Automatic Creation of Marginalia

Aaron Lang and Robin Jegan and Andreas Henrich
Otto-Friedrich-Universität Bamberg
Chair of Media Informatics
{aaron.lang,robin.jegan,andreas.henrich}@uni-bamberg.de

Abstract

In contrast to similar tasks such as keyphrase prediction or text summarization, the automatic creation of marginalia has not been explored yet. This paper takes the first steps to advance research in this area. For our experiments, we used marginalia data extracted from German computer science textbooks and analyzed the marginalia's properties. We utilized methods from the similar task of keyphrase prediction to test how suitable they are for the creation of marginalia and adapted the methods according to the observed marginalia properties and for the use with the German language. The results are evaluated quantitatively and qualitatively. We also highlight limitations of frequently used measures such as F1 and describe why it is desirable to advance research on suitable evaluation measures. We find that GPT-4o and mT5 perform best while for the non-LLM-based methods, SIFRank+ and TF-IDF show promising results.

1 Introduction

Problems in Natural Language Processing (NLP) that focus on producing a shorter version of a text while keeping relevant information – keyphrase prediction and text summarization – are popular research topics due to their practical applicability among other factors. They can lead to time savings as shorter texts can be read faster and search operations can be enhanced by including keyphrases[1].

However, to the best of our knowledge, this work is the first to explore a similar problem: the automatic creation of marginalia. Marginalia are notes on the margin of a text document, often found in textbooks. While any such note regardless of their content fulfills the definition of marginalia, they usually serve as a descriptor of the text section they are adjacent to. For examples taken from German textbooks, see A.4-A.6. We look at *present* marginalia that are taken from the input paragraph as well as *absent* marginalia that do not appear in that order in the source text.

Marginalia can have significant benefits to the reader: They can structure the text document further beyond chapters and subchapters and prime the reader of what the text section will cover. Additionally, they can help the reader save time when skimming the text, searching for a specific text section, or skipping parts that are not relevant. Information retrieval could also be improved when marginalia are available: E.g., if a reader wants to find a text section that describes what algorithms are, the full-text search for "algorithm" will find all text locations of this term which can be plenty in a computer science textbook. If marginalia are incorporated into the search, the text section with the marginalia "algorithm", that is actually concerned with this term, could be returned instead.

We present first steps of research in this domain by analyzing the properties of a marginalia dataset extracted from German computer science textbooks (Section 3), the selection, adaptation and application of existing methods from similar NLP tasks (Sections 4 and 5) and a quantitative and qualitative evaluation of the results (Section 6).

2 Related Work

Automatic creation of marginalia is yet an unexplored task to the best of our knowledge. Hence, a short overview of the similar tasks of text summarization (TS) and keyphrase prediction (KP) will instead be given. However, after creating the marginalia dataset in Section 3 and subsequent analysis of the marginalia therein, it became apparent that for most marginalia, the task of KP is much more similar to the creation of marginalia than TS is. As a consequence, coverage of TS will be kept to a

[1] In some works keyword and keyphrase are used synonymously. In this work, we use the definition that keyphrases consist of one or more words while keywords consist of only a single word.

Proceedings of the 21st Conference on Natural Language Processing (KONVENS), Volume 1: Long and Short Papers, pages 228–240, 2025

minimum.

Whereas TS and KP tasks are inherently similar in reducing a long document into a shorter document while preserving key information, they differ in the precise characteristics of their output. TS creates full-sentence summaries of the original document as a continuous text. KP on the other hand produces a list of one or more keyphrases, which are terms or concepts relevant to the document.

Since TS has many useful practical applications, it is a major research topic in NLP with a multitude of TS methods ranging from simple heuristic methods all the way to employing Deep Learning and Large Language Models (LLMs).

TextRank is an example for a TS method and was the first approach to model documents as graphs with sentences as vertices and their relations to each other as edges (Mihalcea and Tarau, 2004). The approach connects those edges (sentences) that have overlapping content and are thus considered similar. TextRank employs a similarity measure that is dependent on the number of overlapping words that appear in both sentences which is then used for edge-weighting. A ranking algorithm finds the n highest-ranked sentences and concatenates them to a single summary. In addition, TextRank can also be applied to operate on a word-level instead of a sentence-level basis, which transforms it into a KP method.

While this is an extractive method, i.e., it can only create summaries from sentences that appear verbatim in the original document, there are also abstractive methods able to create summaries by generating new text that paraphrases the input. These include various approaches such as the use of sequence-to-sequence architectures (e.g., Chen et al. (2021)) or reinforcement learning techniques (e.g., Gunasekara et al. (2021)).

A similar distinction between extractive and abstractive summarization can be made for KP methods where keyphrase extraction (KE) methods identify which phrases in the original text are suitable keyphrases (present keyphrases) and keyphrase generation (KG) methods can additionally create phrases not present in the original text (absent keyphrases).

YAKE! (Campos et al., 2018) is a statistical KE method and a commonly used baseline. It incorporates several features of the tokenized input document, such as the word frequency or the position of the sentence the word appears in. To extend this to keyphrases of up to three words, keyword components are multiplied, normalization by length to avoid bias towards longer keyphrases is applied and the term frequency of the keyphrase is used so that more frequent candidates get a better (i.e. lower) score.

Beyond statistical methods, KE encompasses a wide range of other methods following various techniques including graph-based approaches such as SingleRank (Wan and Xiao, 2008) (an extension of TextRank) or PositionRank (Florescu and Caragea, 2017). Topic-based methods aimed at extracting representative keyphrases for a document's topic, e.g. by using Latent Dirichlet Allocation (LDA), such as TopicRank (Bougouin et al., 2013) or MultipartiteRank (hereafter: MPRank) (Boudin, 2018) are similarly in use, while embedding-based approaches such as EmbedRank (Bennani-Smires et al., 2018) or SIFRank+ (Sun et al., 2020) utilize sentence embeddings like Doc2Vec or others.

