...format with a better sound quality is better.
9	...file system which offers more security is better.
10	...image sensor with less power consumption is better.
11	...color encoding system with a higher resolution is better.
12	...more dense metal is better.
13	...harder material is better.
14	...more durable material is better.
15	...connection which delivers better image quality is better.
16	...country with a lower poverty rate is better.
17	...university with more endowment is better.
18	...larger country is better.
19	...country with less air pollution is better.
20	...beverage with fewer calories is better.
21	...metal with a higher melting point is better.
22	...bird with larger size is better.
23	...city with higher population is better.
24	...sport burning more calories is better.
25	...stove with a faster boiling time is better.
26	...operating system is better with an average higher app quality.
27	...language providing better performance on execution is better.
28	...food with more protein is better.
29	...cheaper building material is better.
30	...radio transmission with a higher frequency is better.
31	...sugar type which tastes sweeter is better.
32	...cheaper software is better.
33	...material which initially costs less for building roads is better.
34	...more elastic fabric is better.

Table 6.1.2.: Comments corresponding to the triples displayed in 6.1.1 to clarify the comparisons. For example in the comparison *earth* vs. *uranus* with respect to *mass*, it can be unclear if more or less mass is better.

6.2. Study Setup

6.2.1. Objectives

The goal of the study was an evaluation of CAM. To obtain this goal a comparison between CAM and a basic keyword search application was conducted. Both systems operated on the same dataset to make a fair comparison possible.

- CAM: The final version of the comparative argumentative machine (CAM) interface, as described in Subsection 4.3 and the latest backend features, as described in Chapter 3 (if not explicitly stated otherwise).
- Keyword search: A query interface based on Kibana was developed to be able to measure times of the user automatically. As an extension to Kibana, the interface removes duplicates to have a system that can be fairly compared to CAM. Furthermore, if an off-the-shelf search engine would be used, it would not be possible to use the identical dataset for both systems, which is also needed to have a fair comparison. In Figure 6.2.1 the interface is shown.

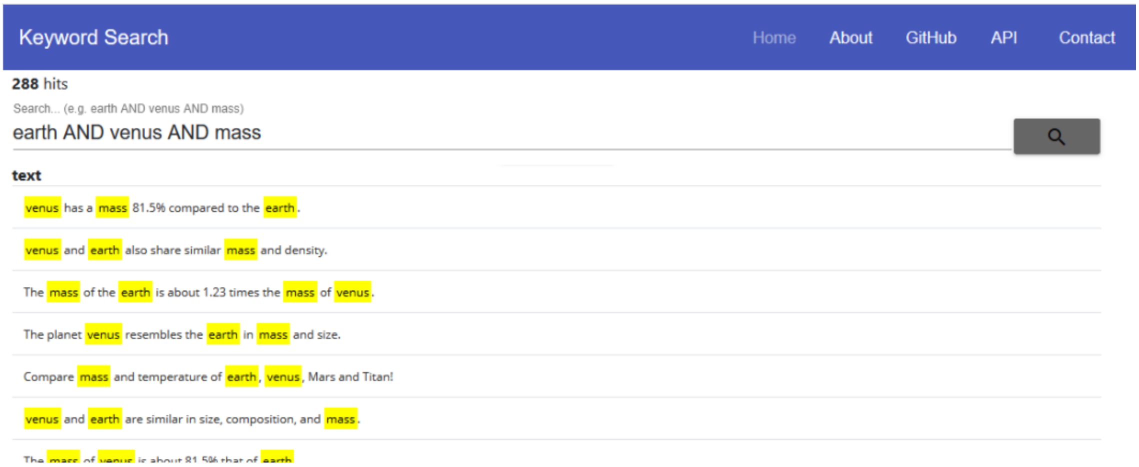


Figure 6.2.1.: The keyword search system used for the final study. For example, the header is kept just as in CAM to minimize the influence of different colors. It is possible to sent queries just like in *Kibana*[3].

6.2.2. Conduction

The study conduction included an alternating use of the two study systems to obtain a similar learning curve for both. For example, the participant had to get used to a foreign keyboard or to the way the comparisons are presented. To make the conduction as fair as possible for both systems, the system to start with was randomly selected. Furthermore, the processing order of the comparison triples was randomized to reduce order-biases.

[3]Lucene Query Syntax `https://www.elastic.co/guide/en/elasticsearch/reference/6.x/query-dsl-query-string-query.html` (accessed: 30.08.2018)

To not always select a triple for the same system, the randomization was paired with the triple index. All triples with an odd index were used for the system the participant started with, whereas all with an even index were assigned to the other system. Every triple was only used once per participant. The study generally took about one hour and ended with a questionnaire. The questionnaire contained free-form questions like: *Which feature did you like most? (Why?)* and *What would you like to improve? (Why?)*, checkable questions and questions for diversification. The participants had the opportunity to get redirected to the questionnaire after every second completed comparison to enable shorter and more spontaneous participation. In addition to the questionnaire in the end, before (see Figure 6.2.2) and after (see Figure 6.2.3) each comparison questions were asked.

6.2.3. User Task

The user task was kept just like in the first study. The user had to determine the winner of each comparison of the form shown in Table 6.1.1, by entering it into the system and analyze the given answer. All participants had to decide by themselves how much evidence is needed to give an answer. The possible answers again were the labels described at the beginning of Chapter 3 (*BETTER*, *WORSE* or $\neq$).

6.2.4. Measurements

Besides the questionnaire and the questions before and after each comparison (described above), two other metrics were used to measure and compare the system quality. One metric was the time the participant needs to obtain different phases of the comparison:

- *Time to start* is the time needed to orientate until the participant starts typing (the preparation time starts).
- *Preparation time*:
  - *Time typing* is the time the participant needs to type a query. This is represented by the difference of the *preparation* and the *system loading time*. It is not measured directly.
  - *System loading* is the time needed by the system to process the query until the result is presented.
- *Determination time*:
  - *Time reading*: This time starts when the results are showing until the participant clicks “Give Answer”.
  - *Time looking at the context*: When the participant decides to look at the context of a sentence, an extra time is taken. This additional measurement helps to make the *determination time* of CAM more comparable to the keyword search, where no context can be viewed.

The accuracy of the participants' answers was the second metric. After each comparison, the participant should give the answer to the processed comparison (as shown in Figure 6.2.3). To obtain a classification of the answers (if it is correct or incorrect) the gold labels of the evaluation dataset in Table 6.1.1 were used. If the answer is equal to the gold label it is correct, else it is incorrect.

As described above, before and after each comparison there were asked additional questions. The questions after (see Figure 6.2.3) were used to determine the confidence of the participants and how difficult the participants experienced the comparison (indirectly measuring if the use of CAM influences the experienced difficulty).

Before each comparison the participants were asked if they already know the answer (as shown in Figure 6.2.2) (if yes, the participant can submit what he/she thinks the answer is), e.g. to be able to later on discard comparisons that can be considered as to easy or use this information for further analysis.

6.2.5. Study System

For the purpose of automatic study conduction a system able to work without further given data, like a participation code or a list of comparison triples, was developed. The system itself generates a unique participation code to assign results to an anonymous user. That code also determines with which system the user had to start with and therefore the order of systems (as described above in Subsection 6.2.2).

In the beginning, the system shows an introduction (see Figures A.0.1, A.0.2, and A.0.3) on how to use the system and about the study objectives. In addition, a consent text is presented and accepted with continuing. When the participant completed the introduction, the start system is loaded and the first comparison triple is displayed on the top. Furthermore, the system asks if the participant knows the answer beforehand, as Figure 6.2.2 shows. It is also possible to return to the introduction text.

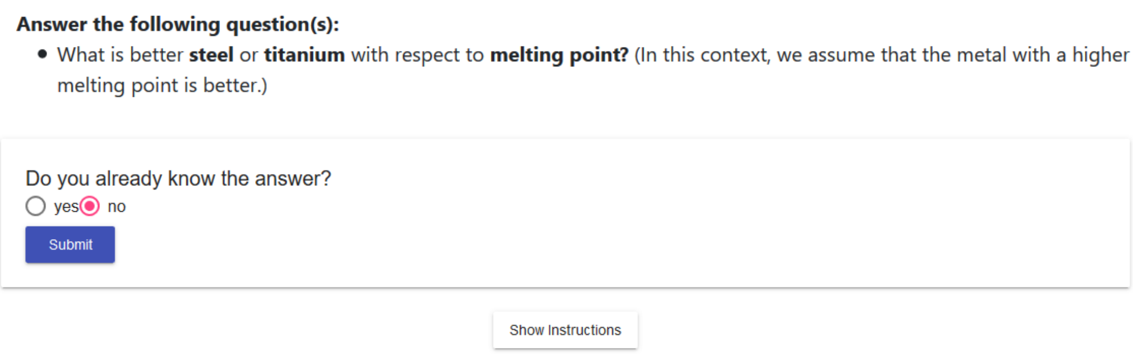


Figure 6.2.2.: Displayed by the study-system after reading the introduction (see Figures A.0.1, A.0.2 and A.0.3) and before the processing of every comparison. At this step, the user is able to reaccess the introduction and (after finishing an even number of comparisons, but at least two) the user also can finish the study by accessing the questionnaire.

The main step is to use the system to determine the winner of the given comparison.

CAM and keyword search interface are used as systems, as described in Subsection 6.2.1. After a participant found an answer and clicked “Give Answer”, the determined result has to be given and two questions should be answered, as shown in Figure 6.2.3.

