

Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Medieninformatik

Crowdsourcing a Corpus for Clickbait Spoiling

Bachelor's Thesis

Jana Puschmann
Born Nov. 28, 1996 in Flensburg

Matriculation Number 115753

1. Referee: Prof. Dr. Benno Stein
2. Referee: PD Dr. Andreas Jakoby

Submission date: June 17, 2019

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, June 17, 2019

.....
Jana Puschmann

Abstract

Clickbait in online news is a growing problem. Young people, in particular, are consuming their news via social media, and the trend is rising. To fight this problem, the concept of automated clickbait spoiling is introduced. In the context of this thesis, the Webis-Clickbait-19 Corpus is constructed, which contains 3,042 entries. Each entry consists of a clickbait post, a related article and a spoiler. The spoilers are sentences that have been extracted from the articles by human annotators on Amazon Mechanical Turk. With this data, three simple clickbait spoiling approaches are developed and evaluated. Each approach ranks the sentences of an article from highest to lowest based on a maximum of two features.

Contents

1	Introduction	1
2	Background and Related Work	5
3	Corpus Construction	9
3.1	Preprocessing	9
3.2	Crowdsourcing	10
3.2.1	Clickbait Spoiling HIT	11
3.2.2	Annotation Review	13
3.3	Postprocessing	15
3.3.1	Corpus Structure	17
4	Clickbait Spoiling	19
4.1	Measures	19
4.2	Naive Ranking	20
4.3	Clickbait-Sentence-Similarity	20
4.3.1	TF-IDF	21
4.3.2	Cosine Similarity	22
4.4	Logistic Regression Model	23
4.5	Evaluation	26
5	Future Work and Conclusion	29
5.1	Webis Clickbait Corpus 2019	29
5.2	Clickbait Spoiling	30
A	Source Code and Data	32
	Bibliography	33

Acknowledgements

Foremost, I would like to thank my supervisors Tim Gollub, Junior-Prof. Dr. Martin Potthast and Prof. Dr. Matthias Hagen for their ongoing support during the creation of this thesis. I also wish to thank Prof. Dr. Benno Stein and PD Dr. Andreas Jakoby for examining my work.

A special thanks to Kristof Komlossy for helping me with the Mturk Manager and answering any questions in that regard.

Many thanks also go to Jonas Dorsch, Fabienne Hubricht, Jiani Qu and my father for proofreading this work and for the suggestions for improvement. Last but not least, I would like to express my endless gratitude to my family, who supported me throughout my studies.

Chapter 1

Introduction

This thesis deals with the construction of the Webis Clickbait Corpus 2019 and analyzes some characteristics in preparation of an automated clickbait spoiling process.

As digitalization progresses, traditional media such as newspapers, magazines, and tabloids have to venture into the online world. The entry into online journalism offers a lot of advantages, like faster delivery, less to no printing costs, and easy nationwide reporting. Nevertheless, it also makes the entry for fresh news publishers easier. Today, consumers are able to choose between a variety of different news websites, however, this means that the competition between news publishers is even bigger than at the time of print media.

Other than traditional print media, online news are usually not sold to the consumer, but are financed through advertising revenues. The *pay-per-click model* (ppc) allows publishers to be paid per visit to their website (Fearn [2017]). Since the competition is strong, publishers have to find ways to attract attention in order to make profit.

This has led to a recent phenomenon that we call *clickbait*. Even though there is no single way to define clickbait, it generally describes social media posts designed to entice a reader to click on a related link, at the expense of informativeness and objectiveness (Potthast et al. [2018a]). Usually this is done by creating curiosity in the reader. Figure 1.1 shows that in many cases, clickbait will exaggerate (1, 2, 3, 4), cut out important information (1, 2, 3, 4, 5), or mislead the reader (5). A typical clickbait post on Twitter may look like the one shown in Figure 1.2¹. Even though readers might not be particularly interested in Anne Hathaway, they could assume, for example, that she is struggling with a drinking problem.

¹<https://twitter.com/HuffPost/status/1118702722793058305> [accessed May 13, 2019]

- (1) You Won't Believe What Emma Roberts Has Done Now
- (2) Prevent Your Kids From Playing Near This Dangerous Common Plant That Can Kill Them
- (3) Green Energy Surging and you'll never Guess Why
- (4) Skinny Jeans May Look Great But It Could Do This To Your Body. Shocking!
- (5) Melissa McCartney Says Her Goodbyes to The World!

Figure 1.1: Examples of clickbait from the new Webis Clickbait Corpus 2019.

To confirm or disprove their theory, they will then follow the link to the news portal, where they find out that Anne Hathaway simply does not like the length of her hangovers. This information can be found in the second sentence of the article. Since most readers are probably not interested in details, they will leave the website again immediately after.

According to a survey conducted by the Pew Research Center, the number of U.S. Americans that get their news from social media on a regular basis is rising. Especially among young people aged between 18 and 29, social media is the main source of news (Shearer [2018]). Even though these studies were conducted in the U.S. only, it is imaginable that this development will also apply in other countries with similar social media behavior. In another article by Matsa and Shearer, it was reported that 68% of Americans receive their news from social media at least occasionally, although 57% of the respondents also doubt their accuracy. While it seems that people like the convenience of news on social media, they are also afraid of clickbait, fake news and unreliable news sources (Matsa and Shearer [2018]). Additionally, it has been found that 59% of all shared URLs on Twitter are shared without actually following that URL to the related article (Gabelkov et al. [2016]). This leads to the conclusion that many users did not actually read an article at all before sharing it. As a result, many dubious articles with exaggerated or outrageous titles can easily spread on social media.

Due to strong competition, even some reputable news publishers resort to clickbait headlines to attract more readers. Although, the related article can be of good quality, readers may not be able to differentiate them from articles that are actually just baiting them for the ad revenue.

Avoiding clickbait makes it less lucrative to the publishers and forces them to find different ways to reach their audience. However, as Loewenstein describes it in his *information-gap theory*, the less information is needed to gain



Figure 1.2: A typical clickbait post on Twitter.

full knowledge on a topic, the more an individual will try to fill that gap (Loewenstein [1994]). As a result, we are still clicking, even though we know, that an article will most likely not live up to our expectations.

A counter-movement of social media users like Jake Beckman (@SavedYouAClick²) or Alex Mizrahi (@HuffPoSpoilers³) are actively spoiling clickbait articles on social media platforms. Figure 1.3⁴ shows a spoiler post to the tweet seen in Figure 1.2. However, this does not help a user who is confronted

²<https://twitter.com/savedyouaclick> [accessed May 19, 2019]

³<https://twitter.com/HuffPoSpoilers> [accessed May 19, 2019]

⁴<https://twitter.com/HuffPoSpoilers/status/1118703465398710273> [accessed May 19, 2019]



Figure 1.3: A spoiler to Figure 1.2

with several clickbait posts and his social media timeline on a daily basis.

Instead of taking the bait or manually spoiling clickbait articles, there is a need for automated clickbait spoiling. Spoiling a clickbait usually means to provide the information that is missing in the clickbait, so users only have to "click" if an article genuinely interests them. This thesis explores the possibility of extracting a spoiler sentence or sentences from an article that is referenced by a clickbait, rather than using complex natural language processes for answer generation. Chapter 2 presents research that has already been done in the fields of clickbait detection and clickbait spoiling. Subsequently, Chapter 3 discusses the creation of the Webis Clickbait Corpus 2019, which consists of 3,042 clickbait posts, the related articles, and human-annotated spoiler sentences extracted from those articles. Chapter 4 introduces three simple clickbait spoiling approaches, that rank sentences based on the positions of sentences and their similarity to a clickbait. Each ranking shows improvements in performance compared to the proposed baseline approach. In the end, an outlook on future work is given in Chapter 5.

Chapter 2

Background and Related Work

This Chapter discusses research that has already been done in the field of clickbait. It starts with an introduction of several clickbait detection approaches and practical applications that have resulted from it. Afterwards, a thesis in the field of clickbait spoiling is presented.

Up until now, there has already been a lot of work done in the field of clickbait detection. In 2016 Potthast et al. introduced the first machine learning approach to detect clickbait using *Random Forest* for classification and a first evaluation corpus of 2,992 tweets, of which 767 were clickbait (Potthast et al. [2016]).

