

## Construct relation extraction from scientific papers: Is it automatable yet?

Jonas Scharfenberger  
 Leuphana University Lüneburg  
[jonas.scharfenberger@leuphana.de](mailto:jonas.scharfenberger@leuphana.de)

Burkhardt Funk  
 Leuphana University Lüneburg  
[burkhardt.funk@leuphana.de](mailto:burkhardt.funk@leuphana.de)

### Abstract

*The process of identifying relevant prior research articles is crucial for theoretical advancements, but often requires significant human effort. This study examines the feasibility of using large language models (LLMs) to support this task by extracting tested hypotheses, which consist of related constructs, moderators or mediators, path coefficients, and p-values, from empirical studies using structural equation modeling (SEM). We combine state-of-the-art LLMs with a variety of post-processing measures to improve the relation extraction quality. An extensive evaluation yields recall scores of up to 79.2% in construct entity extraction, 58.4% in construct-mediator/moderator-construct extraction, and 39.3% in extracting the full tested hypotheses. We provide a manually annotated dataset of 72 SEM articles and 749 construct relations to facilitate future research. Our findings offer critical insights and suggest promising directions for advancing the field of automated construct relation extraction from scholarly documents.*

### 1. Introduction

Structural equation modeling (SEM) is a powerful and frequently applied statistical tool for empirical research in information systems (Evermann & Tate, 2009; Henriksen & Pedersen, 2007; Urbach & Ahlemann, 2010). This family of statistical methods supports researchers in theory building from empirical studies by testing hypothesized causal relationships between latent variables on observed data. Approximately 20% of published articles in the major information systems (IS) journals are SEM-based (Urbach & Ahlemann, 2010). Recent discussions (e.g.,

by Evermann and Rönkkö, 2023) critically reflect methodological advances and underline the ongoing relevance of SEM in IS research. The popularity of SEM extends beyond IS research to various behavioral and social sciences, as underlined by over 19,000 SEM-related articles available on PubMed.

In light of the substantial body of work utilizing SEM and the research principle of "standing on the shoulder of giants", effectively identifying and leveraging existing research is a pivotal component of theoretical advancements. This challenge is further amplified by the jingle-jangle-fallacy (Larsen & Bong, 2016) and the jungle conundrum (Song et al., 2021). The former describes the misconception that two different constructs (latent variables) are equivalent because they share the same label, or that two identical constructs are different because they have different names across different studies. The latter represents the challenge of identifying similar causal models. To address these problems, IS researchers have proposed a variety of knowledge repositories (Dann et al., 2019; Li et al., 2020; Watson & Webster, 2020) and models (Larsen & Bong, 2016; Ludwig et al., 2020; Song et al., 2021) and thereby facilitate the process of reviewing literature. However, these approaches all share an expensive prerequisite: an immense human effort to retrieve the required information from articles which naturally limits the amount of data available. The required information usually consists of a subset (depending on the use case) of the following information on construct relations: the related constructs' names and definitions, names of moderators or mediators (if applicable), the path coefficient and p-value.

Recent efforts towards an automation of the knowledge extraction task in this field can be categorized into natural language processing (NLP) and

computer vision approaches. Vision attempts train deep learning models to extract construct relations from graphical representations of SEMs (Huettemann et al., 2023; Scharfenberger & Funk, 2023; Scharfenberger et al., 2021; Schoelch et al., 2022). Textual advances train models on sentences retrieved from research articles to extract information on causal claims (Li & Larsen, 2011; Mueller & Abdullaev, 2019; Mueller & Huettemann, 2018) or build question answering systems for IS research articles (Ebert et al., 2023).

Recent progress in large language models (LLMs) have set a variety of new benchmarks. These include interacting with PDF files or long text sequences to summarize them, answer questions or extract structured information. Researchers from other fields present promising results in structured information retrieval from scientific articles (Buehler, 2024; Dagdelen et al., 2024; Polak & Morgan, 2024). Thus, LLMs hold the potential to further enhance the quality of the construct relation extractions from articles' texts and complement image-based approaches.