In addition to present keyphrases found in the input document, KG methods can create absent keyphrases that do not appear verbatim in the input document. Again, there exist a multitude of approaches and methods, many of which are based on an encoder-decoder framework. The training strategies are diverse and include reinforcement learning (Chan et al., 2019) or application of multi-task models (Ahmad et al., 2021) among others.

Wu et al. describe a prompt-based learning KG method that uses predefined input prompts to purposefully control the creation of absent keyphrases (Wu et al., 2022). While some other approaches do not handle present and absent keyphrases separately (e.g. in Meng et al. (2017)), Wu et al. argue that this can create absent keyphrases that are not relevant for the input document. As a first step, overlapping words between the input document and absent keyphrases in the ground truth are extracted as Wu et al. notice that many absent keyphrases contain words that are present in the input document (Wu et al., 2022). Those overlapping keywords kw are used to create prompts of the form "`phrase of kw is [MASK] [MASK] kw [MASK] [MASK]`" or "`other phrases are [MASK] [MASK] [MASK] [MASK]`" with special `[MASK]`-tokens getting concatenated to create a common absent keyphrase. While the former prompt enforces the inclusion of the keyword kw in the keyphrase, the latter also accounts for keyphrases that do not contain overlapping keywords. The application of multi-task-training makes the training of overlapping keyword extraction, the identification of

present keyphrases, and the generation of absent keyphrases possible simultaneously.

While there are similarities between the creation of marginalia and KP, KP differs in so far that it produces a list of keyphrases that can cover multiple aspects of the corresponding document. As an example, the author keywords for Sun et al. (2020) are "keyphrase extraction, pre-trained language model, sentence embeddings, position-biased weight, SIFRank" which encompass various aspects at different granularity such as the name of the broad NLP task, the specific name of the proposed model and the utilized technologies. Marginalia creation on the other hand works on smaller parts of a coherent document and produces only a single marginalia for each document part. Thus, marginalia can depend on the context of prior or later document parts. In a chapter about the programming language Java for example, one paragraph could present its history and another paragraph its installation. "Java" could be a fitting keyphrase for both but too generic as marginalia for either, since the whole chapter is about Java.

3 Data

Due to the lack of publicly available marginalia datasets, we created a dataset by automatically extracting marginalia and their corresponding text section from German computer science textbooks using Python scripts. The textbooks are published by dpunkt.verlag GmbH and are accessible via O'Reilly's online library[2]. Since they are viewed in the browser, the textbooks' HTML can be used for extraction. Having the same publisher also makes their HTML structure more homogeneous. Still, some adaptations must be made manually for each book, which – along with finding books containing marginalia – makes creating a large dataset costly. Hence, our dataset is limited to 39 textbooks resulting in 11,808 data tuples.

Each tuple contains a text section, the corresponding marginalia and metadata such as the book's title, chapter name, ISBN and also the text of the surrounding (sub)chapter. Since it cannot be derived from the HTML which text sections exactly correspond to which marginalia, it is assumed that the text section a marginalia belongs to is all the text following the marginalia until the next marginalia or until a new chapter is encountered.

Following extraction, data cleaning steps are performed, such as the removal of full sentences. Marginalia that are full sentences most often act as a summary of the corresponding text section, which is not regarded typical for marginalia and thus is not desired in our case. From initially 13,118 tuples, approximately 8% (1,057) are detected as full sentences. However, this depends on the included textbooks and on the precision of the mechanism used for full sentence detection.

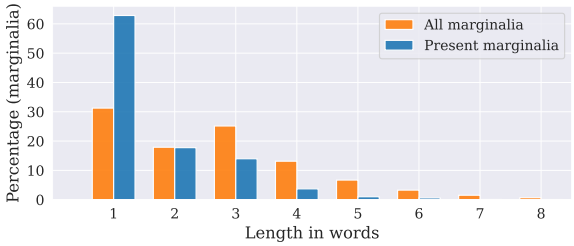

Figure 1: Relative frequency of all marginalia (orange) and present marginalia (blue) in words.

Figure 1 shows the length distribution of all extracted marginalia in orange, while the blue bars are constrained to present marginalia only. It can be seen that marginalia in the dataset usually consist of only a few words, while present marginalia are on average even shorter and consist of just a single word approximately twice as often. However, since checking for the presence of marginalia involves finding the marginalia string as a whole in the text section, shorter marginalia are more likely to be present. Text sections that have marginalia are on average 133.32 words long with a median of 100 words.

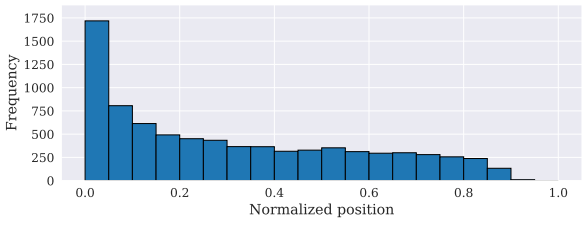

Figure 2: Normalized position of present marginalia in the text sections.

In the marginalia dataset 34% of marginalia are present while 66% are absent. This was determined by searching for the lemmatized string of the marginalia inside the corresponding lemmatized text section.

[2]See https://learning.oreilly.com/home/ for the main search page. For a concrete example, see https://learning.oreilly.com/library/view/react-2nd-edition/9781098123857/ (all web resources last accessed on: 07.05.2025)

Some KP methods use the word position in the text section to rank keyphrase candidates. Figure 2 shows the frequency of present marginalia's position in the text sections. It shows that present marginalia occur especially frequently in the first 5% of the text sections.

4 Methods

The distribution of present (34%) vs. absent (66%) marginalia observed in the marginalia dataset calls for the use of KP methods that are capable of finding both present and absent keyphrases. However, this does not necessarily mean that extractive methods can only be successful (i.e., find a marginalia identical to the reference) in the marginalia dataset in 34% of the cases. It is possible that while the reference (ground-truth) marginalia is absent, there is a similarly fitting present marginalia in the text which can be extracted. To test this assumption, we used not only generative but also extractive methods for creating marginalia.