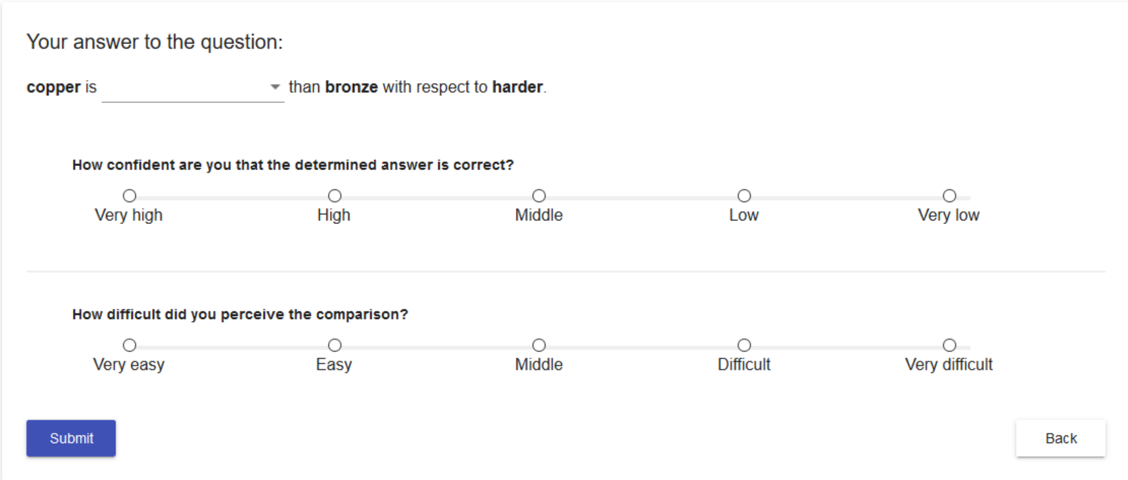


Figure 6.2.3.: Displayed by the study-system after the participant determined the answer to a given comparison and clicked “Give Answer”.

When one comparison answer is submitted, the next system and comparison are loaded. On completing all 34 triples, the participant automatically gets redirected to the final questionnaire.

6.3. Participants

6.3.1. Determine Needed Sample Size Using G*Power

To calculate the needed number of participants to measure a statistically significant improvement, G*Power [Faul et al., 2007] was used:
Test family: When analyzing the results of the first study (see Figure 5.5.1 and Figure 5.5.2), it was found that the probability distribution is a log-normal distribution. This result allowed to use the *t-test* on the logarithm of each value.
Statistical test: *Means: Difference between two dependent means (matched pairs)* was selected since the means for each triple grouped by the used system should be compared. Both systems were evaluated on the same dataset and therefore the two result sets are dependent.
Type of power analysis: *A priori* was selected because the required sample size should be estimated in advance.
Input parameters:

- **Tail:** *One* was selected since the test is only successful if CAM is faster than the other system and not the other way around (tail two).

- **Effect size:** In view of the results of the first study, a rather small effect size was expected. For a small effect size, *0.2* is suggested by [Cohen, 1988].
- α **err prob:** The G*Power default *0.05* was taken.
- **Power (1-β err prob):** The G*Power default *0.95* was taken.

Figure 6.3.1 summarizes the above-described parameters.

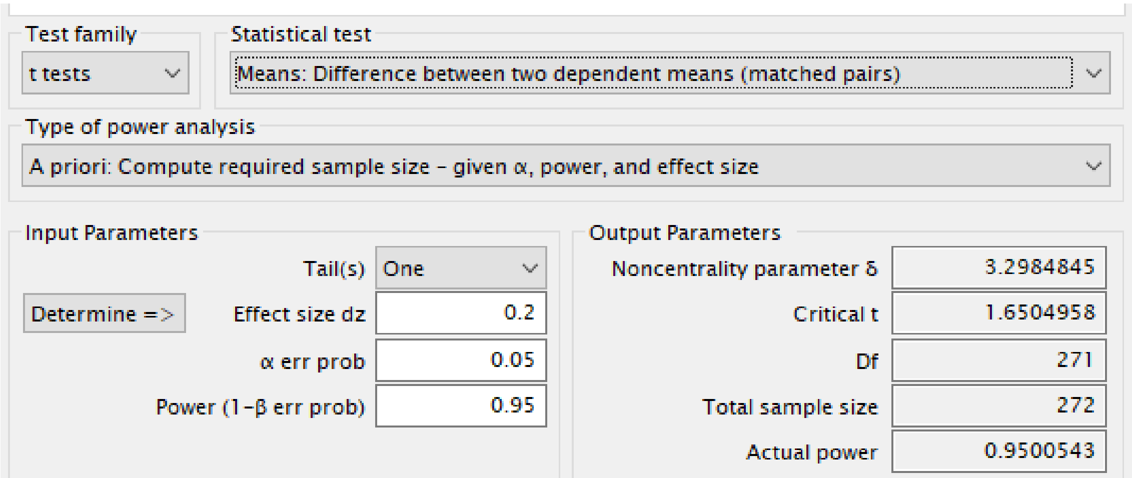


Figure 6.3.1.: The setup of the G*Power tool to determine the needed sample size. A total sample size of 272 was determined to show a statistically significant difference of the compared system results (if an effect size of 0.2 is assumed).

For the described parameters G*Power estimates a needed total sample size of 272 comparisons. In order to achieve the calculated sample size and to show a statistically significant effect, at least 8 participants were required processing all 34 comparisons.

6.3.2. Diversification

There were two different study setups for the participants.

In **Group A** 14 participants performed 477 comparisons; they were asked to process all 34 comparisons in a row without breaks. 13 participants were between 18 and 24 and one was between 25 and 34 years old (see Figure 6.3.2). Three of the participants are active in the *Arts, Culture & Entertainment* career field, eight in *Engineering & Computer Science*, one *Law & Public Policy* and two selected *other* as the answer (see Figure 6.3.6). Most participants (9) selected *Bachelor's degree* as Educational Background as can be seen in Figure 6.3.4. Participants had an intermediate (5 participants) or proficient (9) English level (Figure 6.3.5). As shown in Figure 6.3.3, five of the participants were female and nine were male. Finally, seven participants stated to use comparison websites rarely or never (once a year or less), whereas five use them once a month and two even once a week (shown in Figure 6.3.7).

In **Group B**, 9 participants performed 85 comparisons; they were free to test things out and to finish after a few comparisons. In this group, 5 participants were between

25 and 34, two between 18 and 24, one between 13 and 17 and one between 35 and 44 years old (see Figure 6.3.2). The selected carrier field again is dominated by *Engineering & Computer Science* (5 of 9). The following fields were selected by one participant each: *Education*, *Business*, *Arts, Culture & Entertainment* and *other*. The selections of the career fields are presented in Figure 6.3.6. Four participants own a Master's degree, two were students, one had a Bachelor's degree, one a Doctorate degree and there was one *other* selection (see Figure 6.3.4). They again were asked to rate their English level: the result is presented in Figure 6.3.5 — Six rated their level as proficient and three as intermediate. 5 of 9 participants were female and four male (see Figure 6.3.3). Finally, five participants stated to use comparison websites rarely or never (once a year or less), whereas two use them once a month and two even once a week or more, as can be seen in Figure 6.3.7.

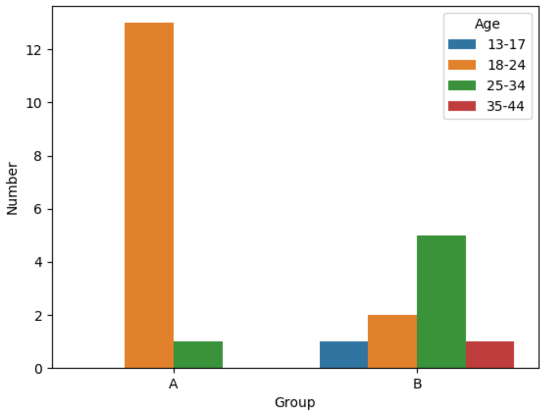


Figure 6.3.2.: The different selected age ranges for both groups (not selected are not shown).

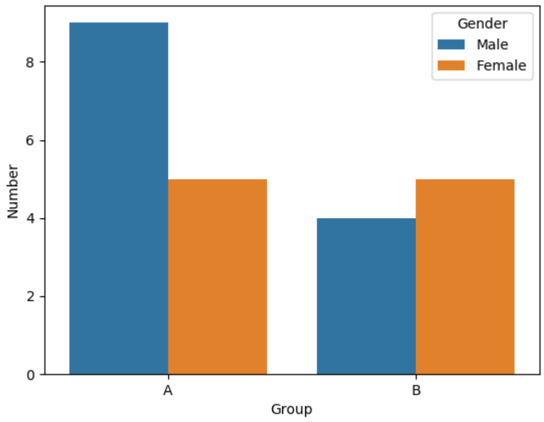


Figure 6.3.3.: The gender distribution for both groups.

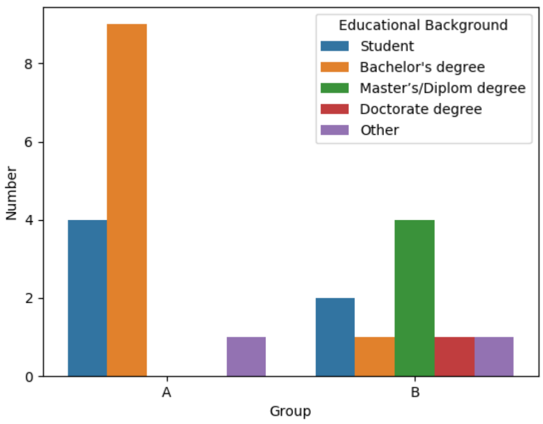


Figure 6.3.4.: The educational backgrounds of the participants, partitioned by the groups. Only options, which were selected at least once are presented.