Almost at the same time, Biyani et al. presented a clickbait detection approach using *Gradient Boosted Decision Trees* with a data set of 4,073 news headlines from different news publishers, of which 1,349 were clickbait (Biyani et al. [2016]).

Later that year Agrawal proposed a clickbait detection approach that was trained using a *Convolutional Neural Network*. Their corpus consisted of 2,388 headline samples from Twitter, reddit, and Facebook, where 814 of them were clickbait (Agrawal [2016]).

Chakraborty et al. created a corpus of 18,513 non-clickbait and 8,069 clickbait headlines. They selected 7,500 headlines from each category to train three different kinds of classification models: *Support Vector Machines* (SVM), *Decisions Trees*, and *Random Forests* (Chakraborty et al. [2016]). Anand et al. used the same corpus to train a *Recurrent Neural Network* (Anand et al. [2016]).

In 2017 Rony et al. collected a media corpus of 1.67 million Facebook posts for a large scale analysis by a set of mainstream and unreliable media. They trained a *neural network based on word-embeddings* on their own data and evaluated their approach by using the dataset of 32,000 headlines curated by Chakraborty et al. mentioned above.

That same year, the Webis group¹ started a continuously shared task to evaluate clickbait detectors known as the *Clickbait Challenge*², to build a common foundation to evaluate and compare the performance of clickbait detectors. To evaluate the clickbait detectors, the Webis Clickbait Corpus 2017, consisting of 38,517 human-annotated Twitter tweets, was constructed. In contrast to the corpora from the other publications, which only distinguish between clickbait and no clickbait, the tweets in this corpus were evaluated on a 4-point scale from not click baiting to heavy click baiting (Potthast et al. [2018b]). The paper related to the Clickbait Challenge evaluated all the approaches mentioned before. It shows that the performances in terms of F1-measure were reported at about 0.75 (Agrawal [2016], Biyani et al. [2016], Potthast et al. [2016]) and 0.95 (Anand et al. [2016], Chakraborty et al. [2016], Rony et al. [2017]). Additionally, a total of 13 approaches were also submitted to the Clickbait Challenge 2017, which achieved further improvements in detecting clickbait (Potthast et al. [2018a]).

Generally, it can be said, that we are able to detect clickbait effectively in a variety of ways. This led to a couple of browser extensions and other solutions based on some of those clickbait detection models.

While the browser extension *Downworthy*³ does not really "detect" clickbait, but rather works on a set of rules based on the syntax of clickbait, it is still worth mentioning. Instead of blocking clickbait content, it takes popular phrases known to be used in clickbait and turns the often exaggerated headlines into more "realistic" and satirical versions (Hooton [2014]). Figure 2.1⁴ shows an example of a headline altered by the Downworthy extension.

Chakraborty et al. introduced their browser extension *Stop Clickbait*⁵ for Chrome browsers. It gives users the opportunity to block certain types of clickbait. The application will learn from the decisions and block similar clickbait content in the future (Chakraborty et al. [2016]).

Rony et al. used their clickbait detector to build the *BaitBuster*⁶ extension to detect clickbait on Facebook. If a potential clickbait post is found on the users' timeline, the extension notifies the user and explains the decision. It also calculates the cosine similarity between the headline and the article content and extracts the summary from the article (Rony et al. [2017], Rony et al. [2018]).

¹<https://webis.de> [accessed June 5, 2019]

²<https://www.clickbait-challenge.org/> [accessed June 5, 2019]

³<https://downworthy.snipe.net> [accessed June 6, 2019]

⁴<https://theawesomer.com/downworthy/267348/> [accessed June 6, 2019]

⁵<https://chrome.google.com/webstore/detail/stop-clickbait/iffolpdcmehbghbamkgobjjdeejinma> [accessed June 6, 2019]

⁶<http://dear.cs.olemiss.edu/baitbuster> [accessed June 6, 2019]

20 Things Every Modern Office Should Have In 2014

Cubicles are out, high-tech napping pods are in.

Inane Listicle of 20 Things You've Already Seen Somewhere Else Every Modern Office Should Have In 2014

Cubicles are out, high-tech napping pods are in.

Figure 2.1: Comparison between a clickbait headline and the altered version by Downworthy.

In early 2019, Rubin et al. introduced their LiT.RL News Verification Browser, that rates news headlines on a 4-point scale similar to the scale introduced by Potthast et al. in 2018b and highlights them accordingly (Rubin et al. [2019]).

Clickbait detection gives us the opportunity to deal with clickbait headlines and articles in a lot of ways. As many of the existing extensions show, they can actively highlight the problem to the user or block it from the users' view entirely. A survey conducted on Facebook suggests, that in 80% of cases the respondents "preferred headlines that helped them decide if they wanted to read the full article before they had to click through" (El-Arini and Tang [2014]). Instead of keeping the users from taking the bait, the information to complete a headline could be provided, so that users can decide for themselves whether they want to continue reading or not.

As mentioned above, the BaitBuster extension already provides a summary extracted from the article to the user. It uses the automated text summarizer *TextRank* from the Gensim⁷ library for python. However, it is not guaranteed, that this summary contains all the information a user would need to complete the headline.

In his thesis Ter-Akopyan introduces a corpus consisting of clickbait-spoiler-pairs and a semi-automated pipeline for clickbait spoiling based on named entities. He distinguishes between *fact-based* and *complex / narrative clickbait*.

⁷<https://radimrehurek.com/gensim/> [accessed June 13, 2019]

- (1) A cup of coffee will cost you 8\$ in this city (*fact-based*)
- (2) Here's why summer in New York City smells so awful (*complex*)

Figure 2.2: Comparison of factual and complex / narrative clickbait.

Figure 2.2 shows two examples. While fact-based clickbait (1) asks for a specific entity, complex clickbait (2) relies on a discourse in a context.

The corpus created as part of his work consists of 5,787 clickbait-spoiler-pairs which originated from social media accounts that spoil clickbait on Facebook, Twitter, and reddit. Additionally, the related articles were saved as well.

The approach for clickbait spoiling proposed in his thesis focused on spoiling factual clickbait by using named-entity recognition. He determines the named-entity that is referenced by a cataphora in the clickbait. Afterwards, the text is analyzed for named-entities of that type and spoiler candidates are extracted and ranked. Unfortunately, his work only concentrates on named-entities that either depict a person or a location. Of the 5,787 clickbaits saved in his corpus, only 559 fit this profile. My thesis is supposed to bypass the problem by extracting a sentence from the text that serves as a spoiler. Thus, fact-based and complex clickbait do not have to be distinguished anymore.

Chapter 3

Corpus Construction

The following chapter explains the creation of the Webis Clickbait Corpus 2019. In the first section, the preprocessing of the data to create a new base corpus is stated. The second section discusses the crowdsourcing process in order to annotate that base corpus data. Finally, the third section elaborates on the creation of the annotated Webis Clickbait 2019 corpus.

3.1 Preprocessing

In order to create the Webis Clickbait Corpus, existing data from the *Webis Clickbait Corpus 2017*¹ and also from the *Webis Clickbait Corpus 2018*² was used.

Webis-Clickbait-17 consists of 38,517 annotated Twitter tweets that were sampled from the 27 most retweeted news publishers on Twitter (Potthast et al. [2018b]). The articles have been rated on a 4-point scale from "not click baiting" (0.0) to "heavily click baiting" (1.0) in a crowdsourcing campaign by 5 annotators each. By averaging the opinion of the annotators for each article the *truthMean* results. Since the task is to spoil clickbait posts, we chose only those articles for which the *truthMean* was higher than 0.8. This way only articles are considered that are clickbait. With this restriction, 1,845 articles have been adopted from Webis-Clickbait-17.

Additionally, all 5,787 articles from Webis-Clickbait-18 are used. The articles and spoilers of Webis-Clickbait-18 were crawled from the social media platforms Twitter, Facebook, and reddit where certain users are actively spoiling clickbaits by revealing the missing information of a clickbait. Other than the articles in Webis-Clickbait-17, these articles were only classified as clickbait

¹Webis-Clickbait-17

²Webis-Clickbait-18

Table 3.1: Distribution of social media platforms in the base corpus.