In this study, we seek to thoroughly assess the current automation potential, explore the effect of text preprocessing, and identify current issues that can be addressed by future work in this field. To that end, we develop a construct relation extraction tool using LLMs and pose the following research questions:

*RQ1: How big is the performance gap between our LLM approach and an error-free extraction of constructs and their relations?*

*RQ2: How does the oftentimes inevitable PDF parsing impact the overall quality of extracted construct relations?*

*RQ3: How do individual sections of a research article contribute to the quality of the extractions?*

Thereby, we contribute to (1) the ongoing efforts in IS research to tackle the jingle-jangle-fallacy and jungle conundrum, and (2) the structured information extraction from scholarly articles by proposing a manually labelled dataset of 72 SEM articles with a total of 749 construct relations.<sup>1</sup>

## 2. Related Work

In this section, we briefly summarize key findings from related work in the context of construct relation or more general structured information extraction from scientific papers. We present closely related computer vision and broader natural language processing (NLP) approaches.

<sup>1</sup>Dataset, code and prompts are publicly available at: <https://github.com/purplesweatshirt/ConstructRelationExtraction>

### 2.1. Computer vision approaches

Previous work in the context of construct relation extraction often leverages figures that compactly summarize the key findings from SEM research. These figures cover the related constructs, path coefficients, indications of significance, and sometimes observed variables. Scharfenberger et al. (2021) propose a sequential pipeline consisting of various models to identify pages containing these figures, detecting and cropping the figures, and detecting nodes and path coefficients. Schoelch et al. (2022), Scharfenberger and Funk (2023), and Huettemann et al. (2023) pursue a similar approach and use synthetic data to train deep learning models to reconstruct the graph structures from the images. Recall scores of 0.46 for detecting the related constructs (no mediators or moderators) and their associated path coefficients and 0.144 when including optical character recognition are reported (Scharfenberger & Funk, 2023). Huettemann et al. (2023) limit their approach to graphical models without intersecting edges and report an  $F_1$  score of 0.72 where they allow the construct names to deviate from the ground truth by a Levenshtein distance of up to three. The text corpus of the underlying articles is ignored in all of these works and offers a potential to further improve the output quality.

### 2.2. NLP approaches

Approaches closely related to our work seek to extract causal claims, which among other things consist of a hypothesis identifier, causing and affected variables, moderators, mediators, and the signs of the effect or interaction (Mueller & Abdullaev, 2019; Mueller & Huettemann, 2018). Both rule-based solutions (Mueller & Huettemann, 2018) and deep learning models (Mueller & Abdullaev, 2019) are proposed and evaluated by measuring precision, recall, and  $F_1$  scores of the aforementioned subelements (between 0.6 for mediators and 0.9 for signs) of a causal claim (Mueller & Huettemann, 2018). However, the authors do not report the extraction quality of entire causal claims.

More recent research applies both off-the-shelf and fine-tuned LLMs to extract structured information mostly in the field of material sciences. Dagdelen et al. (2024) seek to extract chemical names, formulas, descriptions, and applications in a JSON format from articles' abstracts. They fine-tune a GPT-3 and a Llama-2 model on this task. The models are evaluated in three different ways: (1) automatically measuring exact matching of ground truth and extraction, (2) manually evaluating the output and considering equivalent ground

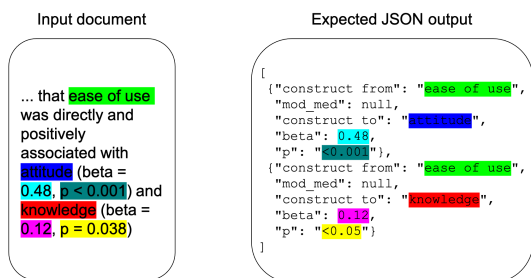


Figure 1. Exemplary document input and desired JSON output.

truth and predictions as correct even if they would not have satisfied the automatic matching criteria, (3) measuring the time saved by correcting the predictions rather than labeling from scratch. The manual  $F_1$  scores of the subelement (formula, name, etc.) extractions in a JSON group range from 0.5 to 0.943. In a related use-case, Polak and Morgan (2024) use ChatGPT to extract material names, their units and values from scientific articles' full-texts in HTML or XML format. A series of tailored prompts is applied to each sentence of the article. An extracted triplet is considered correct if the extracted and ground truth units and values are identical and the material names are equivalent, which is determined manually. Their approach using GPT-4 achieves precision and recall scores of 90.8% and 87.7%.