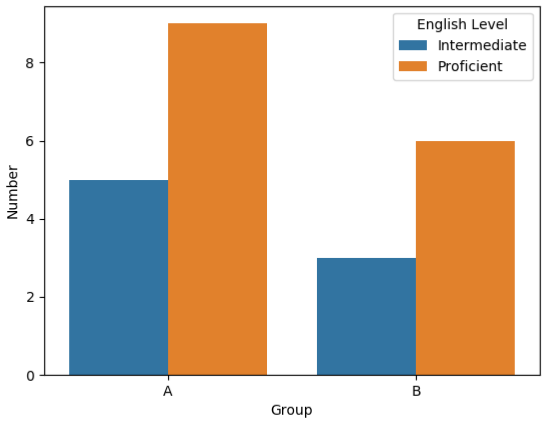


Figure 6.3.5.: The self-assessed English-levels of the participants for both groups. No one selected "Beginner" as level.

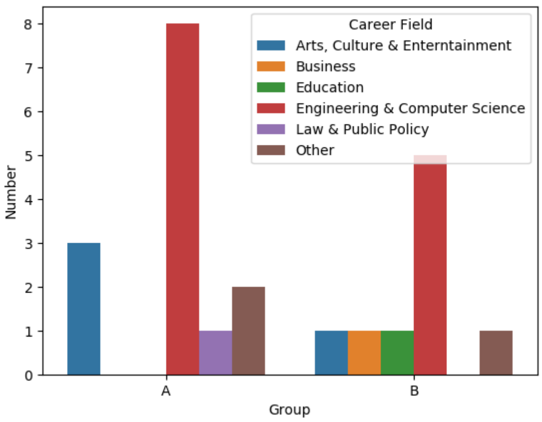


Figure 6.3.6.: The selected career fields are presented for both groups. Not selected fields are not shown here.

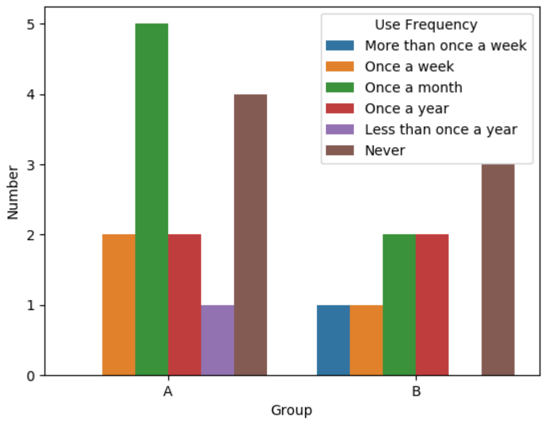


Figure 6.3.7.: The selected frequencies of comparison website use for both groups are presented. All options were selected at least once.

6.4. Discussion of the Results

A Shapiro-Wilk test [Shapiro and Wilk, 1965] on the logarithm of values was used to verify the visual assumption of a log-normal distribution. For $\alpha = 0.05$, H_0 was accepted for the determination (p-value: 0.06) and total time (p-value: 0.29) needed using CAM and the total time using the keyword search. For the determination time of the keyword search the test failed relatively scarce (p-value: 0.0006). However, for the total time needed using the keyword search it was accepted (p-value: 0.25). Therefore, a t-test can be used.

6.4.1. Time Measurements

As described in Subsection 6.2.4, a variety of phases was timed while using the systems. In Figure 6.4.1 the underneath described results are visualized.

The *until typing* boxes present the times' participants needed to orientate and to start typing when the system is shown, see Figure 6.4.1 for Group A and Figure 6.4.2 for Group B. The participants needed about 19% less time on CAM and the measured times varied less than on keyword search for A. For B they needed about 25% less, but they spent more time in general.

Typing is the time measured from the first key hit until the query is sent. The participants again needed less time in Group A with CAM (about 24% less on average). For Group B the difference is very small, but also in favor of CAM. The participants of Group B needed about twice as long as the ones of Group A.

The *Loading* phase contains the values the system needs to process the answer (from sending the query until the result is presented). On average keyword search loads faster than CAM, but since it is a hardware-dependent time it is not too relevant for our study.

Most importantly, measuring the quality of answer presentation by the time users need to give the answer (*determination* in Figure 6.4.1 for Group A and Figure 6.4.2 for Group

B), A was statistical significantly ($t(474) = 5.86$, $p < 0.00001$) faster (by about 39%) using CAM. As the figure shows, the effect is strong for the determination time (Cohen's d [Cohen, 1988] = 0.54). In B the participants were slower in general, but interestingly they were slightly slower using CAM, probably because they tried out more and gave comments about the interface during the trial.

For the overall task, Group A was significantly faster, using CAM ($t(474) = 4.31$, $p < 0.00001$) for Group B the time needed was almost equal for both systems with a slight advantage for the keyword search. In A, a little smaller (but still significant) effect size than for determination time is reached (Cohen's d [Cohen, 1988] = 0.4).

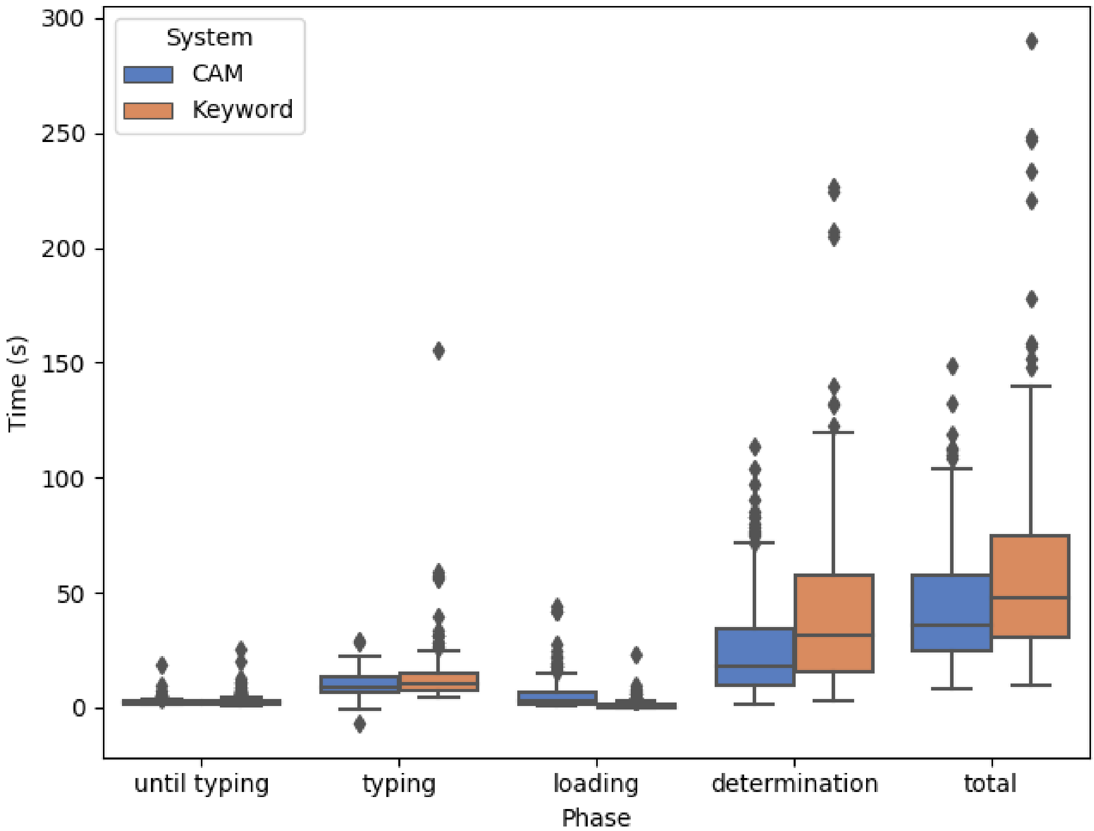


Figure 6.4.1.: Times of question answering phases (Group A). Except for the time needed to load the answers, the participants were faster, using CAM in comparison to the keyword search.

Each individual comparison triple was processed by six to eight participants for both systems. Summarizing these, it shows that 25 of 34 were processed faster using the CAM compared to the keyword search when looking at the medians (see Figure 6.4.3). For mean values, there are also 25 that were less time-consuming using the CAM. *(hdmi, dvi, quality)* and *(ccd, cmos, power)* are two out of five triples where median **and** mean were lower (better) using the keyword search. For the first one (index 15) all participants using keyword search determined the right label (NONE), whereas in CAM only 62.5% determined the right label. Analyzing the given results of both systems shows a shortcoming of CAM: for the equality of objects (included in the $\neq$ label) CAM can not give an appropriate answer, since a sentence like "*Yeah, DVI and HDMI provide identical quality.*", given