	Number of Articles			
	Twitter	reddit	Facebook	Σ
Count	3706	2427	1081	7214

by the user that posted the spoiler to the article.

This results in a base corpus of 7,632 articles. However, some of the articles were missing either the *postText*, i.e. the clickbait, or the article text and were therefore discarded. It would not make sense to spoil an article without the actual clickbait or vice versa. Finally, the base corpus consists of 7,214 clickbait articles and is saved in the JSON lines text file format³. Table 3.1 shows the distribution of articles over the three social media platforms.

3.2 Crowdsourcing

*Amazon Mechanical Turk*⁴ (Mturk, for short) is a crowdsourcing marketplace where *requesters* can outsource processes to a distributed user workforce, the so-called *workers*.

Requesters will post *Human Intelligence Tasks* (HITs) that are then *assigned* to workers. Each HIT can be assigned to multiple workers, which means that the same task will be annotated by more than one worker.

Since the goal is to automatically find a sentence or sentences in an article to spoil a related clickbait post, all of the 7,214 articles need to be annotated to gather as much data as possible. This task was outsourced to workers via MTurk.

The MTurk project was created using the *MTurk Manager*⁵. The Mturk Manager is a graphical user interface that simplifies tasks like creating, viewing, and annotating Amazon MTurk projects. Projects in the MTurk Manager consist of four different templates.

First of all, the requester has to create a *worker template*, which is the HTML view that is displayed to the worker. It contains the instructions, the task, and a submit button. Variables used by the requester are replaced by data from the *batch* that is uploaded as a CSV-file.

³<http://jsonlines.org> [accessed May 31, 2019]

⁴<https://www.mturk.com> [accessed April 19, 2019]

⁵<https://github.com/webis-de/mturk-manager> [accessed April 19, 2019]

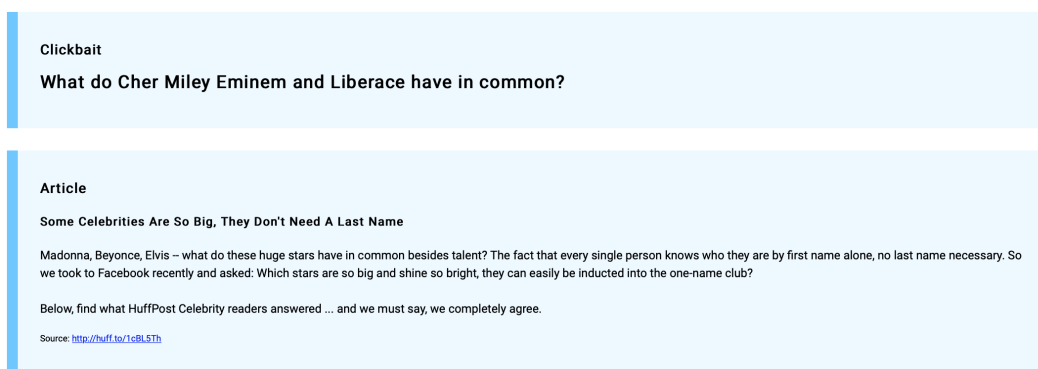


Figure 3.1: Worker view of a clickbait and its related article in a HIT.

Afterwards, an *assignment template* is necessary to process the input and output data. The requester chooses the relevant HIT attributes and creates a view that makes it easier to review the worker data.

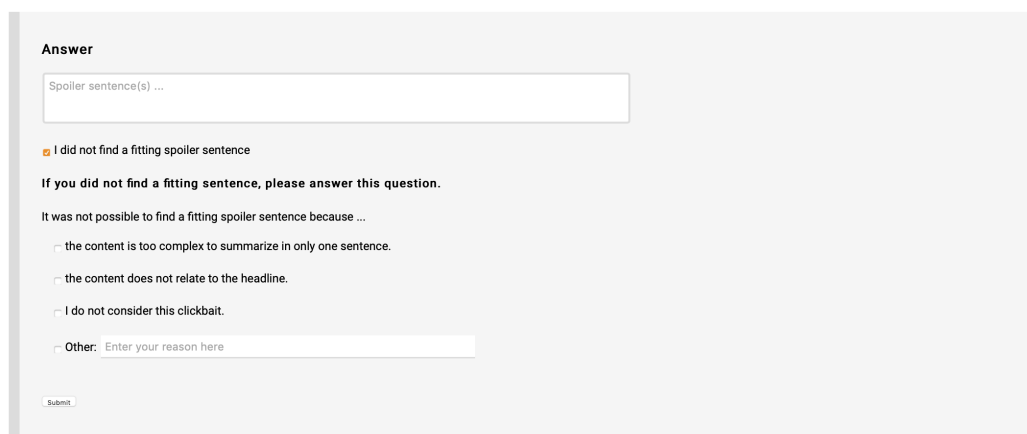
In addition, the *HIT template* prevents overhead if a HIT is assigned multiple times by grouping assignments to a HIT. This means that data from a HIT will only be shown once, while without it the HIT data would be shown for each of its assignments. For example, if a clickbait is annotated twice, the clickbait is shown only once, while each of the annotated spoilers is shown.

Finally, a requester can create a *global template*, which saves code snippets that are relevant for all templates, e.g., CSS styles.

After all templates have been created, the requester adds a *batch profile*. A batch profile saves important data like the title, rewards, the duration, the lifetime of the batch, the number of assignments per HIT, and worker qualifications. Additionally, the templates that are used in the batch are assigned. Then, the batch data is uploaded as a CSV file along with its batch profile so workers can accept the HITs.

3.2.1 Clickbait Spoiling HIT

The HIT asked workers to extract sentences from clickbait articles to spoil a related clickbait headline. To do this, the worker template was split into four different sections. The first section displayed the introduction and instructions to the task, the second section showed the clickbait post, while the third section presents the headline and text of the article. Figure 3.1 shows an example clickbait and its related article displayed in a HIT for a worker. In the last section, workers could either copy a sentence or sentences into the submission box, or decide that they did not find a spoiler in the article. If they did not find a spoiler, a set of checkboxes would appear to specify the reason why they



The screenshot shows a web interface for a HIT. At the top, there is a section titled "Answer" with a text input field containing the placeholder "Spoiler sentence(s) ...". Below this, there is a radio button option labeled "I did not find a fitting spoiler sentence". Underneath, a bold instruction reads: "If you did not find a fitting sentence, please answer this question." This is followed by a text prompt: "It was not possible to find a fitting spoiler sentence because ...". Below this prompt are three radio button options: "the content is too complex to summarize in only one sentence.", "the content does not relate to the headline.", and "I do not consider this clickbait.". There is also an "Other:" label followed by a text input field with the placeholder "Enter your reason here". At the bottom left of the form is a "Submit" button.

Figure 3.2: Annotation possibilities of a worker in a HIT.

thought the article could not be spoiled (see Figure 3.2). Afterwards, a worker submits the answer.

Each worker had 15 minutes to complete the task. Afterwards, the task would be returned and would be assigned to a different worker. The completion of an assignment by a worker was rewarded with 0.10\$ per assignment. Workers were allowed to work on 10 HITs each. After that, the limit had to be reset or unlocked in order for them to work on more. This procedure ensured that workers whose work did not meet our quality standards, would not be able to take HITs away from good workers.

To qualify for the HITs, workers had to have at least 95% approved assignments and at least 100 approved HITs. It was also decided to restrict the HIT to workers from specific locations. Only workers from the US, Great Britain, Canada and Australia were allowed to work on them. This had two reasons. On the one hand, workers from those locations are most likely native speakers, which makes it easier for them to understand and skim articles in the English language. On the other hand, the articles were mostly targeted towards a western audience, particularly North Americans.

In the beginning of gathering the data, it was decided to assign each HIT twice. However, after uploading the first two batches, it was noticed that two annotations per HIT would only increase the amount of data that had to be reviewed in the end. In many cases, the reviewer would have needed to decide between two possible spoiler candidates. That is why each HIT was only assigned once for the remaining batches.

The overall lifetime, i.e., the time the HIT would be accessible to workers, was set to two days. However, most batches were already fully processed after a few hours.