For KE, we employed several statistical methods as baselines: TF-IDF (Luhn, 1957; Sparck Jones, 1972), YAKE! (Campos et al., 2018), PositionRank (Florescu and Caragea, 2017) and MultipartiteRank (Boudin, 2018). In a 2023 survey of KP methods, Xie et al. found that SIFRank+ is the best unsupervised deep learning KE method (Xie et al., 2023).

SIFRank+ combines smooth inverse frequency (SIF) (Arora et al., 2017) as sentence embedding model and ELMo (Peters et al., 2018) as pre-trained language model. SIFRank+ creates word embeddings of candidate keyphrases and the input document using ELMo and aggregates them into sentence embeddings using SIF (Sun et al., 2020). The sentence embeddings of each candidate keyphrase and the input document are subsequently compared by cosine similarity. The higher the similarity, the more fitting the candidate keyphrase is considered to be. Eventually, the scores of each candidate keyphrase are weighted by their first occurrence position in the input document.

Due to the increasing capabilities of LLMs throughout various NLP tasks, we employed mT5 (Xue et al., 2021) fine-tuned on the marginalia dataset. Furthermore, we experimented with GPT-4o (Hurst et al., 2024) using naive retrieval-augmented generation to add more context to the input documents. For all methods except GPT-4o, each input document consists of a text section for which a marginalia should be created. Thereby, context in the sense of previous and succeeding text sections is not used and cannot be utilized as it is not supported by the KP methods. For GPT-4o, we enrich the input document by its previous text paragraph and its marginalia, if available, as well as the succeeding text paragraph to experiment if additional context is helpful in creating marginalia.

5 Implementation

To use the aforementioned methods for the automatic creation of marginalia, some adjustments must be made, especially to adapt for the German language and the characteristics of marginalia in the dataset. Aside from the necessary adjustments all methods are already implemented and can be used as part of the Python library pke[3] or are provided by the respective authors.

Adjustments for the different baseline methods include the use of a German stopword list provided by pke, specification of German as language parameter and – depending on the method – restriction of candidate selection based on word length or based on most frequent POS combinations in the marginalia dataset in the form of a list or grammar.

To use SIFRank+, German ELMo options- and weights-files need to be employed. German lemmatization is done via spaCy and a German stopword list is used. For word weighting, a domain-specific word frequency list is created from the marginalia dataset plus a generic word frequency list that uses the most frequent words of German Wikipedia[4]. Finally, the most frequent POS combinations in the marginalia dataset are used for candidate selection.

For mT5, a checkpoint that is pre-trained on German TS is needed in order to subsequently fine-tune for creation of marginalia on the marginalia dataset. Several checkpoints are available on HuggingFace[5]. After experiments on a sample basis, we deemed mt5-small-finetuned-amazon-en-de[6] as the best suited checkpoint for marginalia creation. However, this needs to be evaluated further. A checkpoint pre-trained on German KP would have been preferable but was not available.

Prompts for GPT-4o were designed to include preceding and succeeding text sections as context for marginalia creation. GPT-4o is directly and ex-

[3] https://github.com/boudinfl/pke

[4] https://github.com/IlyaSemenov/wikipedia-word-frequency

[5] https://huggingface.co/

[6] https://huggingface.co/anibahug/mt5-small-finetuned-amazon-en-de

plicitly asked to produce marginalia instead of producing keyphrases or learning to produce marginalia (see A.3). Prompts were processed via OpenAI's Batch API[7] which lead to costs of $13.23 for all 10,075,771 input-tokens and $0.38 for the output-tokens including initial tests.

6 Evaluation

We quantitatively evaluated all methods using the F1 measure, Mean Reciprocal Rank (MRR) and MoverScore while also qualitatively evaluating the selected methods in the form of a survey. Quantitative evaluation took place on the marginalia dataset's test split containing 1182 instances. Qualitative evaluation was limited to 100 randomly sampled instances.

6.1 Quantitative Evaluation

Most papers on KP employ the F1 measure, among others, to evaluate their approach, many of them exclusively. The F1 measure only lexically measures if system keyphrases (produced by the method under consideration) agree with reference keyphrases (ground-truth keyphrases), thereby not considering semantically equivalent or similar keyphrases. Still, we decided to include it for easier comparison with other approaches on one hand, and on the other hand for comparison of the same method between the task of KP in other works and the creation of marginalia in our work.

Table 1 shows the F1 score for all methods. It is calculated by first determining the F1 score for each individual instance, then averaging these individual scores to a single (macro) F1 score. Since all methods except mT5 and GPT-4o return multiple marginalia, we consider the F1 scores at rank 1 for better comparability. This is also a more realistic approach since in a real-world scenario, we are only interested in one marginalia per text section. It can be observed that both LLM-based methods – mT5 and GPT-4o – achieve the best F1 scores by a large margin. When only absent marginalia are considered for the calculation of F1, GPT-4o (0.0274) performs considerably better than mT5 (0.0008) in generating absent marginalia (not shown in Table 1). Among the other methods, SIFRank+ performs best. In addition, it can be seen that the position-based weighting of SIFRank+, which SIFRank lacks, appears to have a significant positive effect on marginalia extraction. SIFRank+ was originally designed for longer documents (Sun et al., 2020) but in this case performs better even for short documents. Although it is the simplest method, TF-IDF achieves the fourth-best F1 score by a significant margin. F1 scores at rank M (considering all produced keyphrases) paint a similar picture and are given in the appendix in Table 4.

Method	F1	MRR	MoverScore
TF-IDF	0.0431	0.0750	0.5626
YAKE!	0.0085	0.0324	0.5732
SingleRank	0.0000	0.0060	0.5193
TextRank	0.0008	0.0095	0.5350
PositionRank	0.0161	0.0517	0.5463
TopicRank	0.0228	0.0376	0.5569
MPRank	0.0245	0.0386	0.5573
SIFRank	0.0161	0.0405	0.5492
SIFRank+	0.0668	0.1050	0.5580
mT5	0.1320	0.1320	0.6376
GPT-4o	0.1299	0.1299	0.6543

Table 1: For both, F1 and MoverScore, values are calculated for the first generated marginalia only. For MRR, all generated marginalia are scored, but note that mT5 and GPT-4o only produce a single marginalia instead of a list of marginalia as the other methods do.