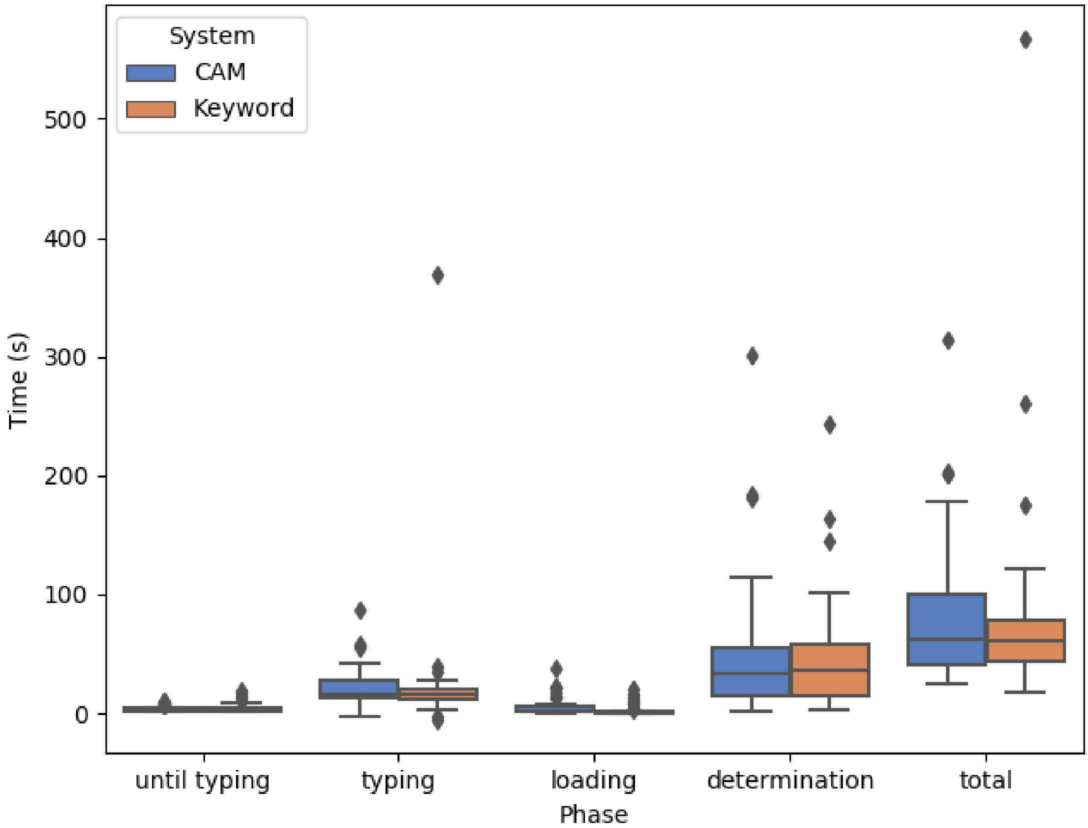


Figure 6.4.2.: Times of question answering phases (Group B). The participants needed more total time using CAM in comparison to the keyword search. For determination time there is a slight advantage using CAM. The slower time of CAM probably was needed because the participants were allowed to play around and comment on the interface.

by keyword search, does not contain any comparison (and therefore is not included by CAM), but gives exactly the answer to the comparison question. The given sentences for the second triple (index 10), working better on keyword search, are very expedient for both systems. An assumption, why the participants were faster, using keyword search is, that it can be beneficial to have less information (only one column with sentences) for comparisons where the answer is very clear.

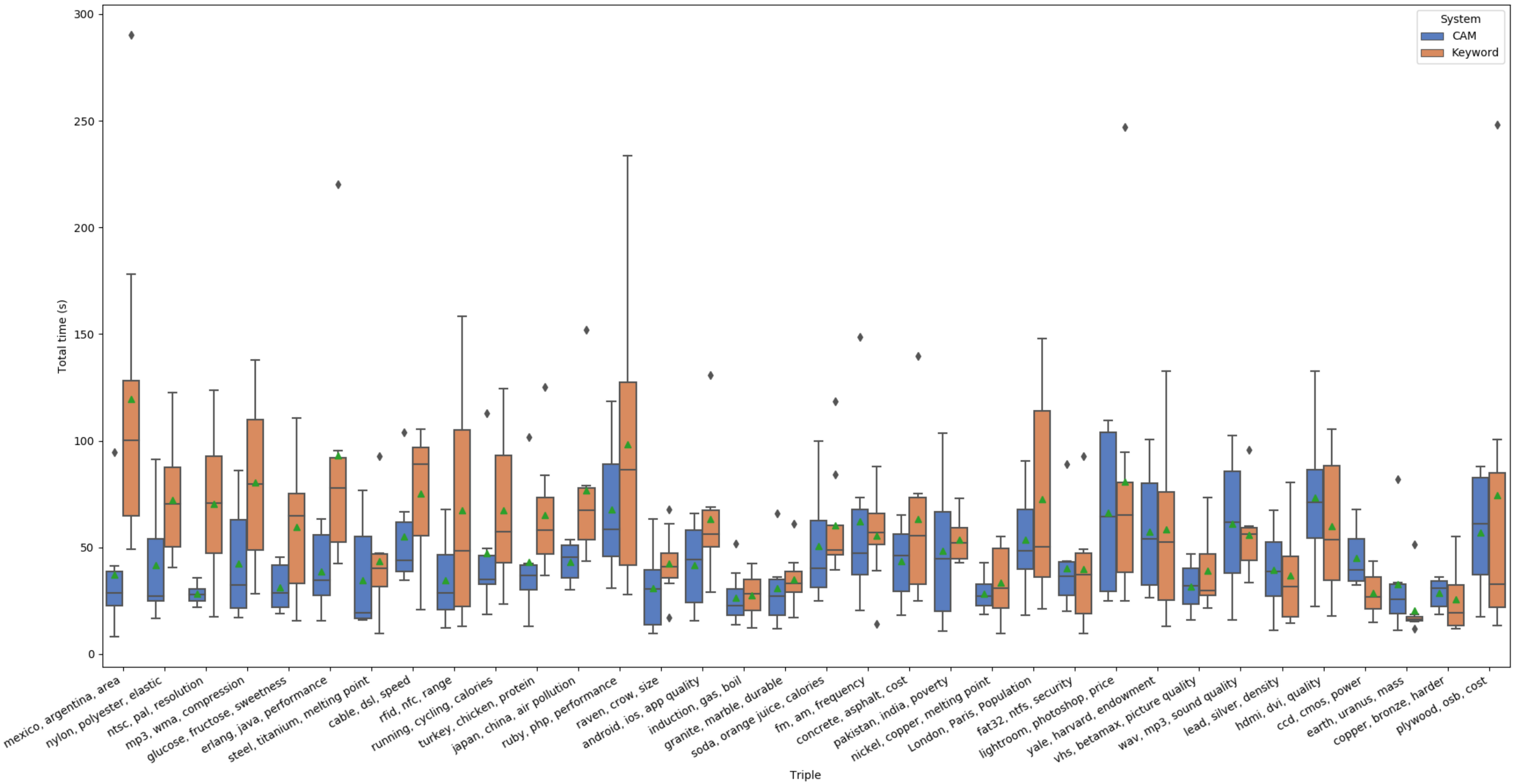


Figure 6.4.3.: The total times of all 34 triples for both systems. The triples (A, B, c_k) are ordered by the difference of medians and selected as labels. The green triangles represent the mean of the measurements. 25 of 34 triples were processed faster using CAM (according to the median and also according to the mean).

6.4.2. Accuracy, Confidence, and Difficulty

Furthermore, the participants using CAM made fewer errors: For CAM an average accuracy of about 95% was reached (9 of 14 participants reached 100%), whereas for the keyword-search 81% was reached (the best participant reached 94%). The described results are visualized in Figure 6.4.4. The participants of Group B also were more accurate using CAM, but only 84% against 75% as shown in Figure 6.4.5.

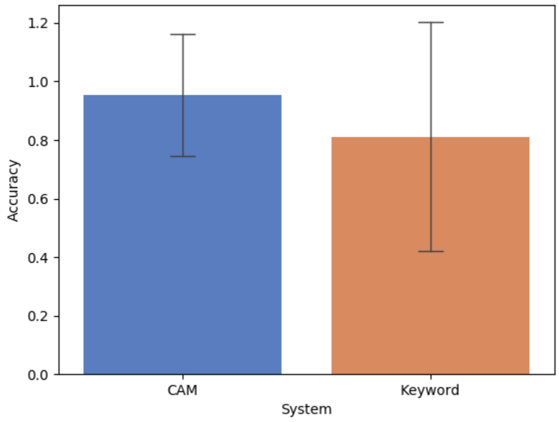


Figure 6.4.4.: The accuracies of question answers (Group A).

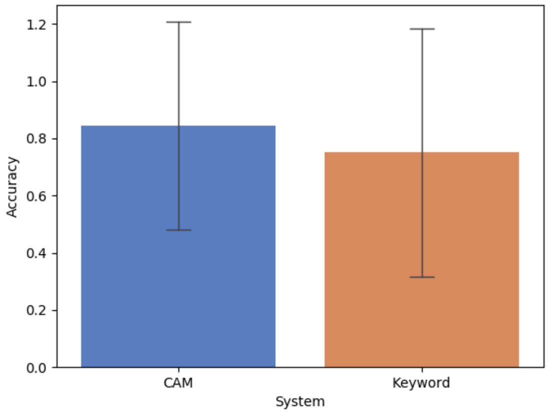


Figure 6.4.5.: The accuracies of question answers (Group B).

In addition, on a scale from 1 to 5 where 5 was the best, the participants of both groups on average were almost one point more confident that the answer is correct. The selection rations of Group A are visualized in Figure 6.4.6, whereas the selection ratios of Group B are visualized in Figure 6.4.7.