### 3. Methodology

In this section, we introduce our approach to transform a scholarly article (in the form of an XML or PDF file) into a structured JSON representation of the construct relations contained in this document. The desired construct relation extraction consists of the following 5-tuples: the names of the two related constructs (and the name of the moderator or mediator, if applicable), the path coefficient, and the p-value (if explicitly stated). An example can be seen in Figure 1.

We design a pipeline structure with the purpose of our evaluation and the following requirements in mind. First, the pipeline should allow various sizes of input text chunks, to investigate its effect on the output quality. We hypothesize that smaller chunks yield a higher recall and larger chunks a better precision. Second, the pipeline should incorporate components that aim at reducing the number of hallucinated extractions. Third, post-processing steps should be applied to enhance the output quality (e.g., categorization of the significance, mapping of acronyms, etc.) which should improve both recall and precision. Thus, we use three-phased pipeline

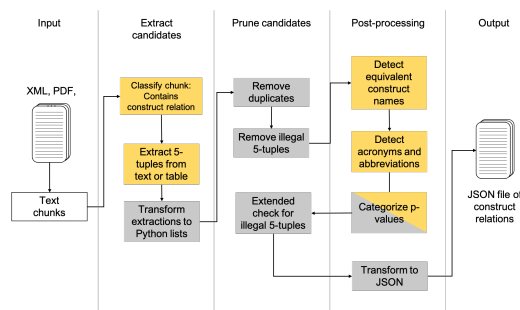


Figure 2. Proposed pipeline. Orange boxes represent LLM queries and gray boxes represent rule-based scripts.

structure (see Figure 2) that enables us to answer our research questions. Each phase consists of simple heuristics and (except for the pruning phase) of LLM queries.

#### 3.1. Generate relation candidates

The first step of our pipeline is to transform the input file (e.g., XML or PDF) into text chunks. If the underlying file is an XML file, we retrieve the texts on the paragraph level and obtain additional information on the section type the text is obtained from. While a naive assumption is that only the result section and tables within it contain relevant information on construct relations, we keep all sections (except for supplementary files) for extractions to later evaluate this hypothesis. In case a PDF file is inputted, the file is parsed and the resulting text is split into chunks of 150 words (slightly longer than the average paragraph length) with overlaps of 15 words. Both values could be further optimized and will be explored in the evaluation.

Independent of the underlying file, the text chunks are then classified by an LLM using a zero-shot prompt to decide whether the chunk contains a construct relationship or not. If they do, they are passed to the candidate generation phase, where we follow up by letting the LLM extract the construct relations from the paragraph as a list of lists. We opt for this nested list structure as an intermediate result rather than the JSON format, since the JSON structure introduces a lot of additional tokens in the output string through the repeated key names. Assuming that the chunk contains information whether it is a table or not, we process tables differently by first letting the LLM take the text as input and reformat it, and then in a follow-up query let it extract the construct relations in the nested list format. In both cases, we extract the lists from the response string by using regular expressions. Furthermore, we store the chunk index in this intermediate result to

be able to provide the input text as context for the extraction.

### 3.2. Prune Candidates

In an attempt to increase the precision of our extractions and reduce the amount of "hallucinated" extractions, we prune these lists by applying certain rules. We state that a valid extraction has to fulfill the following conditions: (1) it needs to have exactly five elements, (2) it must have a non-empty `construct_from` and `construct_to`, (3) it may not have a non-numerical path coefficient, and (4) the `construct_from` and `construct_to` may not be equal. All extractions that are duplicates or violate at least one of these conditions are removed

This pruning routine is applied twice. The first time directly after extracting the relation candidates and the second time after executing the post-processing functions (see next subsection). The second time the routine is extended by an additional rule, where construct names have to have a length of more than four characters. Since we assume that after post-processing all acronyms have been mapped to construct names, the remaining short names would represent observed variables which we do not want to extract.