For those methods that produce multiple marginalia, we used the MRR to evaluate how early in the result list methods are able to rank matches. A match denotes the system and reference marginalia being identical after stemming. Table 1 shows the MRR for all methods. Here, SIFRank+ and TF-IDF show the highest MRR after the LLM-based methods, thus their matches appear earlier on average in the result list compared to other methods.

Without further adaptation, these measures – F1 and MRR – only consider system and reference keyphrases to match when they are identical (usually after stemming as done in this work) which neglects alternate wordings, synonyms or grammatical variance.

In contrast to the aforementioned measures that work solely lexically and to tackle the problem of the need for exact matches, MoverScore (Zhao et al., 2019) is an example of a semantic similarity measure that compares the system and reference marginalia. MoverScore is typically used for the evaluation of automatic text summarization methods, where the similarity between the summary created by the system and the ground-truth summaries is quantified. This means system and reference marginalia do not necessarily have to be exact matches. Unlike summaries, however, marginalia are usually much shorter, which could reduce MoverScore's expressiveness. Table 1 shows the

[7] https://platform.openai.com/docs/guides/batch

MoverScore values for all methods at rank 1 between system marginalia and reference marginalia. Again, LLM-based methods perform best, whereas the other methods are closer together than in the previous measures. Unlike TF-IDF, YAKE! achieves a much lower F1@1 score compared to SIFRank+, but a moderately higher MoverScore.

6.2 Qualitative Evaluation

To mitigate said limitations of both lexical and semantic measures (see Limitations) and to see how well quantitative and qualitative results agree, we conduct an online survey with 100 questions containing marginalia and text sections randomly sampled from the marginalia dataset. Each participant answers as many questions as they like, which consist of a text section and two randomly selected system marginalia for that text section. The participant has to choose which of the two marginalia is better fitting to the text section or if both are equally fitting, hereafter called "rating" the text section. The marginalia for each text section are produced by five different KP methods (GPT-4o, mT5, SIFRank+, PositionRank and YAKE!), two of which are randomly selected for each question. Hence, different participants can have different marginalia to choose from for the same text section. This comparison between two alternatives was deemed to be easier and also faster than ranking or scoring all five methods for each text section. To increase the likelihood of the same text section being rated by multiple participants with different marginalia alternatives, the number of KP methods for this survey was limited to five.

During the course of the survey, 412 questions were answered by 38 participants. Consistent with the quantitative findings, GPT-4o and mT5 are the two most preferred methods with a relative frequency of 24.46% and 21.84%, respectively. SIFRank+ (14.32%) and PositionRank (13.35%) are both on a similar level, while YAKE! (7.28%) is far less often preferred than the other methods. SIFRank+ being more often preferred than YAKE! is in congruence with the respective F1@1 scores in Table 1. While this also holds for PositionRank, the difference to YAKE! is much smaller. In contrast, the MoverScore of YAKE! is higher than that of SIFRank+ and PositionRank, which does not agree with the qualitative findings, while it does agree with GPT-4o's and mT5's MoverScores. Notably, in 16.75% of the ratings both given methods were judged as equally good.

Additionally, the left side of Figure 3 shows how often methods were preferred depending on pairwise combinations. This means that if, e.g., GPT-4o and YAKE! are presented to participants, it shows how often each alternative was preferred over the other. Since the possible pairwise combinations are not equally distributed, each value in Figure 3 is normalized by the number of occurrences of the respective combination. The matrix can be read in a row-wise manner: As an example, the row for GPT-4o shows the probabilities that when paired with some other method, a participant will prefer GPT-4o. Given GPT-4o and YAKE!, participants choose GPT-4o over YAKE! with a probability of 0.78, while the reverse has a probability of 0.11. Since participants can also rate two methods as equally good, the probabilities do not necessarily add up to 1. At a glance, the green coloring of GPT-4o's row indicates that it is predominantly preferred over any other method. The same holds for mT5 with the exception that it is less frequently preferred over GPT-4o. The red color of the row for YAKE! shows the low probability that YAKE! is preferred over any other method. For the rows of SIFRank+ and PositionRank, such a uniform pattern of preference is less clearly visible. Again, GPT-4o and mT5 take the lead in concordance with F1@1 and MoverScore values, while YAKE! comes in last. As before for the relative frequency of preference, SIFRank+ and PositionRank are close together, whereas their F1@1 scores indicate a significantly higher difference.

The similarity regarding the preference towards SIFRank+ and PositionRank can also be seen on the right in Figure 3, which shows how frequently two methods were regarded as equally good. Here, SIFRank+ and PositionRank are especially often regarded as equally good. Surprisingly, SIFRank+ and mT5 are similarly often regarded as equally good, which was less apparent in the quantitative results.

Due to the restriction on pairwise comparisons, the survey did not yield a direct ranking of methods. One way to derive a ranking from the survey results is the use of the Elo rating (Elo, 2008). Originally designed for the rating of chess players, the Elo rating assigns a rating to every player. With each match between two players, a defeat decreases, while a victory increases the player's rating. When treating the pairwise comparison of methods (questions) as matches between the methods with the preferred method as the winner and