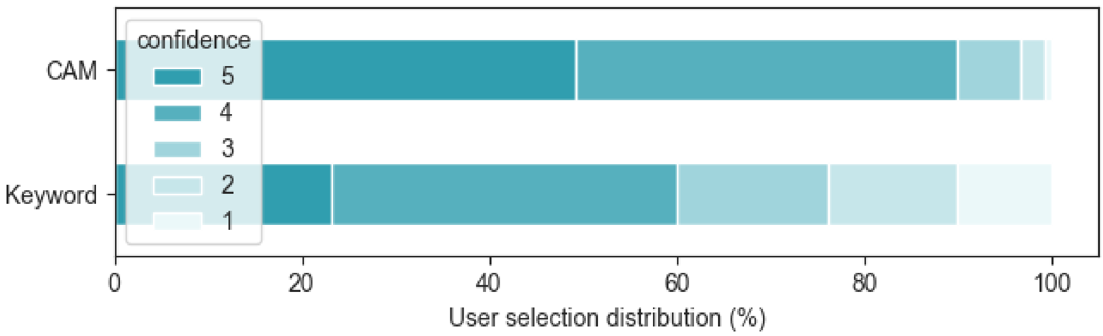


Figure 6.4.6.: User answers on the question "How confident are you that the determined answer is correct?" (Group A).

The participants also were asked how difficult they perceived the comparison they processed using the same scale. For Group A on average, the participants selected a value almost one point higher after using CAM (4.04, which is rather *Easy* against 3.08, which is rather *Neutral*). For Group B on average, the participants even selected a value over one point higher after using CAM (4.24 against 2.95). The ratios of selections of A are presented in Figure 6.4.8 and the rations of selections of B are presented in Figure 6.4.9.

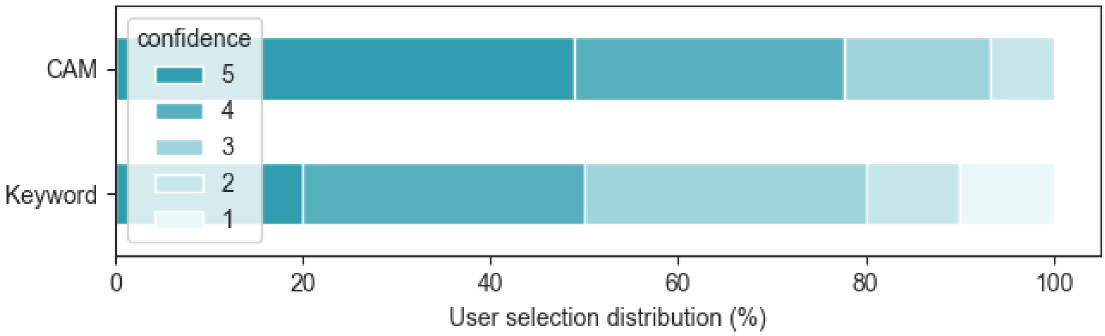


Figure 6.4.7.: User answers on the question "How confident are you that the determined answer is correct?" (Group B).

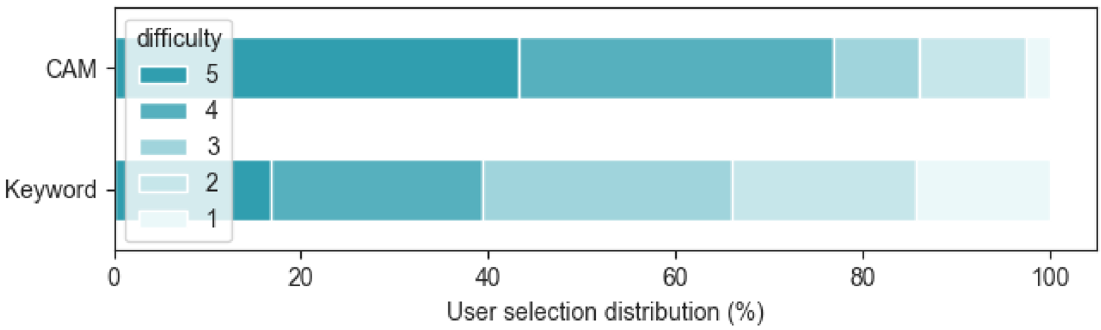


Figure 6.4.8.: User answers on the question "How difficult did you perceive the comparison?" (Group A).

6.4.3. Questionnaire

The questionnaire answers always ranged from 1 (positive) to 5 (negative). The question "How convenient was it to use CAM?" and the statement "Learning the usage of CAM is..." on average achieved values between one and two for both groups, which is very positive. On the subjective statement "I spent more time on..." a low selected value (1 or 2) indicates a higher time consumption using CAM. 1 never got selected and 2 only once. Interestingly, the only participant that selected 2, still had an overall higher (about 35%) time consumption on the keyword search. This participant stated that he thinks that the keyword search is faster due to the less needed mouse operations. In Figure 6.4.10 and

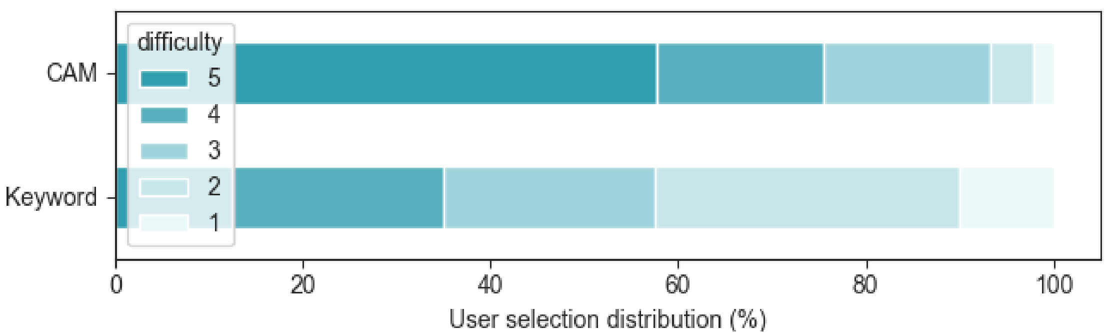


Figure 6.4.9.: User answers on the question "How difficult did you perceive the comparison?" (Group B).

Figure 6.4.11 the selection ratios are presented.

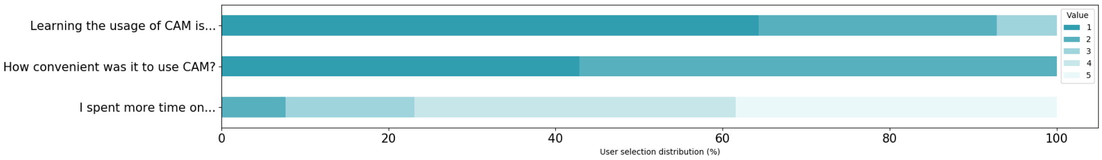


Figure 6.4.10.: User selection ratios for the given questions. The answers ranged from 1 (positive) to 5 (negative) (Group A).

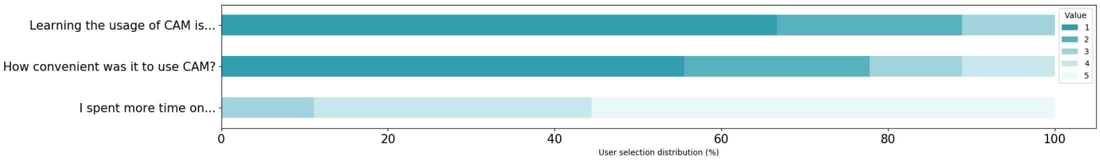


Figure 6.4.11.: User selection ratios for the given questions. The answers ranged from 1 (positive) to 5 (negative) (Group B).

Many participants explicitly mentioned the developed score presentation when answering the question: "Which feature did you like most? (Why?)". For example, one participant answered: "*The graphics made the answers very clear and I felt like when using the CAM the statements found were a lot more helpful than the ones coming up in the Keyword Search.*", furthermore, another wrote: "*the green bar for cam was amazing, however, I was always trying to find a sentence that proves the green indicator bar right.*" and just another answered: "*summarization bar makes the decision easy*" just to name a few of the many positive answers.

6.5. Conclusion

The second user study evaluated the final developed version of CAM and was able to show, that it is not only statistically significantly faster to use CAM, instead of a keyword search (about 39%), but also allows the user to determine a correct answer for most comparisons (which is not the case for the keyword search).

7. Conclusion and Future Work

This thesis dealt with the creation and evaluation of a system capable of answering comparative questions from an arbitrary domain (open domain). Two user studies were conducted to assess the system quality: the first evaluated a prototype and gave invaluable information about how to improve the answer presentation to allow a faster answer determination of users and the second evaluated the final contribution of this thesis. The main frontend feature decisions are taken on basis of guidelines described in [Shneiderman, 2010] and [Johnson, 2014], whereas the main backend feature decisions are taken on basis of calculated scores and manual examination. The second user study was conducted in two different environmental settings and was able to show that the score presentation, designed as multiple charts for different smaller scores, was well accepted and liked by the participants. Nevertheless, they also needed confirmation by finding the evidence in presented answer sentences.

As described in Section 6.4, the participants not only were able to achieve statistical significant faster times (about 39%) determining the answer and in total (about 27%) but also achieved a higher accuracy answering comparisons using CAM in comparison to a standard keyword search. Furthermore, all participants were more confident about given answers and assessed the comparisons easier after using CAM.

However, some aspects were not covered in this thesis. At the end of Subsection 6.4.1, individual triples were analyzed and it is stated that a comparison, comparing objects that are equal with respect to an entered aspect, cannot be satisfactorily answered. Furthermore, the introduced heuristic to handle negated sentences and place them to the *correct* site improved the sentence assignment, but still is not able to solve the basic problem of showing contrary sentences for the same object. To achieve a better negation handling it may be appropriate to add such feature to the machine learning classifier described in [Franzek et al., 2018] and used in the system. The described results are promising, but the system was not tested in a real-world application, like a search engine. So the results could just be good for the selected comparisons. However, the studies had a wide range of domains (e.g. materials, programming languages, countries and data formats), which is an important factor for generalizability in this case.