To split the base corpus into batches, the 7,214 articles in the base corpus were divided into 15 files. Unfortunately, the script that was used to randomize the order of the articles, assumed each file to be 500 and the last one to be 214 lines long. However, the table head also counts as a line, which is why 15 articles are missing from the further process. This was noticed only after the first two batches were uploaded. Recreating the CSV files would have caused a new randomized order of the articles in a batch. Besides, filtering those articles that were already annotated would have been too much effort for 15 articles. So, in the end, only 7,199 articles were annotated through MTurk.

While adding the batches to the MTurk Manager, it was noticed that emojis and control characters were not allowed with Amazon MTurk. Luckily, a workaround exists to include emojis in MTurk projects written by *charman* on GitHub⁶, so there was no need to filter out any emojis. The workaround replaced the emojis by javascript snippets in the batch files. Later, they were changed back to unicode characters in the worker template. Since the control characters are invisible to the reader and removing them did not change the actual content of the article, they were not filtered and not included in the batches.

3.2.2 Annotation Review

Reviewing the data turned out to be a much harder task than expected. Of the 7,199 articles, the HITs of the first and the fifteenth batch⁷ were annotated twice. This means that there were 499 additional annotations for the first batch and 213 additional annotations for the last one. Overall, 7,911 annotations were made by the workers.

As seen in Table 3.2, of 7,911 annotations in total, 4,298 (54.32%) were approved⁸ and 3,613 (45.67%) were rejected⁹.

A spoiler was found for 5,864 assignments. Of those 5,864 assignments, 3,477 spoiler annotations were approved and 2,387 annotations were rejected. If spoiler sentences were rejected, it was mostly because workers tried to find a sentence that summarized the important information rather than a sentence that actually completed the headline. 821 of the assignments were approved, because they correctly classified an article without a spoiler. The content of articles without a spoiler sentence usually did not relate to the headline, e.g.

⁶<https://github.com/charman/mturk-emoji> [accessed April 21, 2019]

⁷The fifteenth batch was actually annotated first, because it created the least amount of HITs.

⁸"Approved" means all annotations that have been approved by the reviewer. No annotations have been "internally approved".

⁹"Rejected" means all annotations that have either been "internally rejected" or "rejected" by the reviewer.

Table 3.2: Review of the 7911 assignments issued via MTurk.

Worker assignment review			
	Approved	Rejected	Σ
Spoiler found	3477	2387	5864
No spoiler found	821	1204	2025
Empty submissions	0	22	22
Σ	4298	3613	7911

because the actual article text was missing or the article turned out to be an ad instead. In some cases the content was overly complex or the article was a listicle¹⁰, where workers would have needed to copy sentences from different sections of the article. On the contrary, 1,204 assignments in which no spoiler was found were rejected, because the reviewer believed that an article most likely did contain a spoiler. Unfortunately, 22 assignments had to be rejected, because the workers did not give any information. They did not copy a spoiler sentence, nor did they say there was no spoiler.

Generally, it was noticed that this task seemed to be more difficult to some workers. Of course there are articles that are more complex than others, but it also appears that some workers had a better understanding for clickbait than others. Many of the rejected workers did not seem to understand what part of the clickbait actually needed to be spoiled. Figure 3.3 shows an annotation a worker has done on a clickbait (1). While the annotated sentence (2) does say that "a different answer is beginning to emerge", a reader cannot be really sure whether this answer is for or against the claim that was made in the clickbait. A better answer sentence from the article is shown in sentence (3).

Nowadays, lurid headlines are part of our everyday life, some people may not even realize that certain articles are baiting them. This is also why the review process had to be changed after the first batch. Instead of actually rejecting the workers, they were only "rejected internally", that means they were still rewarded, but their data was rejected. It's also important to mention that workers on MTurk are used to work on fast and simple tasks. Most of them worked on the task in less than a minute which was not enough time to read through some of the articles in the HITs.

¹⁰A "listicle" describes an article that follows a the structure of a list, e.g., "The 25 most expensive ZIP codes in America"

- (1) Does Marijuana Make You Stupid? (*Clickbait*)
- (2) But now a different answer is beginning to emerge, thanks to an authoritative new study led by Robert Tait at the Australian National University. The scientists looked at the long-term cognitive effects of marijuana use in nearly 2,000 subjects between the ages of 20 and 24. (*Worker annotation*)
- (3) In other words, the amount of pot consumed had no measurable impact on cognitive performance. (*Possible spoiler*)

Figure 3.3: Rejected example annotation from the crowdsourcing task.

3.3 Postprocessing

After all batches were reviewed, the gathered data was used to create an annotated version of the Webis-Clickbait-19. This corpus only consists of articles for which a spoiler has been found and approved. Other annotations were left out.

Even though the instructions clearly stated that full sentences should be copied, many workers only submitted sentence fragments. Therefore, the complete matching sentence in the article had to be found. To do this, a string containing the article title and the article paragraphs was constructed. Then, both the given spoiler and the article string were split into lists of sentences with the help of the *Natural Language Toolkit*¹¹, a python library. Afterwards the sentences from the spoiler were each compared to the sentences from the article. If an exact match was found, the spoiler sentence given by the worker was replaced by the actual article sentence. This way we also made sure that the annotated sentence actually appears in the article. Nevertheless, some challenges were faced while the annotation results were processed.

Blacklisting assignments

Sometimes annotated spoiler sentences could not be matched to a sentence in the article. After closer examination, it was found that a few workers used the given source of the article, which was referenced in the HIT, to spoil the clickbait. However, some articles in the corpus were missing the proper article content. In other cases, only part of the content was saved because the source content was split into several subpages.

All articles for which the spoiler match did not work, were reviewed again.

¹¹<https://www.nltk.org> [accessed April 20, 2019]

If the article content was insufficient, the *assignment_id* was added to a black-list, so the annotation would not be taken into account, even though it was approved. Should a similar annotation process be repeated, it could be specified that the spoiler should be copied from the HIT and not from the source material. Alternatively, if the sources are still available, the content could be downloaded in the future.

Multiple assigned HITs

As mentioned before, the HITs of two batches were assigned twice. Unless both workers chose the same spoiler, the HIT had to be re-annotated manually by the reviewer who chose the most fitting spoiler of the two annotations. If it was decided that the spoiler of the second annotation was the better match, the corresponding *assignment_id* was saved in a list. As the articles were written to the corpus, it was checked whether the same entry with a different spoiler already existed. The spoiler was overwritten if the entry already existed and if the *assignment_id* of the second annotation was saved in the list.

Non-consecutive spoiler sentences

It was decided that the data should only contain consecutive spoiler sentences. That means that at least the listicle spoilers which were approved in the beginning had to be filtered again. Additionally, discovering non-consecutive sentences while reviewing the data was not a simple task. In order to filter those sentences, the spoiler positions were checked before actually writing them to a file. If the first character of the second sentence did not follow the last character of the first sentence, the entry was discarded. The same applied to the following sentences in a spoiler.

Re-annotation of modified sentences

Even if the content of an article was complete, some spoiler sentences still could not be matched to a sentence in that article. After reviewing those annotations again, it was noticed, that some sentences were altered. Most of those sentences differed from their counterparts because a worker added a full stop or a quotation mark, even though originally there would not have been one. Punctuation was mostly added in the end of a sentence, either to create symmetry or to indicate the end of the spoiler. Fortunately, this meant that this problem was easy to fix. The punctuation in the end of a spoiler was removed while it was matched to the sentences from the article.

Some additional modifications were also found in spoiler sentences. For instance one worker left out the sub-ordinate clause in the middle of a spoiler

Table 3.3: Distribution of social media platforms in the annotated corpus.

	Number of Articles			
	Twitter	reddit	Facebook	Σ
Count	1428	1085	529	3042

sentence. Assignments that were not easily fixed by adding a small improvement to the processing script, were re-annotated manually.

In the post-processing step it was also noticed, that some spoiler sentences were not part of the article at all. They were most likely summary sentences written by the workers themselves. The *assignment_ids* of those articles were also added to the blacklist mentioned before.