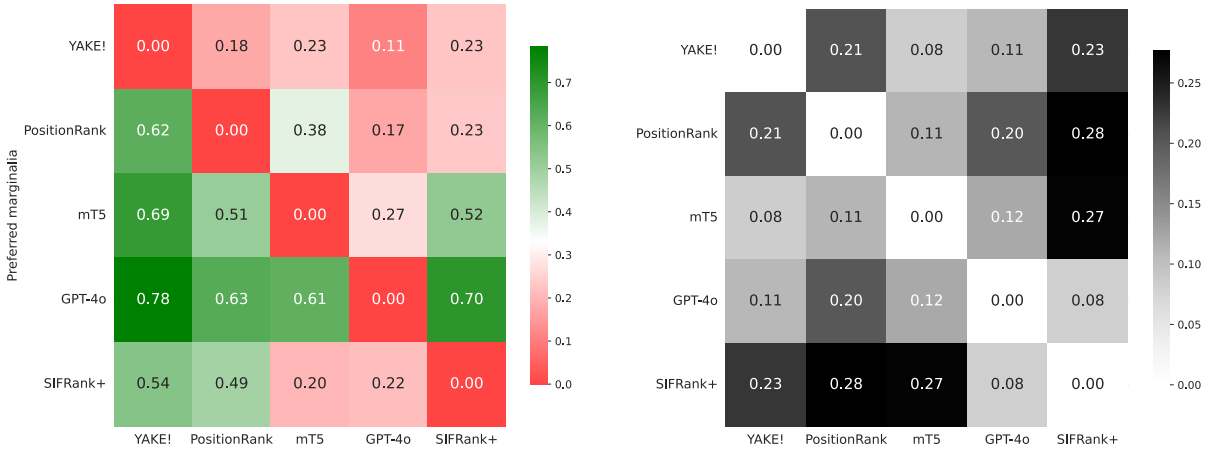

Figure 3: The values in the cells show the normalized probabilities of participants preferring a method (row) over another method (column) in the left part of the figure and the normalized probabilities of participants judging two methods as equally good in the right side of the figure.

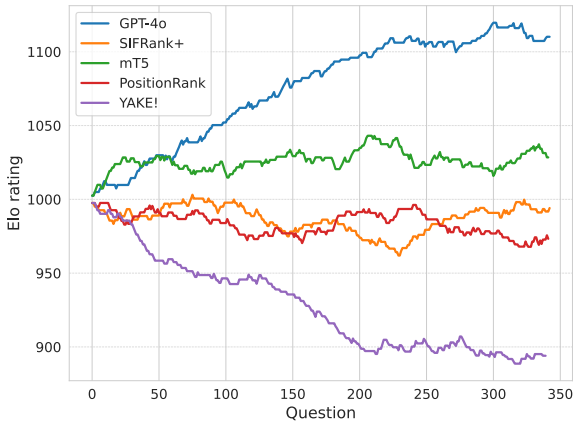

Figure 4: Development of Elo ratings throughout the survey questions. If marginalia were deemed equally good, they were excluded for the computation of the Elo rating.

the other method as the loser, we can apply the Elo rating to derive a rating for each method and subsequently create a ranking.

The development of the Elo ratings throughout all questions can be seen in Figure 4. GPT-4o has the highest rating, while YAKE! has the lowest. Both are clearly separated from the rest of the methods. SIFRank+ and PositionRank are close together and change their ranking throughout the questions multiple times. mT5 is separated from both but with less distance than GPT-4o or YAKE!.

Through manually reviewing text sections as well as the created marginalia and the reference marginalia, cases can be observed where the results produced by KP methods are fitting keyphrases but not fitting marginalia. E.g., for a text section (see Section A.1) that describes how from usage profiles so-called load profiles are generated using an appropriate tool, SIFRank+ extracts the phrase "nutzungsprofilen"[8] (usage profiles) which can be considered an appropriate part of a set of keyphrases for the text section. However, as a marginalia, this would be misleading since the text section does not describe usage profiles but load profiles instead. As the task of KP is not constrained to produce only a single result, it can be assumed that this problem is less relevant for KP but important for marginalia creation. However, as checking for this issue is a manual review process, due to time constraints, we cannot quantify the scope of the issue.

Finally, despite the limitations of quantitative and qualitative evaluation (see Limitations), based on consistent tendencies for F1 score, MRR, MoverScore and survey results, it can be concluded that GPT-4o and mT5 create the best marginalia out of the considered methods.[9] SIFRank+ is the third-best method concerning F1 score and MRR and surpasses PositionRank in those measures by a large margin, but according to survey results is on a similar level to PositionRank.

Among the baseline methods, TF-IDF performs best despite its simplicity. In contrast to the other baselines whose information is constrained on the current document, TF-IDF incorporates corpus-wide document frequency information, which

[8]This example, however, is grammatically incorrect in German as a stand-alone marginalia. See Limitations for a brief discussion of this problem.

[9]Admittedly, the scores are rather low overall, which is however to be expected for this limited quantitative evaluation.

might be one of the reasons for its success in creating marginalia. It seems worthwhile to include TF-IDF as a baseline in future research and examine how it performs in qualitative evaluation.

7 Discussion/Conclusion

In this paper, we have, to the best of our knowledge, presented the first analysis, implementation and evaluation of automatically generating marginalia. In order to accomplish this task, a dataset was compiled by extracting marginalia from German computer science textbooks. The automatic extraction of marginalia was tested with techniques from different NLP-tasks, i.e., text summarization, keyphrase extraction and keyphrase generation. During implementation, the production of keyphrases – both extractive and generative – were identified as more applicable for the use-case presented in this paper. Thus, various extractive statistical methods such as TF-IDF and SIFRank+ and the generative methods mT5 and GPT-4o were applied.

Evaluation was conducted purely on the dataset produced for this paper, hence a comparison with other datasets or other systems was not possible. While no single method led in all metrics, a strong preference for generative methods such as GPT-4o and mT5 compared to other methods, including all extractive methods, was noticed in most experiments. A qualitative evaluation was furthermore conducted through a user study, which showed a commanding lead for the marginalia produced by GPT-4o compared to all other methods. Extractive methods lag behind generative models, however, of the extractive methods, SIFRank+ and TF-IDF, which was only included as a baseline, show promising results.

Future work in this field of research is possible in different aspects. The dataset is one such aspect, which can be extended, both in size and by incorporating different textbooks or other texts with marginalia apart from computer science books. Thus, more training data would be available and the applicability to different domains could be explored. Additionally, making such a dataset publicly available would be highly desirable to drive research in this area forward and make reproducing research results easier. More experiments with other techniques, e.g., by parameter-tuning or updated fine-tuning, or incorporating larger context-windows for the methods are further logical next steps. Dedicated approaches for marginalia generation can also be conceived, e.g., by incorporating not only more text as contextual information but also previously produced marginalia for prior sections of the same text.