Due to the limited time of the project, not all planned features could be implemented and tested, which makes them possible future extensions of the developed system.

One of these features is to generate a natural language answer based on the evidence sentences. Two participants even demanded exactly that feature in the questionnaire

(“*An answer formulated in natural language with links to sources would be nice.*”) when asking them what they would like to improve. Such a feature could very well take the score with percentages as a basis to estimate how much better one object compared to the other is.

Another future work on the project could be to not only search exactly for the user entered aspect but also search for synonyms, for example, extracted from WordNet [Miller, 1995].

Another system extension could be to improve the precision by using a structured data source like DBPedia [Auer et al., 2007]. The used unstructured (CommonCrawl) data source has the ability to cover a broad spectrum of comparisons, whereas the structured data may be more precise for contained objects.

Auto-completion is an extension that widens the system's capabilities and allows the user to explore comparisons for an entered object. To realize this feature the in Figure 7.0.1 presented steps can be used.

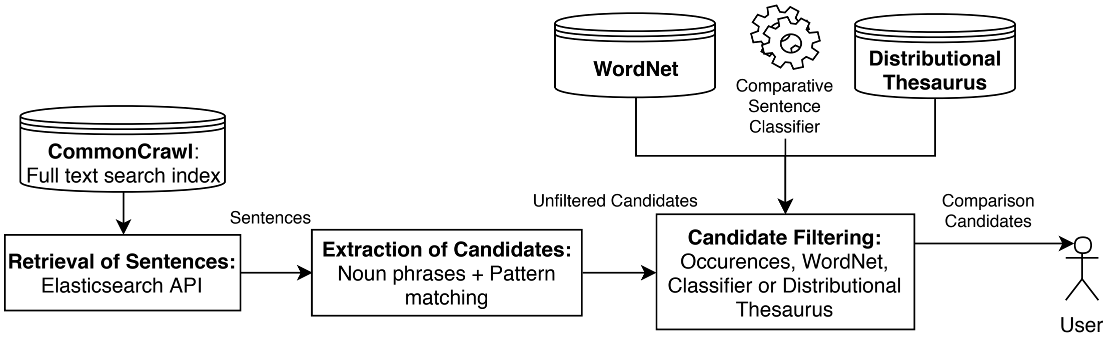


Figure 7.0.1.: The steps for receiving comparison candidates.

First, to retrieve candidates for further filtering steps, the big size of the in Section 3.1 described corpus can be exploited: Sentences containing the entered object (e.g. *Python*) and “*vs*” are queried to obtain all sentences explicitly comparing the entered object with another one (comparison candidate). To extract the candidates from queried sentences, dependency parsing, and regular expressions can be used. Dependency parsing can be used to only consider noun phrases as candidates. Regular expressions can be used to check the positions of the noun phrases: if the position is next to *vs* or *vs.* and on the other side is the entered object, the noun phrase is taken into account as comparison candidate. To evaluate, the suggestions by Google when entering “"*<object> vs*"” were retrieved and used as a gold set. For example for *Python* the candidates after using the first two steps of the approach (sorted by the number of occurrences; see Figure 7.0.3) already contain a good amount of the retrieved suggestions (80% in total and 60% in the first ten) given by Google (see Figure 7.0.2). However, for all the first objects of the evaluation triples presented in Table 5.2.1 on average only about 33% are contained. Nevertheless, there are candidates beyond the ones suggested by Google, for example for *Java*, Google does not suggest *.Net* and *Ruby*, although they are also legit comparison candidates. Taking that observation into account, the comparison with Google suggestions should only be viewed as a reference.

Figure 7.0.2.: Suggestions shown by Google when entering “*"Python vs"*”.

perl	lua	gator	cython
java	gatoroid	matlab	tiger
ruby	matlab gc	ruby deathmatch	jlizard
php	ruby ruby	print 'weave	arc
boa	brython	print 'f2py	lisp
alligator	matlab/eeglab	jython	gql
julia	prothon	python-novaclient	film boa
net	deer	rhinoscript	africanized honey-
c++	aqueon	octave	bee
visual	ruby performance	cockatoo photos	node
javascript	alligator watch	kruger	stones
qml	thinking up-	python programs	
crocodile	side down ruby	profiling pypy	
cat	sas	pycuda	

Figure 7.0.3.: The comparison candidates after the first and second processing step. Sorted by number of occurrences (column wise and from left to right). The candidates emphasized in green are also contained in the Google search presented in Figure 7.0.2.

However, the majority of the extracted comparison candidates for *python* are not of a big advantage for a user, for example, *ruby deathmatch*, *africanized honeybee* or *stones* (see Figure 7.0.3) are candidates that should be filtered out, which is done in the third part of the pipeline (Figure 7.0.1). All approaches reduce the number of comparison candidates to a maximum of ten. As extracted candidates are already ranked by the number of above described explicit comparisons, the easiest option to filter is to just take the ten highest ranked candidates. Another approach can be to look for common hypernyms, to see if it makes sense to compare the entered aspect with the candidate. To do so WordNet [Miller, 1995] was tried, but was too sparse and therefore filtered out too many candidates. A more complex approach can be to use the machine learning classifier described in Section 3.3 to retrieve the number of comparative sentences, comparing the entered object and the candidate. That number can be used to filter out candidates with a low amount and to rank candidates with many comparative sentences higher. However, the approach takes very long if the amount of comparative sentences is determined for every extracted comparison candidate, therefore some kind of database would be needed to enable this approach for a real-time usage. The last approach described and evaluated uses a Distributional Thesaurus (DT) [Biemann et al., 2013] to filter out candidates not similar enough to compare it to the entered object. Elasticsearch can be used to make the DT available and to query the similar objects to the entered one. If an extracted candidate also is contained as s similar entity, it is considered as comparison candidate for the feature.

To evaluate the different filter approaches as described above the first objects of the evaluation triples presented in Table 5.2.1 are used. And for all of them, the suggestions by Google (as presented for *python* in Figure 7.0.2) were retrieved and used as a gold set. The average percentages of correctly contained candidates are presented in Table 7.0.1. Surprisingly, the most basic filtering leads to the best results. Further improvement of the described approach just as the implementation of the approach can be part of future work.

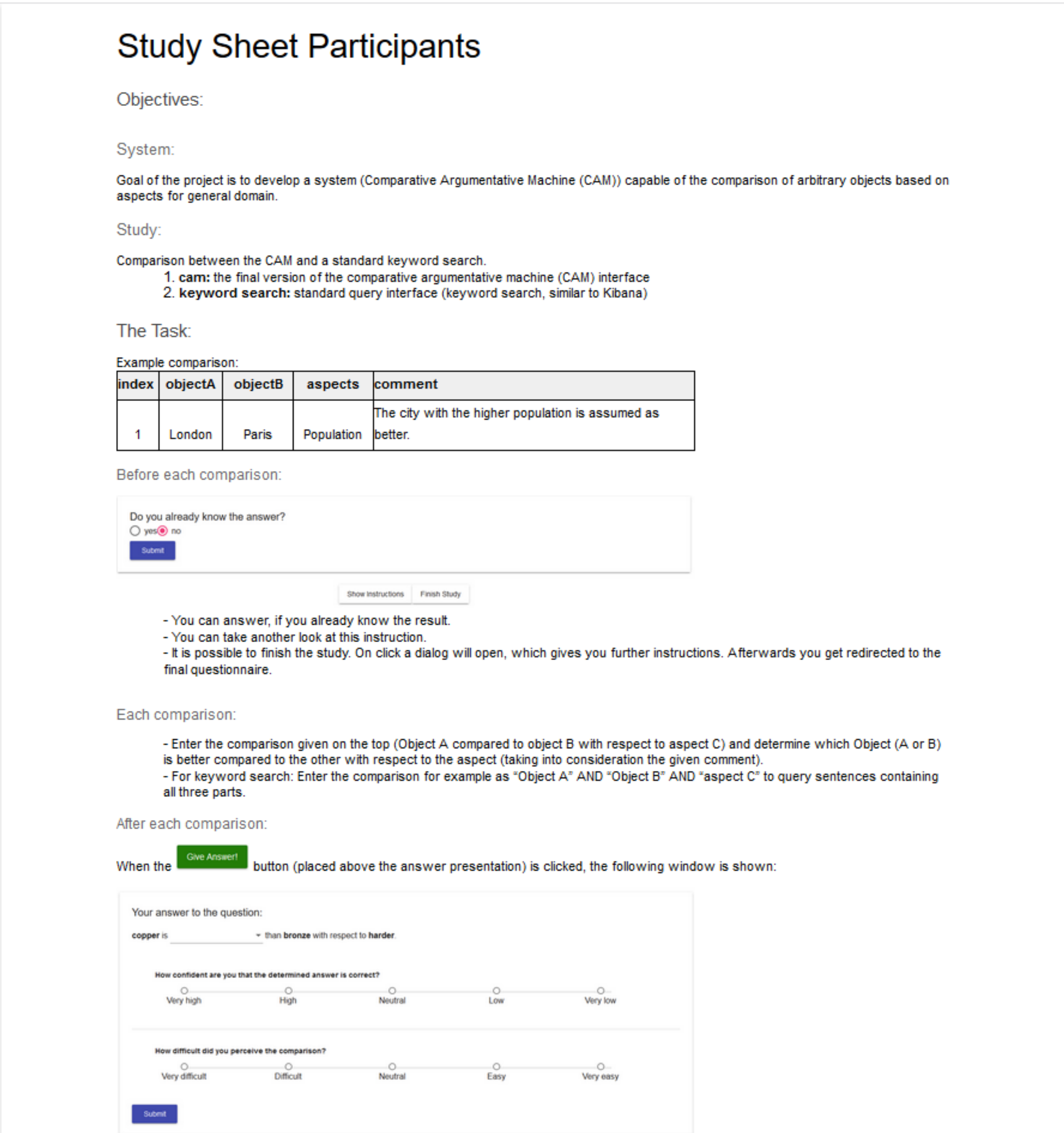
Basic Filtering	WordNet	Classifier	Distributional Thesaurus
18.85%	9.8%	17.33%	15.45%

Table 7.0.1.: Percentages of how many candidates match the ones suggested by Google. The ranking is not taken into account.