Finally, the Webis Clickbait Corpus 2019¹² consists of 3,042 articles and their annotated spoilers. Of those 3,042 entries, 367 were adopted from Webis-Clickbait-17. The remaining 2,675 articles were originally part of Webis-Clickbait-18. Table 3.3 shows the distribution of the social media platforms the entries originated from. Of all spoilers in Webis-Clickbait-19, 80.80% consist of only one sentence. An additional 14.07% of all spoilers consists of two sentences. Even though this means that nearly 95% of all annotated spoilers consist of a maximum of two sentences, a spoiler sentence in the corpus may contain up to seven sentences in total.

3.3.1 Corpus Structure

The overall structure of the corpus is oriented towards Webis-Clickbait-17 and is shown in Listing 1. Additionally, the *article_imgs*, *article_url* and *cb_spoiler* keys were adopted from Webis-Clickbait-18. These keys were later renamed to *targetMedia*, *targetUrl* and *humanSpoiler* to adjust them to the new naming.

Each article has been given a unique *uuid* and the corpus from which the article originated from has been saved in the *originalCorpus* key.

Keys that include the word *post* relate to the clickbait that was posted on a social media platform. In this case the *postText* describes the clickbait text that is to be spoiled. On the contrary, a key that includes the word *target* describes attributes of the linked article. The article content is stored in a list of paragraphs as *targetParagraphs*.

¹²Webis-Clickbait-19

```
{
  "uuid": "8ceed760-2fdb-49ec-b8a9-cd08ddd83426",
  "originalCorpus": "webis-clickbait-17",
  "postId": "810551255659585536",
  "postPlatform": "Twitter",
  "postMedia": ["media/photo_810551253520551936.jpg"],
  "postText": ["Washington Capitals' \u2018Social Night ..."],
  "targetTitle": "Washington Capitals' \u2018Social Night ...",
  "targetDescription": "The Washington Capitals messed up ...",
  "targetKeywords": "",
  "targetMedia": null,
  "targetParagraphs": ["A very smart woman once advised ..."],
  "targetUrl": null,
  "humanSpoiler": null,
  "spoilerSentences": ["An interaction with one fan got ..."],
  "spoilerSentencePositions": [[2, 0, 150], [2, 151, 240]]
}
```

Listing 1: Example excerpt of an entry from Webis-Clickbait-19.

The *humanSpoiler* is an exclusive attribute of articles from Webis-Clickbait-18. It stores the spoiler that was posted by a social media user on either Twitter, reddit, or Facebook and was adopted in the new corpus to help with the annotation. Since the reviewer could not skim every article, the *humanSpoiler* was a good indication whether the annotated sentence was a spoiler or not.

As mentioned before, the spoiler was cut into sentences, which were then stored as *spoilerSentences* in a list. For each sentence, the paragraph in which it can be found and the position in that paragraph was determined. The position can be found as *spoilerSentencePositions*, where the index of the position corresponds to the index of the sentence in *spoilerSentences*. In the example shown in Listing 1, a spoiler position can be interpreted as follows: the first index describes the *paragraph index*, the second index represents the *starting character* of the sentence in that paragraph, similarly, the third index represents the *end character* of the sentence. A paragraph index equal to -1 means that the sentence can be found in the *targetTitle*. It is worth mentioning that the paragraph index starts counting from 0. In the example, the paragraph index 2 therefore refers to the third paragraph in the list.

Chapter 4

Clickbait Spoiling

This Chapter introduces three clickbait spoiling approaches and compares their results in Section 4.5. It starts by setting a baseline. Afterwards, a ranking on the basis of the similarity between sentences and clickbait is explained. Lastly, a logistic regression classifier based on sentence similarity and position is presented.

4.1 Measures

This section briefly explains the measures that are used to evaluate the introduced clickbait spoiling approaches in the course of this chapter. The goal of the approaches is to rank the sentences of an article in the corpus based on different features. A sentence in the first position of a ranking is considered the highest ranked sentence.

Average Rank The *average rank* describes the average position of an annotated spoiler in a ranking. To calculate the average rank, each sentence in a ranking is matched to the determined human-annotated spoiler sp , the ground truth, for every article d in a corpus D . Afterwards, the sum of the ranks in the corpus is calculated and divided by the overall number of articles in the corpus $|D|$. As all of the introduced approaches only rank single sentences, the rank of a spoiler consisting of several sentences is determined by finding the first match with one of the spoiler sentences.

$$average_rank = \frac{\sum_{d=1}^{|D|} rank_{sp,d}}{|D|}$$

Precision@n To get an overall idea of the performance of an approach, the number of annotated spoiler sentences that are ranked in the range between

Table 4.1: Distribution of annotated spoilers in the first ten ranks of the Naive Ranking approach.

	Precision@n				
Precision@	1	2	3	4	5
Spoiler count in %	6.28	22.22	35.04	45.30	53.52
Precision@	6	7	8	9	10
Spoiler count in %	60.82	67.19	72.42	76.92	80.60

rank 1 and n is determined.

$$precision@n = \sum_{i=1}^n |\{sp \in D : rank(i)\}|.$$

4.2 Naive Ranking

In this section, the *Naive Ranking* approach is introduced as a baseline technique. The approach takes the natural ranking, i.e., the order in which the sentences appear in an article, and assumes the first sentence to also be the highest ranked sentence. Since a clickbait cannot spoil itself, sentences that are exact duplicates of a clickbait are not considered as a spoiler candidate in the ranking. As a result, the second sentence is the highest ranked spoiler if a title repeats the clickbait. Table 4.1 shows that about 6.3% of the highest ranked sentences are matching the human-annotated spoilers. About 53.5% of all human-annotated spoilers are found in the first five ranks. 80.6% of spoilers from the corpus can be found in the first ten ranks. The average annotated spoiler is ranked in the 7.73-th rank. Since an average rank of 1.0 would mean that all human-annotated spoilers were also highest ranked by an approach, the goal of the following approaches is to reduce the average rank to 1.

4.3 Clickbait-Sentence-Similarity

As a first feature, the similarity between a sentence and its related clickbait is evaluated. The assumption is, that sentences which do not share any similarities with the clickbait are most likely not spoiling it. On the contrary, sentences that do share many similarities with a clickbait may have a higher probability

of spoiling that clickbait. This approach creates a ranking of sentences based on the cosine-similarity between that sentence and the clickbait.

4.3.1 TF-IDF

Before the similarity between two sentences can be calculated, the sentences have to be converted into a vector representation. This is done by using the *term frequency-inverse document frequency* measure (TF-IDF).

TF To determine the *term frequency*, the number of occurrences of a term t in a document d is counted. Common English stop words that do not increase the information content, are filtered before hand.

$$tf_{t,d} = |\{t \in d\}|$$

It should be mentioned that the order of terms in a sentence is not relevant in this approach. Therefore, two equal word vectors do not necessarily describe the same sentence (Manning et al. [2008]).

IDF Since all terms in the raw term frequency are equally important, the *inverse document frequency* is a measure for the information content a term provides in the context of a corpus. Terms that are rarer in the context of a corpus may be more important to a sentence than those which occur more often. The *document frequency* is defined as the number of documents $d \in D$ that contain a term t

$$df_{t,D} = |\{d \in D : t \in d\}|.$$

The inverse document frequency is then calculated as follows

$$idf_{t,D} = \log \frac{|D|}{df_{t,d}}.$$

This results in a high idf weight if the term is rare, and a low idf weight if the term is rather common. Finally, the TF-IDF can be computed by multiplying the tf and idf values (Manning et al. [2008]).

$$tf.idf_{t,d,D} = tf_{t,d} \cdot idf_{t,D}$$

Instead of calculating the inverse document frequencies of terms in the corpus, they were calculated from all of English Wikipedia¹ to ensure a natural distribution of terms. The data that was used for this process was downloaded from Wikimedia Dump². The resulting word vectors can be used to calculate the corresponding similarity between those vectors.

¹<https://en.wikipedia.org/> [accessed June 1, 2019]

²<https://dumps.wikimedia.org/enwiki/> [accessed June 1, 2019]

Table 4.2: Distribution of annotated spoilers in the first ten ranks of the Cosine Similarity Ranking approach.