Also, as discussed in Limitations, none of the applied metrics are entirely fitting for the use-case presented here. Reference-based metrics are naturally bound to the reference marginalia used in the dataset, while more complex metrics such as MoverScore are more suited to other task areas such as summarization approaches. Still, quantitative evaluation as well as qualitative discussion of the results are necessary and need further research.

Limitations

Regarding the construction of the marginalia dataset, the automatic extraction can on the one hand introduce noise or errors into the dataset and on the other hand, is dependent on the consistency and lack of uniformity of the book's HTML structure. The simplifying assumption mentioned in Section 3 regarding the mapping of text sections to marginalia can further introduce noise and impair the performance of the utilized methods.

But even if the extraction would yield perfect results, data quality would still be an issue as only a certain type of marginalia is desired for our purpose: marginalia that describe what their text section is about. Although data quality is checked on a sample basis, not all 11,808 tuples can be checked by hand. Thus, during data cleaning, the first 500 marginalia with the shortest text sections were manually examined. Here, the aim was to remove marginalia that are considered atypical – such as summaries or marginalia that describe the purpose rather than the content of their text section, e.g. "Einführung" (introduction) – and especially those that could only be created using external information, as they are likely to impact training. Of those checked, 89 were removed. While this proportion may seem high, this does not necessarily reflect the rest of the dataset. Assuming that the shorter text sections are, the less likely it is that marginalia can be derived from them, the percentage of problematic marginalia is expected to be lower in the rest of the dataset. However, it could still be significant enough to impact model training and evaluation. Due to the heterogeneity of atypical marginalia – some are comments, others are summaries and some rely on external information – rigorous au-

tomatic checking of the whole dataset is difficult to implement. Despite efforts during data cleaning and post-processing, it is likely that the marginalia dataset still contains an unknown percentage of atypical marginalia. Apart from the quality of the marginalia dataset which deserves further investigation, the evaluation results are still relevant since all methods are evaluated on the same data and can thus be compared with each other.

Since the marginalia dataset contains copyrighted material, we unfortunately cannot distribute it. However, we provide the steps and scripts we used for extraction in our GitHub repository[10].

In a 2023 survey of KP methods, Xie et al. found that while SIFRank+ is the best unsupervised deep learning KE method, WR-SetTrans is among the best (supervised) generative methods (Xie et al., 2023). Hence, we originally planned to include WR-SetTrans in our work. However, after training, WR-SetTrans did not produce any marginalia and we were unable to find the cause within time. Its predecessor SetTrans (Ye et al., 2021) did produce results, but only for some instances. Therefore, we excluded WR-SetTrans and SetTrans for this work. Making WR-SetTrans work correctly deserves further investigation as it might be capable of producing good marginalia.

Concerning quantitative evaluation, while the measures discussed are commonly used for evaluating KP methods, they have several limitations. Firstly, they consider the reference keyphrases to be the only correct keyphrase while in reality there may be other similarly or equivalently fitting keyphrases, e.g. synonyms. Applied to marginalia, this could prove even more true: With only a single marginalia as a target for a specific text section, the chance is higher that an alternative keyphrase is erroneously deemed incorrect.

Also, when checking whether system marginalia agree with reference marginalia, we used exact matching. This means both need to be identical after stemming to be considered matching. We conducted further experiments using approximate matching (Zesch and Gurevych, 2009), which due to size constraints could not be included and discussed in Section 6. However, the results were not surprising and are shown in the appendix in Table 2. Other forms of less strict matching could yield different results and are worth exploring.

[10] https://github.com/aaronlba/marginalia-creation

For languages such as German with a multitude of grammatical forms for nouns as well as verbs, while extractive methods might identify the correct marginalia in the text section, due to their grammatical form, the marginalia could be grammatically incorrect as a stand-alone word. As an example, “nutzungsprofilen” in the text given in the appendix in Section A.1 is in dative plural in German. For a stand-alone marginalia, it would need to be “nutzungsprofil” or “nutzungsprofile” (nominative singular or plural). Furthermore, the marginalia should be upper-case to represent the grammatically correct German noun “Nutzungsprofile”.

For MoverScore in particular, the brevity of marginalia, which often consist of only a few words, gives MoverScore little context to work with. Also, the comparison is still constrained to the reference marginalia as the only correct solution, while there may be other valid marginalia. To counteract this restriction, we experimented with calculating the MoverScore between system marginalia and their respective text sections instead. However, this approach did not yield meaningful results for the extractive methods since it is dependent on the average marginalia length of those methods. This means that the longer the extracted marginalia, the higher the similarity to the respective text section is considered to be.

For the qualitative results, while the survey results are mostly in accordance with quantitative results and also personal impressions from samples, the significance is limited due to the low number of questions and participants. Since only 100 text sections out of 11,808 were randomly selected for the survey questions, it is possible that for some methods, a disproportionately high number of bad marginalia or good marginalia are part of the survey, which can skew the results.

Additionally, each question received on average only about 4 ratings. This is relatively low given that each question has 10 possible answer pairings, resulting in 1,000 possible combinations across the 100 questions.

References

Wasi Ahmad, Xiao Bai, Soomin Lee, and Kai-Wei Chang. 2021. Select, extract and generate: Neural keyphrase generation with layer-wise coverage attention. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natu-*

ral Language Processing (Volume 1: Long Papers), pages 1389–1404, Online. Association for Computational Linguistics.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 221–229, Brussels, Belgium. Association for Computational Linguistics.

Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 667–672, New Orleans, Louisiana. Association for Computational Linguistics.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 543–551, Nagoya, Japan. Asian Federation of Natural Language Processing.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. *A Text Feature Based Automatic Keyword Extraction Method for Single Documents*, pages 684–691. Springer International Publishing.