ACKNOWLEDGMENTS

I want to thank Alexander Bondarenko, who conducted the user study analyzed in Group B, described in Chapter 6. Furthermore, I want to thank Julian Zenker, who developed the very first prototype of CAM in a two-week Bachelor project. In addition, he developed the aspect extraction approaches, described in Section 3.5, within the scope of his student assistant occupation.

A. Study System Instructions



Study Sheet Participants

Objectives:

System:

Goal of the project is to develop a system (Comparative Argumentative Machine (CAM)) capable of the comparison of arbitrary objects based on aspects for general domain.

Study:

Comparison between the CAM and a standard keyword search.

1. **cam:** the final version of the comparative argumentative machine (CAM) interface
2. **keyword search:** standard query interface (keyword search, similar to Kibana)

The Task:

Example comparison:

index	objectA	objectB	aspects	comment
1	London	Paris	Population	The city with the higher population is assumed as better.

Before each comparison:

- You can answer, if you already know the result.
- You can take another look at this instruction.
- It is possible to finish the study. On click a dialog will open, which gives you further instructions. Afterwards you get redirected to the final questionnaire.

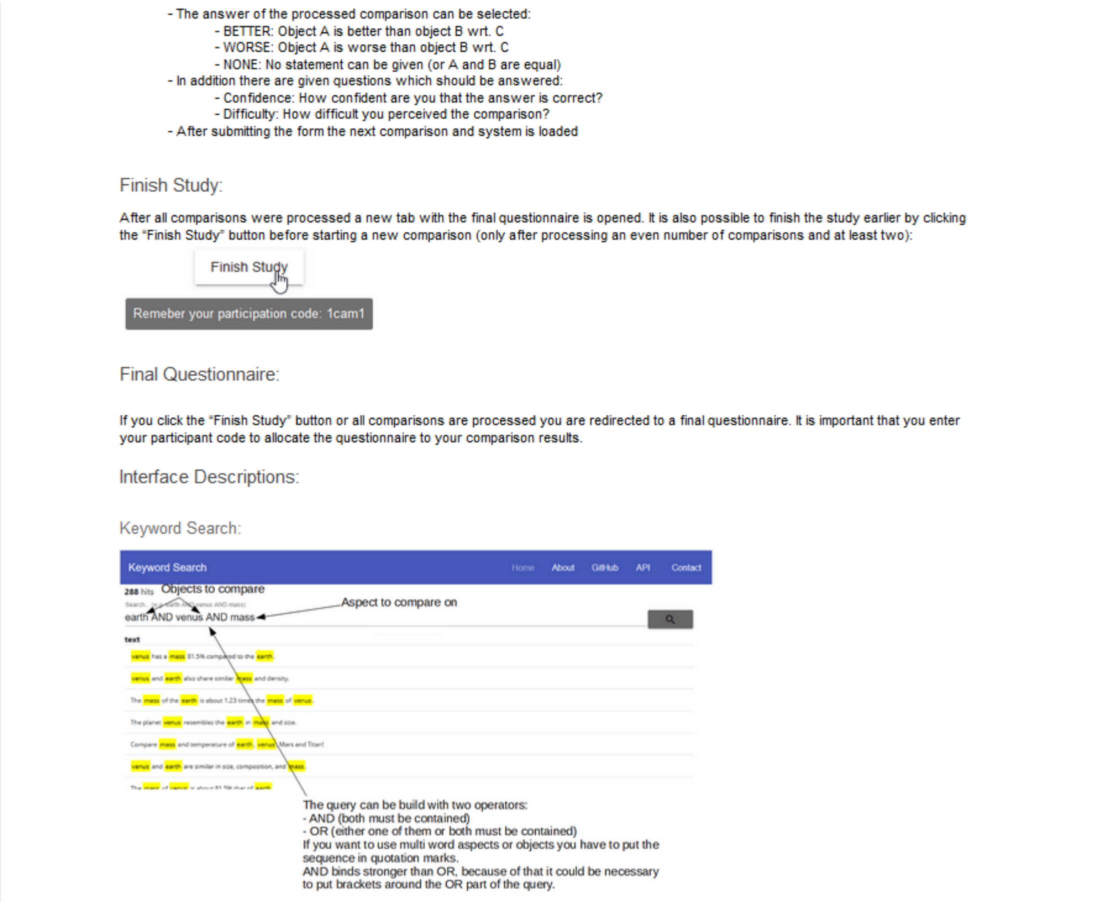
Each comparison:

- Enter the comparison given on the top (Object A compared to object B with respect to aspect C) and determine which Object (A or B) is better compared to the other with respect to the aspect (taking into consideration the given comment).
- For keyword search: Enter the comparison for example as "Object A" AND "Object B" AND "aspect C" to query sentences containing all three parts.

After each comparison:

When the Give Answer! button (placed above the answer presentation) is clicked, the following window is shown:

Figure A.0.1.: The first part of the instruction of the study system used for the main study.



- The answer of the processed comparison can be selected:
  - BETTER: Object A is better than object B wrt. C
  - WORSE: Object A is worse than object B wrt. C
  - NONE: No statement can be given (or A and B are equal)
- In addition there are given questions which should be answered:
  - Confidence: How confident are you that the answer is correct?
  - Difficulty: How difficult you perceived the comparison?
- After submitting the form the next comparison and system is loaded

Finish Study:

After all comparisons were processed a new tab with the final questionnaire is opened. It is also possible to finish the study earlier by clicking the "Finish Study" button before starting a new comparison (only after processing an even number of comparisons and at least two):

Finish Study

Remeber your participation code: 1cam1

Final Questionnaire:

If you click the "Finish Study" button or all comparisons are processed you are redirected to a final questionnaire. It is important that you enter your participant code to allocate the questionnaire to your comparison results.

Interface Descriptions:

Keyword Search:

Figure A.0.2.: Second part of the instruction of the study system used for the main study.

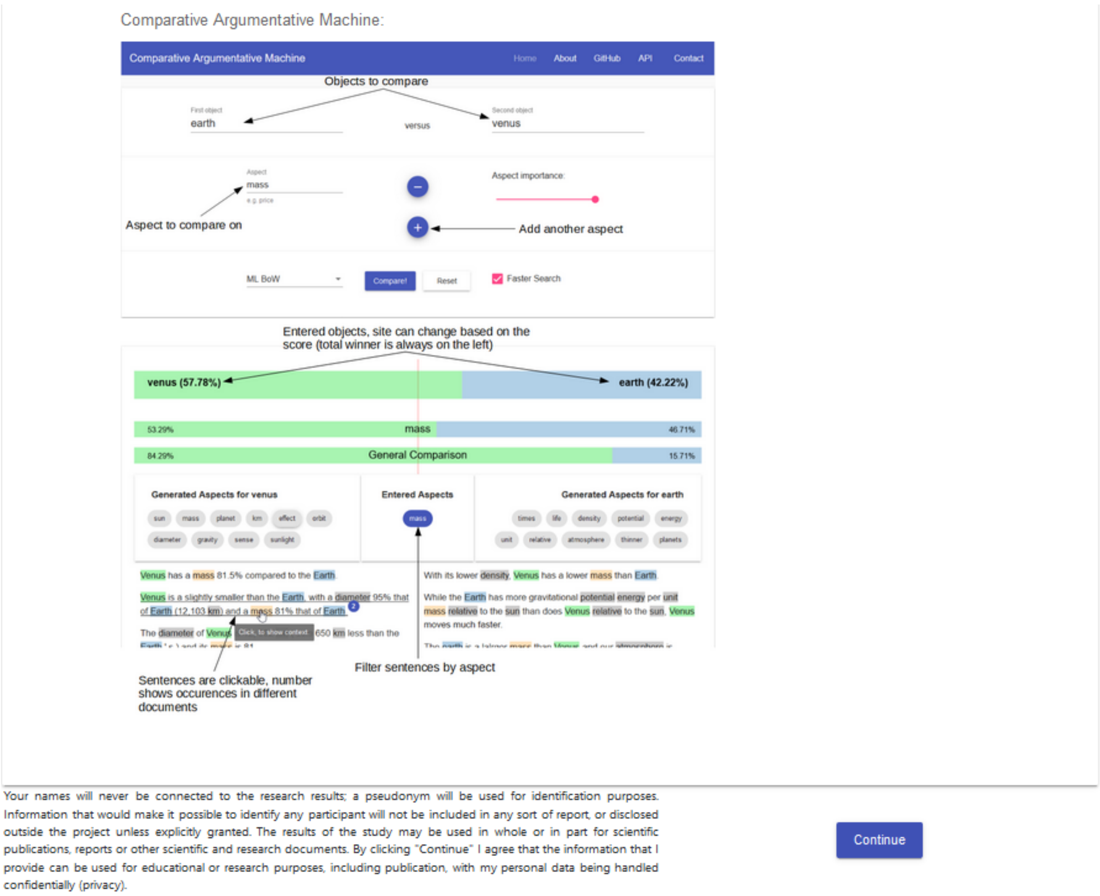


Figure A.0.3.: Third part of the instruction of the study system used for the main study.