### 3.3. Enhance Quality of Candidates

The set of extracted and pruned construct relations are used as input for the post-processing phase, which consists of three substeps. First, we attempt to find equivalent construct names that have different names within the extracted relations of a given paper. We encode each extracted construct name using an embedding model and calculate the cosine similarity between all pairs of embedding vectors. If the similarity of a pair of construct names exceeds 95% as suggested by Buehler (2024), we consider the constructs as equivalent. These construct names are then mapped to the name appearing most frequently in the underlying article. Second, we extract all unique constructs names from the relations and query the LLM for each construct name to determine whether it is an abbreviation or acronym. If so, we provide sentences from the document that contain the string "`<acronym>`" (e.g., "(PEOU)") as context to the LLM and let it reconstruct a mapping of the acronym to its original value in the form of a Python dictionary. We use the dictionary to map the acronyms and have our final nested list. Finally, we process the extracted significance such that it fits into one of the following categories: "`<0.001`", "`<0.01`", "`<0.05`", "`<0.1`", or "`≥ 0.1`". This categorization is inspired by the Disknet knowledge repository (Dann

et al., 2019). If the current extracted p-value is given as number, we simply convert it to one of the categories. If it is given as a string, we extract the comparison operator and float using regular expressions and categorize it. Otherwise, we query the LLM to transform it into one of the categories or "null" if the extracted value cannot be transformed. The resulting list of 5-tuples is pruned again and then used to construct a JSON file of the construct relations.

### 3.4. Implementation details

Our proposed pipeline is model-agnostic. Both, a commercial (OpenAI's "gpt-4o-2024-05-13" model) and open-sourced LLM are used for evaluation purposes. We opt for the "Phi-3-Medium-128k-Instruct" (Abdin et al., 2024) language model, since it is open-source, achieves solid benchmark scores, has a large context size of 128k tokens and can be run on a single A100 GPU. The embedding model used for construct similarity is the widely-applied "bge-large-en-v1.5" (Xiao et al., 2023) model which can be easily exchanged. We implement our pipeline in Python using the Huggingface library (Wolf et al., 2020) and the OpenAI API. For more implementation details see our Github Repository.

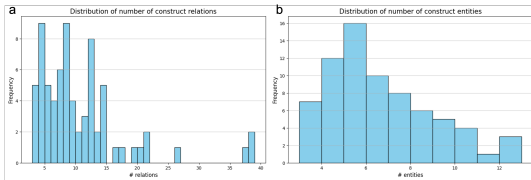
## 4. Evaluation

In this section, we present our evaluation dataset, explain the evaluation strategy, and thoroughly evaluate the pipeline in the remaining subsections.

### 4.1. Dataset

We choose a dataset of PubMed articles to evaluate the pipeline and answer our research questions. This decision is based on the large number of SEM-related articles and the availability of the BioC API (Comeau et al., 2019), which allows us to retrieve the full texts in XML format without having to parse the PDF files. Thus, we can also evaluate the loss of information from the PDF parsing.

We conduct a keyword search in titles and abstracts for "structural equation modeling" and filter the results for "free full texts." This yields approximately 10k articles. We select a random subset of 100 recent articles and manually annotate the articles with our 5-tuple construct relation extractions in JSON format. We only extract tested relations, i.e., that contain a path coefficient. In this process, we exclude 13 articles that do not contain such information in the articles' texts or cannot be accurately labelled with certainty. Another 15 articles are selected to refine the LLM prompts during



**Figure 3.** Distribution of the number of a) construct relations and b) construct entities per article.

the development of the pipeline.