	Precision@n				
Precision@	1	2	3	4	5
Spoiler count in %	12.89	27.94	40.04	49.28	58.71
Precision@	6	7	8	9	10
Spoiler count in %	64.50	70.45	75.12	78.96	81.95

4.3.2 Cosine Similarity

A popular way to measure the similarity of two sentences is to calculate the *cosine similarity*, which is calculated as the normalized dot-product between two sentences \vec{s}_1 and \vec{s}_2 in vector space (Sidorov et al. [2014]).

$$\cos(\theta) = \frac{\vec{s}_1 \cdot \vec{s}_2}{\|\vec{s}_1\| \|\vec{s}_2\|}.$$

The resulting cosine can take values between 0 and 1, where a cosine of 0 means that the two sentences are completely independent, while a cosine of 1 indicates a high similarity in content.

To see how the cosine similarity performed as the only feature, the cosine similarity between a clickbait and the sentences from the related article are computed. The sentence with the highest cosine similarity is also ranked highest by the approach. If multiple sentences have the same cosine similarity, the sentence that appears first in the article is chosen. As mentioned in Section 4.2, sentences that are copies of the clickbait are not considered as spoiler candidates.

To calculate the word vectors and the cosine similarity, the machine learning library scikit-learn³ was used.

Table 4.2 shows that about 12.9% of the highest ranked sentences match the annotated spoilers. Among the first five ranked sentences, 58.7% of spoilers are found. Finally, 81.95% of spoilers are also predicted by the approach within the first ten ranks. The average annotated spoiler is found in the 7.06-th rank. While the average cosine similarity of the annotated spoilers is only 0.167, the average cosine similarity of the highest ranked sentences by this approach is 0.249.

³<https://scikit-learn.org/> [accessed June 2, 2019]

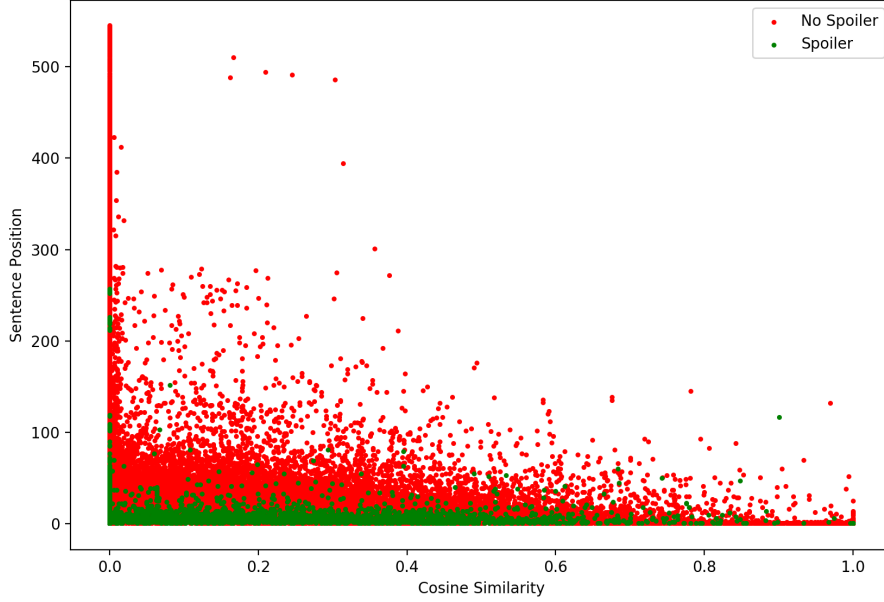


Figure 4.1: Graphical recording of all sentences in Webis-Clickbait-19 according to sentence position and cosine similarity. Red points represent sentences that do not spoil an article. Green points represents sentences that do spoil an article.

4.4 Logistic Regression Model

To increase the classification performance, the position of a sentence in the text is added as a feature. The idea is that, as Table 4.1 suggests, most spoiler sentences can be found in the beginning of an article already. For comparison, Table 4.3 shows that the average length of an article is about 28 sentences. Since outliers can have a great influence on the average, the median and mode are also displayed, as well as the least and highest amount of sentences in the articles from Webis-Clickbait-19. Additionally, Figure 4.1 shows all sentences from Webis-Clickbait-19 by their position and their cosine similarity to their related clickbait. Sentences that are annotated spoilers, are drawn in green (4,028 points), while the sentences that are not, are drawn in red (80,809 points). Even though the amount of points is massive, we can still gather some information from the visualization. While it cannot be said that a sentence is a spoiler, it might be possible to tell which sentences are not spoilers. The assumption is that the higher the position of a sentence, the less likely it will be a spoiler. Even though, this assumption should still be taken with a grain of salt, since the average length of an article is only about 28

Table 4.3: *Mean, Median and Mode* values of the number of sentences of an article from Webis-Clickbait-19. *Min* displays the least amount of sentences in an article, *Max* displays the highest amount of sentences in an article.

Number of Sentenes in an Article				
Mean	Median	Mode	Min	Max
27.9	20.0	13.0	2	545

sentences, but the visualization shows sentence positions up to 500.

To construct a classifier that discriminates between spoiling sentences and non spoiling sentences, a *logistic regression* is used. A logistic regression is a supervised learning method, that is trained on set of input-output-observations and calculates a *sigmoid function* to estimate an output for a set of test inputs. In this case the logistic regression gets two input observations $x^{(p)}$, the sentence position, and $x^{(c)}$, the cosine similarity. Additionally, the related output y is given, where $y = 1$ means the sentence is a spoiler, and $y = 0$ means the sentence is not a spoiler.

The logistic regression starts by learning a vector of *weights* and a *bias term* from a training set. Afterwards, it calculates the weighted sum of evidence z by calculating the dot product between the weight vector and the feature vector and adds the bias term.

$$z = w \cdot x + b$$

This is passed to the sigmoid function $\sigma(z)$, which maps the input into the range $[0, 1]$.

$$y = \sigma(z) = \frac{1}{1 + e^{-z}}$$

Finally, a test instance x can be classified by calculating the probability $P(y = 1|x)$. The decision boundary of a logistic regression is 0.5 (Jurafsky and Martin [2018]).

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

To calculate the logistic regression in Python the scikit-learn library was used. Since there are only 4,028 spoiler sentences and 80,809 sentences that do not spoil, the weights were adjusted inversely proportional to class frequencies in order to compensate for the disproportion in classes. The training process takes all 84,837 observations as input and produces the weights -0.0492 for

Table 4.4: Confusion matrix produced by the Logistic Regression that used all data points for both training and testing.

Confusion Matrix		
	actual positive	actual negative
predicted positive	3406	34163
predicted negative	622	46646

the sentence rank and 0.9262 for the cosine similarity feature vectors. The bias term is 0.633. Afterwards, \hat{y} is predicted on the same set of features to see how good the classification can get.

The confusion matrix is displayed in Table 4.4. It shows the *True Positives* in the upper left cell, the *False Positives* in the upper right cell, the *False Negatives* in the lower left cell and the *True Negatives* in the lower right cell. These values can be used to calculate the *precision*, *recall* and *f-measure*, which are displayed in the classification report in Table 4.5.

Precision "Precision measures the percentage of the items that the system detected that are in fact positive" (Jurafsky and Martin [2018]).

$$P = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall "Recall measures the percentage of items actually present in the input that were correctly identified by the system" (Jurafsky and Martin [2018]).

$$R = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1-measure The F-measure incorporates aspects of both precision and recall. Both values are balanced equally in the calculation (Jurafsky and Martin [2018]).

$$F_1 = \frac{2PR}{P + R}$$

The system does predict 85% of all spoiler sentences right. It should be taken into account that it also predicts 37,567 spoilers in total, even though it was only trained with 4,028 spoiler sentences. In order to allow a better comparison with the previous approaches, the sentences are ranked by their

Table 4.5: The classification report produced by the Logistic Regression Model.

Classification Report			
	Precision	Recall	f1-score
Spoiler	0.09	0.85	0.16
No Spoiler	0.99	0.58	0.73

Table 4.6: Distribution of annotated spoilers in the first ten ranks of the Logistic Regression Model approach.

Precision@n					
Precision@	1	2	3	4	5
Spoiler count in %	13.91	32.58	46.25	55.06	62.46
Precision@	6	7	8	9	10
Spoiler count in %	68.61	73.93	78.11	81.79	84.29

probability of being a spoiler. The decision boundary will not count for the selection of the spoiler candidates.