Bibliography

[Auer et al., 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). DBpedia: A Nucleus for a Web of Open Data. In *The semantic web*, pages 722–735. Springer.

[Biemann et al., 2013] Biemann, C., Coppola, B., Glass, M. R., Gliozzo, A., Hatem, M., and Riedl, M. (2013). JoBimText Visualizer: A Graph-based Approach to Contextualizing Distributional Similarity. In *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing, Seattle, Washington, USA*, pages 6–10. Association for Computational Linguistics.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, San Francisco, CA, USA*, pages 785–794. ACM.

[Choi et al., 2011] Choi, K., Pacaña, R. M., Tan, A. L., Yiu, J., and Lim, N. R. (2011). Processing Comparisons and Evaluations in Business Intelligence: A Question Answering System. In *Uncertainty Reasoning and Knowledge Engineering (URKE), 2011 International Conference on Uncertainty Reasoning and Knowledge Engineering, Bali, Indonesia*, volume 1, pages 137–140. IEEE.

[Cockburn et al., 2009] Cockburn, A., Karlson, A., and Bederson, B. B. (2009). A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Computing Surveys (CSUR)*, 41(1):2.

[Cohen, 1988] Cohen, J. (1988). *Statistical power analysis for the behavioral sciences. 2nd*. Hillsdale, NJ: erlbaum.

[Conneau et al., 2017] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark*, pages 670–680. Association for Computational Linguistics.

[Cutrell and Guan, 2007] Cutrell, E. and Guan, Z. (2007). Eye tracking in MSN Search: Investigating snippet length, target position and task types. In *Proc of ACM Conf. on on Human Factors in Computing Systems*, pages 407–416. MSR-TR-2007.

[Daxenberger et al., 2017] Daxenberger, J., Eger, S., Habernal, I., Stab, C., and Gurevych, I. (2017). What is the Essence of a Claim? Cross-Domain Claim Identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark*, pages 2055–2066. Association for Computational Linguistics.

[Diefenbach et al., 2017] Diefenbach, D., Amjad, S., Both, A., Singh, K., and Maret, P. (2017). Trill: A Reusable Front-End for QA Systems. In Blomqvist, E., Hose, K., Paulheim, H., Lawrynowicz, A., Ciravegna, F., and Hartig, O., editors, *The sematic web: ESWC 2017 satellite events*, volume 10577 of *Lecture Notes in Computer Science*, pages 48–53. Springer.

[Dimara et al., 2018] Dimara, E., Bezerianos, A., and Dragicevic, P. (2018). Conceptual and Methodological Issues in Evaluating Multidimensional Visualizations for Decision Support. *IEEE transactions on visualization and computer graphics*, 24(1):749–759.

[Dumais et al., 2001] Dumais, S., Cutrell, E., and Chen, H. (2001). Optimizing Search by Showing Results In Context. In *Proceedings of the SIGCHI conference on Human factors in computing systems, Seattle, Washington, USA*, pages 277–284. ACM.

[Faul et al., 2007] Faul, F., Erdfelder, E., Lang, A.-G., and Buchner, A. (2007). G* Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior research methods*, 39(2):175–191.

[Franzek et al., 2018] Franzek, M., Panchenko, A., and Biemann, C. (2018). Categorization of Comparative Sentences for Argument Mining. *arXiv preprint arXiv:1809.06152*.

[Gupta et al., 2017] Gupta, S., Mahmood, A. S. M. A., Ross, K., Wu, C. H., and Vijay-Shanker, K. (2017). Identifying Comparative Structures in Biomedical Text. In *BioNLP 2017, Vancouver, Canada, August 4, 2017*, pages 206–215. Association for Computational Linguistics.

[Hoque et al., 2017] Hoque, E., Joty, S., Marquez, L., and Carenini, G. (2017). CQAVis: Visual Text Analytics for Community Question Answering. In Papadopoulos, G. A., editor, *IUI'17*, pages 161–172, New York, New York, USA. ACM Press.

[Hua and Wang, 2017] Hua, X. and Wang, L. (2017). Understanding and Detecting Supporting Arguments of Diverse Types. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Vancouver, Canada*, pages 203–208.

[Jean-Baptiste et al., 2016] Jean-Baptiste, L., Berthelot, H., and Favre, M. (2016). Rainbow Boxes: A Technique for Visualizing Overlapping Sets and an Application to the Comparison of Drugs Properties. In *Information Visualisation (IV), 2016 20th International Conference, Lisbon, Portugal*, pages 253–260. IEEE.

[Johnson, 2014] Johnson, J. (2014). *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann Publishers Inc.

[Joshi and Akerkar, 2008] Joshi, M. and Akerkar, R. (2008). Algorithms to Improve Performance of Natural Language Interface. *International Journal of Computer Science and Applications (IJCSA)*, 5(2):52–68.

[Lamy et al., 2017] Lamy, J.-B., Berthelot, H., Favre, M., Ugon, A., Duclos, C., and Venot, A. (2017). Using visual analytics for presenting comparative information on new drugs. *Journal of biomedical informatics*, 71:58–69.

[Leonhard, 2009] Leonhard, A. (2009). Towards Retrieving Relevant Information for Answering Clinical Comparison Questions. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing, Boulder, Colorado*, pages 153–161. Association for Computational Linguistics.

[Leung and Apperley, 1994] Leung, Y. K. and Apperley, M. D. (1994). A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 1(2):126–160.

[Lin et al., 2003] Lin, J., Quan, D., Sinha, V., Bakshi, K., Huynh, D., Katz, B., and Karger, D. R. (2003). What Makes a Good Answer? The Role of Context in Question Answering. In *Proceedings of the Ninth IFIP TC13 International Conference on Human-Computer Interaction (INTERACT 2003), Zurich, Switzerland*, pages 25–32.

[Lippi and Torroni, 2016] Lippi, M. and Torroni, P. (2016). Argumentation Mining: State of the Art and Emerging Trends. *ACM Trans. Internet Technol.*, 16(2):10:1–10:25.

[Miller, 1995] Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

[Moghaddam and Ester, 2010] Moghaddam, S. and Ester, M. (2010). Opinion digger: an unsupervised opinion miner from unstructured product reviews. In *Proceedings of the 19th ACM international conference on Information and knowledge management, Toronto, Canada*, pages 1825–1828. ACM.

[Moghaddam and Ester, 2011] Moghaddam, S. and Ester, M. (2011). AQA: Aspect-based Opinion Question Answering. In *2011 IEEE 11th International Conference on Data Mining Workshops (ICDMW)*, pages 89–96. IEEE.

[Panchenko et al., 2018] Panchenko, A., Ruppert, E., Faralli, S., Ponzetto, S. P., and Biemann, C. (2018). Building a Web-Scale Dependency-Parsed Corpus from CommonCrawl. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan*. European Language Resources Association.

[Park and Blake, 2012] Park, D. H. and Blake, C. (2012). Identifying Comparative Claim Sentences in Full-Text Scientific Articles. In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse, Jeju, Republic of Korea*, pages 1–9. Association for Computational Linguistics.

[Perera and Nand, 2015a] Perera, R. and Nand, P. (2015a). Answer Presentation with Contextual Information: A Case Study using Syntactic and Semantic Models. In *Australasian Joint Conference on Artificial Intelligence*, pages 476–483. Springer, Association for Computational Linguistics.

[Perera and Nand, 2015b] Perera, R. and Nand, P. (2015b). Selecting Contextual Peripheral Information for Answer Presentation: The Need for Pragmatic Models. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation: Posters, Shanghai, China*, pages 197–205.

[Rauscher et al., 2013] Rauscher, J., Swiezinski, L., Riedl, M., and Biemann, C. (2013). Exploring Cities in Crime: Significant Concordance and Co-occurrence in Quantitative Literary Analysis. In *Proceedings of the Workshop on Computational Linguistics for Literature, Atlanta, Georgia*, pages 61–71. Association for Computational Linguistics.

[Riehmann et al., 2012] Riehmann, P., Opolka, J., and Froehlich, B. (2012). The Product Explorer: Decision Making with Ease. In *Proceedings of the International Working Conference on Advanced Visual Interfaces, Capri Island, Italy*, pages 423–432. ACM.

[Shapiro and Wilk, 1965] Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611.

[Shneiderman, 2010] Shneiderman, B. (2010). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Pearson Education India.

[Stab et al., 2018] Stab, C., Daxenberger, J., Stahlhut, C., Miller, T., Schiller, B., Tauchmann, C., Eger, S., and Gurevych, I. (2018). ArgumenText: Searching for Arguments in Heterogeneous Sources. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, New Orleans, Louisiana*, pages 21–25. Association for Computational Linguistics.

[Sun et al., 2006] Sun, J.-T., Wang, X., Shen, D., Zeng, H.-J., and Chen, Z. (2006). CWS: A Comparative Web Search System. In *Proceedings of the 15th international conference on World Wide Web, Edinburgh, Scotland*, pages 467–476. ACM.

[Wachsmuth et al., 2017] Wachsmuth, H., Potthast, M., Al Khatib, K., Ajjour, Y., Puschmann, J., Qu, J., Dorsch, J., Morari, V., Bevendorff, J., and Stein, B. (2017). Building an Argument Search Engine for the Web. In *Proceedings of the 4th Workshop on Argument Mining, Copenhagen, Denmark*, pages 49–59. Association for Computational Linguistics.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.
Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den