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural keyphrase generation via reinforcement learning with adaptive rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2163–2174, Florence, Italy. Association for Computational Linguistics.

Yongchao Chen, Xin He, Guanghui Wang, and Junyang Yu. 2021. Improving text summarization using feature extraction approach based on pointer-generator with coverage. In *Web Information Systems and Applications*, pages 489–496, Cham. Springer International Publishing.

Arpad Emmerich Elo. 2008. *The Rating of Chessplayers: Past and Present*. Ishi Press International.

Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Chulaka Gunasekara, Guy Feigenblat, Benjamin Sznajder, Ranit Aharonov, and Sachindra Joshi. 2021. Using question answering rewards to improve abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 518–526, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592, Vancouver, Canada. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Yi Sun, Hangping Qiu, Yu Zheng, Zhongwei Wang, and Chaoran Zhang. 2020. Sifrank: A new baseline for unsupervised keyphrase extraction based on pre-trained language model. *IEEE Access*, 8:10896–10906.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, volume 8, pages 855–860. AAAI.

Huanqin Wu, Baijiaxin Ma, Wei Liu, Tao Chen, and Dan Nie. 2022. Fast and constrained absent keyphrase generation by prompt-based learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11495–11503.

Binbin Xie, Jia Song, Liangying Shao, Suhang Wu, Xiangpeng Wei, Baosong Yang, Huan Lin, Jun Xie, and Jinsong Su. 2023. From statistical methods to deep learning, automatic keyphrase prediction: A survey. *Information Processing & Management*, 60(4):103382.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021. One2set: Generating diverse keyphrases as a set. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics.

Torsten Zesch and Iryna Gurevych. 2009. Approximate matching for evaluating keyphrase extraction. In *Proceedings of the International Conference RANLP-2009*, pages 484–489.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

A Appendix

A.1 Example Paragraph

The following paragraph is taken from the textbook *Basiswissen Abnahmetest* to show one of the textbook paragraphs in the marginalia dataset:

"Aus Nutzungsprofilen werden dann mit einem entsprechenden Werkzeug sogenannte Lastprofile erzeugt, also technische Abbilder der zuvor spezifizierten Nutzungsprofile zum Zwecke der Lasterzeugung. Lastprofile können dabei mehrere Nutzungsprofile unterschiedlicher Benutzergruppen oder Personas kombinieren, um ein möglichst realistisches Abbild der zu erwartenden oder üblicherweise anfallenden Last zu simulieren (da es zu erwarten ist, dass nicht immer nur eine Benutzergruppe mit dem Produkt zu jedem Zeitpunkt interagiert)."[11]

A.2 Quantitative Evaluation

In addition to the F1 scores and MRR at rank 1 using exact matching shown in Table 1, Table 2 shows these values using approximate matching.

Method	F1	MRR
TF-IDF	0.1168	0.6413
YAKE!	0.0829	0.4442
SingleRank	0.0606	0.3333
TextRank	0.0718	0.3858
PositionRank	0.0942	0.5694
TopicRank	0.1997	0.3553
MPRank	0.2010	0.3604
SIFRank	0.0857	0.4856
SIFRank+	0.1161	0.6497
mT5	0.2750	0.2750
GPT-4o	0.2991	0.2991

Table 2: F1 scores and MRR for all considered methods at rank 1 using approximate matching.

Table 3 and Table 4 show the precision and recall values for exact and approximate matching at rank M. Values at rank 1 are omitted since they are identical to their respective F1 scores.

A.3 Prompt

The original German prompt template is given in the GitHub repository.[12] The following prompt template was translated from German via DeepL. Text in curly braces is replaced by the corresponding data.

[11] https://learning.oreilly.com/library/view/basiswissen-abnahmetest/9781098129729/ Chapter 4.3.2 High-Level-Performanzabnahmetests

[12] https://github.com/aaronlba/marginalia-creation.

Method	Precision	Recall
TF-IDF	0.0175	0.1591
YAKE!	0.0119	0.1184
SingleRank	0.0045	0.0355
TextRank	0.0058	0.0423
PositionRank	0.0186	0.1591
TopicRank	0.0257	0.0668
MPRank	0.0251	0.0677
SIFRank	0.0148	0.1235
SIFRank+	0.0219	0.1920
mT5	0.1320	0.1320
GPT-4o	0.1299	0.1299

Table 3: Precision and recall for all considered methods at rank 1 using exact matching.

Method	Precision	Recall
TF-IDF	0.0313	0.1894
YAKE!	0.0216	0.1336
SingleRank	0.0079	0.0981
TextRank	0.0099	0.1159
PositionRank	0.0329	0.1557
TopicRank	0.0344	0.2311
MPRank	0.0340	0.2329
SIFRank	0.0261	0.1393
SIFRank+	0.0389	0.1898
mT5	0.1320	0.2750
GPT-4o	0.1299	0.2991

Table 4: Precision and recall values at rank M with M being the number of marginalia produced by a method, using approximate matching.

The following paragraphs are taken from the subchapter "{subchapter_name}" in the chapter "{chapter_name}" in the textbook "{book_title}". Where available, the corresponding marginalia is given for each paragraph:

Paragraph N-1 reads:
"{predecessor_paragraph}" This paragraph has the marginalia "{predecessor_marginalia}".

Paragraph N reads:
"{current_paragraph}"

Paragraph N+1 reads:
"{successor_paragraph}"

Task:
Create a marginalia for paragraph N. If available, the previous paragraph N-1 and the following paragraph N+1 may be helpful as context for creating consistent marginalia (e.g., uniform wording of the marginalia). Important: The marginalia should refer exclusively to the content of paragraph N. The marginalia does not necessarily have to summarize the content of the paragraph, but can also simply indicate what the paragraph is about.

Notes on the length of the marginalia you are looking for:
Most marginalia are between 1 and 4 words long, with very few exceeding 6 words. Keep the marginalia as short as possible.

Note on the output format:
Only output the generated marginalia without a prefix such as "Marginalia:" or quotation marks.