On average, the probability of the highest ranked sentences is 60.73% and their cosine similarity is around 0.198. Approximately 13.9% of the annotated spoilers are also ranked highest by the logistic regression model. 62.5% of the annotated spoilers are found within the first five ranks and 84.3% within the first ten ranks. The average rank is 6.71.

4.5 Evaluation

In order to provide a stronger basis for the evaluation, the precision values of a *random ranking* were calculated in addition to the baseline. On average, an annotated spoiler can be found in the 12.99-th rank in the random ranking. Figure 4.2 shows the Precision@n values that have been computed in the previous sections. It shows that the random ranking does perform better in the highest rank than the naive ranking approach. This is probably due to short articles that only consist of very few sentences, which means the probability of ranking the right sentence at random is quite high. This is why the random

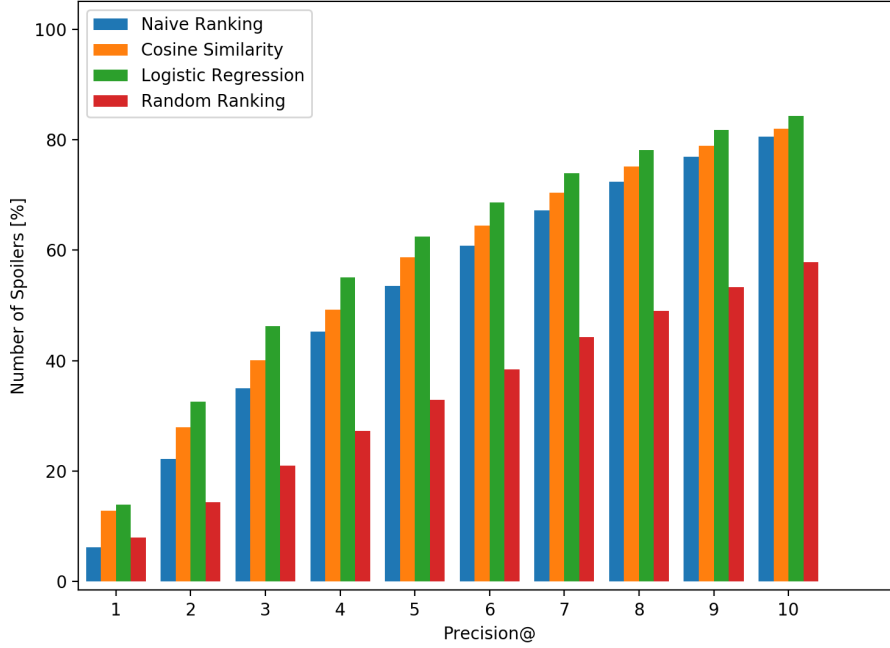


Figure 4.2: Comparison of the precision@n values of the introduced approaches in this chapter.

ranking performs much worse than the other approaches in the following ranks. As a result, all introduced spoiling approaches create a better ranking than a random ranking does.

Moreover, the figure shows that the logistic regression approach performs the best predictions for every rank of the ten precision values, and that the cosine similarity approach still beats the naive ranking approach. These results are also supported by the average ranks (see Table 4.7). At 6.71, the logistic regression is on average one rank better than the naive ranking, which has an average rank of 7.73. The clickbait spoiling approaches proposed by this thesis are still very simple and, as the precision@1 values of each approach show, the annotated spoilers are only predicted in very few cases. However, the performance shown by the precision@10 values show promising results compared to the performance of a random ranking. Each of the introduced approaches ranks at least 80% of the spoilers sentences within the first ten ranks, while only 58% of spoilers can be found within the first ten ranks of the random ranking. One should keep in mind that these approaches use a maximum of two features, and that additional features may improve the clickbait spoiling

Table 4.7: Comparison of the average ranks of the different approaches.

Average Rank	
Approach	Average Rank
Naive Ranking	7.73
Cosine Similarity	7.06
Logistic Regression	6.71
Random Ranking	12.99

process further. Section 5.2 proposes possible ways to continue this work in the field of clickbait spoiling.

Chapter 5

Future Work and Conclusion

This chapter discusses possibilities to pursue this work in the future. First, possible improvements to Webis-Clickbait-19 are explained. Afterwards, an outlook is given on how to continue the work in the field of clickbait spoiling.

5.1 Webis Clickbait Corpus 2019

Since the corpus was merged from two data sources, there are still some inconsistencies to resolve. Entries that originated from Webis-Clickbait-17 are missing data in the *targetMedia* and the *targetUrl* keys. These keys were adopted from the Webis-Clickbait-18 corpus. Additionally, the *targetMedia* that does exist in the corpus is only referenced by hyperlinks and is not stored locally, yet. To preserve the data, these images should be downloaded in the future.

Entries which originated from Webis-Clickbait-18 are missing the *postMedia* key. This is mostly due to the fact that their *postIds* do not reference the actual clickbait post like entries from Webis-Clickbait-17, but the post containing the *humanSpoiler*. Unfortunately, it may not be able to link posts from Facebook or reddit to original social media posts. Mainly because they do not exist in many cases and the clickbait has been found in an ad for example. Some of the spoilers saved from Twitter were retweets, which makes it very simple to find the actual social media posts. However, after a little bit of investigation it turned out that many of the posts on Twitter were actually not retweeted.

While the data of Webis-Clickbait-18 has been cleaned during the merging of the corpora, there are still some fragments in the *targetParagraphs* left. These fragments include advertisements that interrupt the text flow, verbose information about sharing the article on social media, and sometimes sentence duplicates. All these things do not contribute to the actual article content and should be removed so that they are not considered in a possible machine

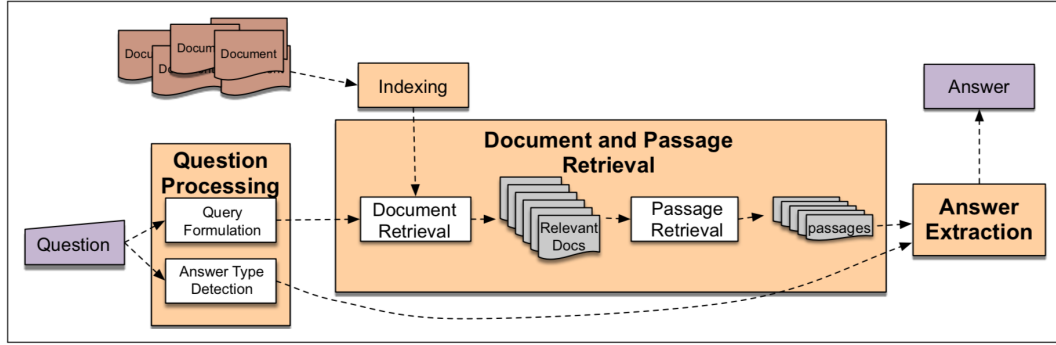


Figure 5.1: Phases of an Open Domain Question Answering process (Jurafsky and Martin [2018]).

learning process.

5.2 Clickbait Spoiling

This work has only started to explore a small amount of features that might be used in a machine learning approach to automate clickbait spoiling.

The cosine similarity in Section 4.1.2 just measures exact similarities. This means that synonyms or words that are related in meaning are not considered in the scoring of the similarity. For example, words like "game" and "play" could be rated more similar than "work" and "play". Sidorov et al. proposed a *soft cosine measure*, which has the potential to improve the performance of cosine similarity as a feature in natural language (Sidorov et al. [2014]).

The work that has already been done by Ter-Akopyan in 2017 could also be used as a possible extension to this approach. Instead of ranking all sentences, a possible spoiler type could be determined. Afterwards, only those sentences that contain entities of that spoiler type could be taken into consideration.

Before working out completely new structures for clickbait spoiling, it might be worth taking a look at another large field of research that could also be linked to the problem of clickbait spoiling. *Open Domain Question Answering* takes natural language questions and turns them into queries to retrieve a potentially relevant text snippet from a set of documents (Harabagiu et al. [2003]). The application is quite similar in a sense that a clickbait can be seen as a question to which an answer, the spoiler, is sought. Figure 5.1 shows the three stages of a question answering process. Similarly to the approach by Ter-Akopyan, an *answer type* is determined through named-entities and question words. Afterwards, the potential documents and passages are retrieved. This step could be shortened, since we already know the answer to our clickbait has to be in the related document. In the end, an answer sentence can be

- (1) What Happened to Frank Ocean's Staircase? (*Direct*)
- (2) How Angelina Jolie Told Brad Pitt She Wanted a Divorce (*Indirect*)
- (3) *This* is the worst Arab state for women (*No question word*)

Figure 5.2: Questions in clickbait.

extracted.