The final test dataset consists of 72 manually annotated articles (24 from 2022, 46 from 2023, and 2 from 2024). In total, the test set is composed of 408 uniquely named construct entities and 749 construct relations. On average, a paper contains 10.4 construct relations and 6.22 different constructs (see Figure 3). Most papers (57) present one model, whereas 11 papers present two and four papers present three models. The papers presenting more than one model contribute to outliers in terms of the number of presented construct relations.

## 4.2. Evaluation strategy

In the following sections, we evaluate four different stages of the construct relation extraction by providing the average recall, precision and  $F_1$  scores on article level. First, we evaluate the correctness of the extracted construct names (in the following tables referred to as "Entity"). Second, we evaluate the extracted 3-tuples which consist of the names of the related constructs and the name of the mediator or moderator if applicable. Third, we evaluate the 4-tuples which extend the 3-tuples by the corresponding path coefficients. Lastly, the entire construct relation extraction is evaluated. We include these intermediate steps since different use-cases require a different granularity of information extracted. For instance, while repositories such as Disknet (Dann et al., 2019) require 5-tuples, methods to identify similar causal models (Song et al., 2021) do not require path coefficients and p-values. An example of how these intermediate steps look like is illustrated in Table 1.

We consider an extracted entity correct, if the lower-cased construct name exactly matches the lower-cased ground truth construct name. Similarly, we consider an extracted n-tuple correct if *all* extracted names and values (lower-cased) exactly match their lower-cased ground-truth counterpart.

As acknowledged in prior work (Dagdelen et al., 2024; Polak & Morgan, 2024), this evaluation technique is very strict and rather provides a lower bound for

Category	Example
Text chunk	"The indirect effect of usability on satisfaction through knowledge is significant ( $\beta = 0.15, p < 0.01$ )."
Entities	'usability', 'satisfaction', 'knowledge'
3-Tuple	('usability', 'knowledge', 'satisfaction')
4-Tuple	('usability', 'knowledge', 'satisfaction', -0.15)
5-Tuple	('usability', 'knowledge', 'satisfaction', -0.15, '<0.01')

**Table 1.** Example of the extracted  $n$ -tuples based on a fictional sentence.

the performance. For instance, minor variations in construct names (e.g., ground truth "Ease of Use" and extraction "Ease of Use (EoU)") can lead to a number of extracted relations which are considered false. Thus, we also manually evaluate the extractions to account for equivalent construct names. Construct names are considered equivalent if they are used interchangeably in the underlying text. However, if multiple predicted constructs are equivalent, only the most frequent one is considered a match, reflecting the assumption that our post-processing routine should have identified these synonyms. We argue that this equivalent construct matching does not further contribute to the problem of the jingle-jangle-fallacy, because in the manual extraction process it would be the annotator who would (somewhat arbitrarily) choose on of the various equivalent names from the underlying text.

## 4.3. Extraction Performance

In this section, the construct relation extraction performance of the proposed pipeline with two different LLMs (Phi-3 and GPT-4o) is evaluated automatically (Table 2) and manually (Table 3). We notice drastic improvements in performance when manually evaluating the predictions. In total, we identify 91 pairs of equivalent construct names of which 19 pairs have a Levenshtein distance of less than or equal to 2.

The recognition of construct names works well, with mean recall values ranging from 0.703 (GPT-4o) to 0.705 (Phi-3) and mean  $F_1$  scores ranging from 0.532 (GPT-4o) to 0.501 (Phi-3) in the automatic evaluation. The manual evaluation yields better scores: a mean recall of 0.831 (Phi-3) and a mean  $F_1$  score of 0.620 (GPT-4o). When considering 3-tuple extractions, we notice a serious performance drop in all metrics for both models. Considering 4-tuples leads to a less severe performance drop for both models suggesting that recognizing the path coefficients is not a bottleneck

as it is in the vision-based related work. The manual  $F_1$  score of the pipeline with the GPT-4o model is 0.424, whereas the Phi-3 model yields a score of 0.327. Another significant loss of construct relation quality is observed, when evaluating the entire 5-tuple extraction. Including the p-values drop the manual recall values to 0.356 (Phi-3) and 0.393 (GPT-4o) and  $F_1$  scores of 0.221 (Phi-3) and 0.293 (GPT-4o). This indicates a major difficulty for the LLMs to extract the p-values from the underlying articles.