A.4 Example Marginalia 1

In this example of a textbook paragraph and its system marginalia, most methods match with the reference.

"JSON (JavaScript Object Notation) ist eine Darstellung von Daten, die vor allem für JavaScript optimiert ist. Wie JavaScript auch, sind die Daten dynamisch typisiert. Mittlerweile gibt es aber eigentlich für alle Programmiersprachen passende JSON-Bibliotheken. Es gibt außerdem Typsysteme wie JSON Schema [16], die für JSON eine entsprechende Validierung ergänzen. Damit steht JSON Datenformaten wie XML in nichts mehr nach."[13]

Method	Marginalia
TF-IDF	json
YAKE!	JavaScript Object Notation
SingleRank	für json eine entsprechende validierung
TextRank	für json eine
PositionRank	json datenformaten wie xml
TopicRank	json
MPRank	json
SIFRank	json
SIFRank+	json
mT5	JSON
GPT-4o	JSON
Reference	JSON

Table 5: Reference marginalia given by the authors of the source text and system marginalia at rank 1 created by the various methods for A.4.

A.5 Example Marginalia 2

In this example, several methods overlap with the reference, but do not match exactly. However, GPT-4o's marginalia can be seen as semantically equivalent to the reference.

"Zu jedem Add-on gibt es eine Kurzbeschreibung des Funktionsumfangs und auf welche Anwendungsfälle es abzielt. Ein Add-on kann im

[13] https://learning.oreilly.com/library/view/microservices-2nd-edition/9781492068686/
Chapter 9 Integration und Kommunikation

Laufe der Zeit in unterschiedlichen Versionen zur Verfügung gestellt werden, diese Versionen werden mit entsprechenden Release-Notes dokumentiert. Des Weiteren gibt es zusätzliche Informationen, zu welcher Vaadin-Version ein Add-on kompatibel ist oder mit welchen Browsern ein Add-on verwendet werden kann, wenn es hier entsprechende Einschränkungen geben sollte. Wir können uns auch über das zugrunde liegende Lizenzmodell informieren oder über weiterführende Links auf z.B. eine Online-Demo, den Sourcecode oder eine Ticketverwaltung zugreifen. Abb. 10-2 Detailseite Add-on am Beispiel Vaadin Calendar Über die Anzahl der Downloads können wir einen gewissen Eindruck über die Popularität eines Add-on gewinnen. Jeder Nutzer hat zudem die Möglichkeit, ein Add-on mit eins bis fünf Sternen zu bewerten. Es kann optional ein Kommentar verfasst werden, um dem Ersteller eines Add-on Rückmeldungen zu geben. Weitere interessante Hinweise und Hilfestellungen aus der Vaadin-Community finden wir zusätzlich im Vaadin-Forum unter der Kategorie Add-ons (siehe [Vaadin-Forum]).”[14]

Method	Marginalia
TF-IDF	add-on
YAKE!	Kurzbeschreibung des Funktionsumfangs
SingleRank	detailseite add-on am beispiel vaadin calendar über die anzahl der downloads
TextRank	ein add-on mit
PositionRank	vaadin-version ein add-on
TopicRank	entsprechende einschränkungen
MPRank	entsprechende einschränkungen
SIFRank	add-on rückmeldungen
SIFRank+	add-on
mT5	Anzahl der Downloads
GPT-4o	Add-on Detailinformationen
Reference	Detailinformationen zu einem Add-on

Table 6: Reference marginalia given by the authors of the source text and system marginalia at rank 1 created by the various methods for A.5.

A.6 Example Marginalia 3

In this example none of the methods match the reference. However, some marginalia describe relevant aspects of the text section, e.g., *konflikte* (conflicts) or *überprüfung und abstimmung* (review and coordination). Furthermore, it can be argued that GPT-4o's marginalia is even more fitting than the reference marginalia. Here, the reference marginalia is less a description of the content of the text section and more a statement of its consequence.

[14] https://learning.oreilly.com/library/view/vaadin/9781457188336/
Chapter 10 Add-Ons

“Die Überprüfung auf Widersprüche und Abstimmung der Anforderungen muss fortlaufend (in unterschiedlicher Intensität) über das gesamte Requirements Engineering hinweg erfolgen. Die Überprüfung und Abstimmung von Anforderungen verursachen dabei zusätzlichen Aufwand und somit zusätzliche Kosten. Der durch die Überprüfung und Abstimmung der Anforderungen erzielte und in den vorangegangenen Abschnitten beschriebene Vorteil (Kostenersparnis, Erhöhung der Akzeptanz des Systems, Unterstützung der Definition innovativer Anforderungen) ist in der Regel jedoch wesentlich höher als die durch die Überprüfung und Abstimmung entstehenden Kosten. Zur Abstimmung der Anforderungen an ein zu entwickelndes System ist es notwendig, Konflikte zu identifizieren und die auftretenden Konflikte aufzulösen. Dies geschieht im Rahmen eines systematischen Konfliktmanagements. Das Konfliktmanagement im Requirements Engineering umfasst die folgenden vier Aufgaben:” [15]

Method	Marginalia
TF-IDF	anforderungen
YAKE!	Requirements Engineering hinweg erfolgen
SingleRank	der durch die überprüfung und abstimmung der anforderungen
TextRank	durch die überprüfung und abstimmung der anforderungen
PositionRank	überprüfung und abstimmung
TopicRank	konflikte
MPRank	konflikte
SIFRank	abstimmung von anforderungen
SIFRank+	überprüfung auf widersprüche
mT5	Überprüfung und Abstimmung
GPT-4o	Kosten und Nutzen der Abstimmung
Reference	Verringerung der Kosten und Risiken in späteren Phasen

Table 7: Reference marginalia given by the authors of the source text and system marginalia at rank 1 created by the various methods for A.6.

[15] https://learning.oreilly.com/library/view/basiswissen-requirements-engineering/9781098129231/
Chapter 4 Praktiken für die Erarbeitung von Anforderungen