Nonetheless, only a fraction of clickbaits are actually questions. Figure 5.2 displays three different types of clickbait. The first clickbait shown is phrased as a question (1). While the second clickbait is not a question, it contains the question word *how* and can be rephrased to the question "How did Angelina Jolie tell Brad Pitt she wanted a divorce?" (2). Even the last clickbait, which does not contain a question word at all, can easily be rephrased by replacing the word *this* with the question word *which* (3). Generally, it can be said, that the problem of clickbait spoiling is very much related to question answering. Instead of engineering a whole new process, it may be easier to adapt already existing structures in question answering to spoil clickbait. Clickbait is still a big problem in social media. This thesis has created Webis-Clickbait-19, a corpus of clickbait-spoiler-pairs, and has shown promising results in clickbait spoiling, which can be expanded by further work in the future.

Appendix A

Source Code and Data

Since there have been some problems with the whereabouts of a data set at the beginning of this thesis work, I want to make sure that it is clear where all my data can be found. As part of this bachelor thesis, three different repositories were created in the Webis GitLab in order to separate the content of different subject areas. The thesis, all references and figures can be found in the *wstud-thesis-puschmann*¹ repository.

The source code to recreate and modify Webis-Clickbait-19 can be found under *wstud-webis-clickbait-corpus-19*². Since the required corpora are quite big, they are not part of the repository. Webis-Clickbait-17 is divided into two sets, the data for training is available on the website of the *Clickbait Challenge*³. The validation set is only available internally for now. Webis-Clickbait-17 and Webis-Clickbait-18 can also both be found in my folder on webis20⁴.

The spoiling experiments from Chapter 4 are saved in *wstud-webis-clickbait-spoiling-19*⁵. More detailed information about the contents can be found in the respective repositories. A final version of all repositories will be available in my webis20 folder and on the data medium submitted with this thesis. I hope that my work will be used for further advances in the field of clickbait spoiling.

¹<https://git.webis.de/webisstud/wstud-thesis-puschmann> [accessed June 14, 2019]

²<https://git.webis.de/webisstud/webis-clickbait-corpus-19> [accessed June 14, 2019]

³<https://www.clickbait-challenge.org/#data> [accessed June 14, 2019]

⁴webis20/data-in-progress/wstud-thesis-puschmann/webis-clickbait-corpus-19/corpora

⁵<https://git.webis.de/webisstud/webis-clickbait-spoiling-19> [accessed June 14, 2019]

Bibliography

- Amol Agrawal. Clickbait detection using deep learning. In *2nd International Conference on Next Generation Computing Technologies (NGCT)*, pages 268–272, October 2016. doi: 10.1109/NGCT.2016.7877426. 2
- Ankesh Anand, Tanmoy Chakraborty, and Noseong Park. We used neural networks to detect clickbaits: You won’t believe what happened next! *CoRR*, abs/1612.01340, December 2016. 2
- Prakhar Biyani, Kostas Tsioutsoulouliklis, and John Blackmer. "8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality. In *Thirtieth AAAI Conference on Artificial Intelligence*. Association for Computational Linguistics, December 2016. ISBN 978-1-57735-760-5. 2
- Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. Stop clickbait: Detecting and preventing clickbaits in online news media. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 9–16, August 2016. 2
- Khalid El-Arini and Joyce Tang. Click-baiting. August 2014. URL <https://newsroom.fb.com/news/2014/08/news-feed-fyi-click-baiting/>. [accessed June 6, 2019]. 2
- Nicholas Fearn. How to make money with a news website. TechRadar.Pro, April 2017. URL <https://www.techradar.com/how-to/how-to-make-money-with-a-news-website>. [accessed May 30, 2019]. 1
- Maksym Gabielkov, Arthi Ramachandran, Augustin Chaintreau, and Arnaud Legout. Social Clicks: What and Who Gets Read on Twitter? In *ACM SIGMETRICS / IFIP Performance 2016*, Antibes Juan-les-Pins, France, June 2016. URL <https://hal.inria.fr/hal-01281190>. 1
- Sanda M. Harabagiu, Steven J. Maierano, and Marius A. Paşca. Open-domain textual question answering techniques. *Nat. Lang. Eng.*, 9(3):231–

- 267, September 2003. ISSN 1351-3249. doi: 10.1017/S1351324903003176. URL <http://dx.doi.org/10.1017/S1351324903003176>. 5.2
- Christopher Hooton. Downworthy plug-in offers upworthy headlines without the hyperbole. January 2014. URL <https://www.independent.co.uk/life-style/gadgets-and-tech/news/downworthy-plug-in-offers-upworthy-headlines-without-the-hyperbole-9083881.html>. [accessed June 6, 2019]. 2
- Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. September 2018. URL <https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf>. [accessed June 9, 2019]. 4.4, 4.4, 4.4, 4.4, 5.1
- George Loewenstein. The psychology of curiosity: A review and reinterpretation. *Psychology Bulletin*, July 1994. doi: 10.1037/0033-2909.116.1.75. 1
- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schuetze. *Scoring, term weighting, and the vector space model*, pages 100–123. Cambridge University Press, 2008. ISBN 978-0-511-80907-1. doi: 10.1017/CBO9780511809071.007. 4.3.1, 4.3.1
- Katerina Eva Matsa and Elisa Shearer. News use across social media platforms 2018. Pew Research Center, September 2018. URL <https://www.journalism.org/2018/09/10/news-use-across-social-media-platforms-2018/>. [accessed June 3, 2019]. 1
- Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. Clickbait detection. In *ECIR*, 2016. doi: 10.1007/978-3-319-30671-1_72. 2
- Martin Potthast, Tim Gollub, Matthias Hagen, and Benno Stein. The clickbait challenge 2017: Towards a regression model for clickbait strength. *CoRR*, abs/1812.10847, 2018a. URL <http://arxiv.org/abs/1812.10847>. 1, 2
- Martin Potthast, Tim Gollub, Kristof Komlossy, Sebastian Schuster, Matti Wiegemann, Erika Patricia Garces Fernandez, Matthias Hagen, and Benno Stein. Crowdsourcing a large corpus of clickbait on twitter. In *The 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, August 2018b. ISBN 978-1-948087-50-6. 2, 2, 3.1
- Md Main Uddin Rony, Naeemul Hassan, and Mohammad Yousuf. Diving deep into clickbaits: Who use them to what extents in which topics with what effects? *CoRR*, abs/1703.09400, 2017. URL <http://arxiv.org/abs/1703.09400>. 2

- Md Main Uddin Rony, Naeemul Hassan, and Mohammad Yousuf. Baitbuster: A clickbait identification framework. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 8216–8217, February 2018. 2
- Victoria L. Rubin, Chris Brogly, Nadia Conroy, Yimin Chen, Sarah E. Cornwell, and Toluwase V. Asubiaro. A news verification browser for the detection of clickbait. *Journal of Open Source Software*, 4(35):1208, 2019. doi: <https://doi.org/10.21105/joss.01208>. 2
- Elisa Shearer. Social media outpaces print newspapers in the u.s. as a news source. Pew Research Center, December 2018. URL <https://www.pewresearch.org/fact-tank/2018/12/10/social-media-outpaces-print-newspapers-in-the-u-s-as-a-news-source/>. [accessed June 3, 2019]. 1
- Grigori Sidorov, Alexander Gelbukh, Helena Gomez-Adorno, and David Pinto. Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model. *Computacion y Sistemas*, 18:491 – 504, 09 2014. ISSN 1405-5546. URL http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462014000300007&nrm=iso. 4.3.2, 5.2
- Bagrat Ter-Akopyan. Korpuskonstruktion und Entwicklung einer Pipeline für Clickbait-Spoiling. Bachelor thesis, Bauhaus-Universität Weimar, Faculty Media, Media Informatics, December 2017. URL https://webis.de/downloads/theses/papers/terakopyan_2017.pdf. 2, 5.2, 5.2