A further manual investigation of the extractions unveils a couple of common problems. For both models, we still notice acronyms and equivalent construct names in the extractions that could not be handled in the post-processing steps. Furthermore, the Phi-3 model exhibits problems in mistaking interactions of abbreviated variables (e.g., "cca x ar" or "cd dui") for single construct names. In rare cases, Phi-3 falsely extracts other information as construct names (e.g., "strong positive" or "pearson correlation coefficient"). As mentioned before, these problems negatively impact the scores across all tasks.

Overall, the evaluation of the pipeline shows that despite achieving a comparable performance in the entity recognition phase, using the commercial model clearly outperforms using the open-source model in the evaluation of the latter  $n$ -tuple phases. We identify two major problems. First, the low precision due to the abovementioned problems indicates the need for additional measures to verify extracted construct names and relations. Second, in contrast to the extraction of path coefficients, the extraction of p-values is very error-prone and is another bottle-neck in performance.

Task	Model	Rec.	Prec.	$F_1$
Entity	Phi-3	0.705	0.427	0.501
	GPT-4o	0.703	0.464	0.532
3-Tuple	Phi-3	0.454	0.214	0.266
	GPT-4o	0.485	0.314	0.358
4-Tuple	Phi-3	0.402	0.193	0.239
	GPT-4o	0.464	0.307	0.348
5-Tuple	Phi-3	0.273	0.126	0.161
	GPT-4o	0.331	0.210	0.243

**Table 2. Mean performance metrics of the automated evaluation for both models.**

#### 4.4. Components of the pipeline

To estimate the importance of the existing post-processing steps, we successively add the post-processing and pruning methods and report the  $F_1$  scores of the automatic evaluation for all tasks.

Task	Model	Rec.	Prec.	$F_1$
Entity	Phi-3	0.831	0.542	0.615
	GPT-4o	0.792	0.558	0.620
3-Tuple	Phi-3	0.574	0.305	0.355
	GPT-4o	0.584	0.400	0.442
4-Tuple	Phi-3	0.521	0.280	0.327
	GPT-4o	0.558	0.383	0.424
5-Tuple	Phi-3	0.356	0.186	0.221
	GPT-4o	0.393	0.262	0.293

**Table 3. Mean performance metrics of the manual evaluation for both models.**

Only applying the rule-based pruning functions yields  $F_1$  scores of 0.455, 0.258, 0.233 and 0.11 on the four tasks using the Phi-3 model, and 0.525, 0.341, 0.326, and 0.155 using GPT-4o respectively. The additional acronym mapping improves the  $F_1$  scores Phi-3 extraction only in the entity recognition (0.501), whereas it increases the  $F_1$  on all tasks for the GPT-4o model (0.533, 0.356, 0.345, and 0.164). Adding the construct similarity mapping only has minor effects for the Phi-3 model (0.501, 0.266, 0.239, and 0.113). The significance categorization, however, boosts the  $F_1$  scores in the 5-tuple extraction to 0.161 (Phi-3) and 0.243 (GPT-4o).

We conclude that various post-processing measures are needed to enhance the quality of the construct relation extractions. Especially the acronym matching and the significance categorization increase the  $F_1$  scores.

#### 4.5. Parsing PDF files

A myriad of published articles, especially in major IS journals and conference proceedings, is only available in PDF format. Thus, we investigate the performance of the pipeline when providing a parsed PDF file as input rather than paragraphs from the XML file. We use the PyMuPDF library (PyMuPDF Developers, 2024) to parse the PDF files. To quantify the assumed loss of information in the construct relation extraction, we compare the parsed texts and XML paragraph inputs in the automatic evaluation setting of the Phi-3 models.

We observe a higher mean recall value (0.717), precision (0.439) and  $F_1$  score (0.510) in the entity recognition when using parsed inputs (Table 4) compared to the XML paragraphs. Similarly, the mean precision (0.228) and  $F_1$  score (0.274) of the extracted 3-tuples are slightly higher. When including path coefficients, the extraction quality with parsed inputs suffers more than in our initial setting with XML inputs.

Surprisingly, the results of the pipeline with parsed inputs achieves better evaluation scores in the entity recognition and 3-tuple extraction. While the context length is fixed to 150 words for parsed PDFs, we have variable context lengths (min = 7, mean = 107.7, max = 608) for the paragraph inputs from the XML files. These findings suggest that the (on average) longer context length is more suitable for this application. The comparatively larger drop in performance when path coefficients and p-values are included may indicate that the PDF parsing introduces errors with respect to numerical values.

Task	Rec.	Prec.	F <sub>1</sub>
Entity	0.717	0.439	0.510
3-Tuple	0.452	0.228	0.274
4-Tuple	0.386	0.184	0.234
5-Tuple	0.263	0.124	0.156

**Table 4. Mean performance metrics of the pipeline with Phi-3 LLM and parsed inputs.**

#### 4.6. Context length

Based on the hypothesis on the context length from the previous subsection, we conduct a study where we vary the context lengths for the pipeline with parsed inputs using the Phi-3 model.

To thoroughly evaluate the influence of context length, we conducted experiments using context sizes of 100, 150, 300, and 600 words of the parsed texts, with overlaps of 10, 15, 30, and 45 words, respectively. The results, as summarized in Table 5, suggest that moderate to long context lengths (300-600 words) are suitable for our extraction task.

Using a shorter context of 100 words which are comparable to the average input lengths using paragraphs from the XML files, yield better entity recognition recall (0.725) than longer context windows (0.621 to 0.717). However, the precision and  $F_1$  scores are higher for all of the longer input sequences. Intermediate context lengths (150 and 300 words) achieve a good balance between recall (0.717 and 0.698) and precision (0.439 and 0.463). However, the precision and  $F_1$  scores are noticeably higher across all tasks when increasing the context lengths from 150 to 300 words. Further increasing the context length to 600 words decreases the recall of the extractions drastically across all tasks, but outperforms the 150 word input in terms of precision and  $F_1$  score.

While Phi-3 is expected to handle inputs of up to 128k tokens, in our experiments the response strings mostly diverged from construct relations to randomly

repeated stop words. In our experiments using GPT-4o with the entire text as context, we notice good results in the automatic evaluation (Table 6). While our pipeline achieves better recall scores, this approach performs best with respect to the  $F_1$  score. In this setting, we observe a noticeably worse performance in the extraction of  $n$ -tuples when using parsed inputs.

Context size	Task	Rec.	Prec.	F <sub>1</sub>
100 words	Entity	0.725	0.352	0.446
	3-Tuple	0.440	0.167	0.229
	4-Tuple	0.366	0.142	0.194
	5-Tuple	0.266	0.104	0.141
150 words	Entity	0.717	0.439	0.510
	3-Tuple	0.452	0.228	0.274
	4-Tuple	0.386	0.184	0.234
	5-Tuple	0.263	0.124	0.156
300 words	Entity	0.698	0.463	0.530
	3-Tuple	0.446	0.297	0.322
	4-Tuple	0.373	0.249	0.270
	5-Tuple	0.264	0.169	0.187
600 words	Entity	0.621	0.495	0.527
	3-Tuple	0.362	0.304	0.301
	4-Tuple	0.279	0.225	0.229
	5-Tuple	0.192	0.158	0.159

**Table 5. Mean performance metrics for different context lengths using Phi-3 and parsed inputs.**

Task	Parsed	Rec.	Prec.	F <sub>1</sub>
Entity		0.651	0.710	0.670
Entity	✓	0.646	0.719	0.667
3-Tuple		0.389	0.443	0.405
3-Tuple	✓	0.364	0.439	0.379
4-Tuple		0.368	0.425	0.385
4-Tuple	✓	0.320	0.395	0.338
5-Tuple		0.292	0.343	0.308
5-Tuple	✓	0.249	0.307	0.261

**Table 6. Mean performance metrics for using the entire article as context for GPT-4o.**

#### 4.7. Paper sections

The underlying XML files provide a categorization of each paragraph into one of the following classes: "title", "abstract", "intro", "methods", "results", "discuss", "concl", "ref", "fig", and "table". For each construct relation extraction, our proposed pipeline stores the paragraph id that was used as context. Thus, we can easily exclude certain sections from the pipeline's output to argue about each section's contribution to the overall extraction (Table 7). To that

end, we use the construct relation extractions obtained from our pipeline with the Phi-3 LLM and XML inputs.

The results suggest that omitting the paragraphs from the discussion section only decreases the performance metrics across all stages in a minor fashion. Similarly, cutting out the abstract and methods section degrades the extraction to a small degree. While ignoring the discussion section increased the precision score, the same is not true for the abstract, methods and results section. Skipping information from the results section decreases the performance measures the most. Not using the papers’ tables moderately decreases the recall scores, but boosts the precision and  $F_1$  score. This suggests that information is currently extracted from tables not containing relevant information or that the table structure poses challenges in the extraction and needs to be further optimized.

Excluded	Task	Rec.	Prec.	$F_1$
None	Entity	0.705	0.427	0.501
	3-Tuple	0.434	0.214	0.266
	4-Tuple	0.402	0.193	0.239
	5-Tuple	0.273	0.126	0.161
Abstract	Entity	0.678	0.404	0.478
	3-Tuple	0.428	0.211	0.260
	4-Tuple	0.361	0.184	0.224
	5-Tuple	0.230	0.114	0.142
Methods	Entity	0.685	0.419	0.490
	3-Tuple	0.435	0.209	0.258
	4-Tuple	0.385	0.188	0.232
	5-Tuple	0.254	0.121	0.153
Results	Entity	0.591	0.435	0.475
	3-Tuple	0.307	0.194	0.211
	4-Tuple	0.205	0.135	0.144
	5-Tuple	0.133	0.09	0.095
Discussion	Entity	0.698	0.444	0.509
	3-Tuple	0.443	0.223	0.268
	4-Tuple	0.391	0.201	0.240
	5-Tuple	0.263	0.133	0.162
Tables	Entity	0.648	0.540	0.567
	3-Tuple	0.384	0.330	0.331
	4-Tuple	0.355	0.304	0.307
	5-Tuple	0.247	0.198	0.207

**Table 7. Mean performance metrics using Phi-3 when excluding sections from the underlying paper.**

## 5. Discussion

Our study explores the current feasibility of leveraging LLMs to extract construct relations (i.e., tuples of related constructs, mediators/moderators, path coefficients, and p-values) from scientific papers,

focusing on the domain of structural equation modeling. Our thorough evaluation on a dataset of 72 SEM papers from PubMed provides detailed insights into the current state of construct relation extraction and answer our three initial research questions.

### 5.1. Construct relation extraction performance of LLMs

The best performing model with respect to the recall is GPT-4o with our pipeline. It is capable of recognizing 79.2% of all ground-truth construct entities and achieves a recall rate of 58.4% for 3-tuples, 55.8% for 4-tuples, and 39.3% for 5-tuples. Using the open-source Phi-3 model yields slightly worse recall scores (83.1%, 57.4%, 52.1%, and 35.6% respectively). However, both models struggle with a low precision and thus a low  $F_1$  score as well.

As our results suggests, enlarging the context provided to the article can increase the precision. When taking an extreme approach and using the entire text as context for the LLM query, GPT-4o achieves the best  $F_1$  scores in the automatic evaluation for all tasks (0.67, 0.405, 0.385, and 0.308 respectively). The recall scores, however, are higher for using paragraphs as inputs instead of the entire texts. While our results show an advantage in using commercial models, the *applicability of commercial models is limited* due to the costs of API inference calls and licensing issues arising when considering restricted-access articles.

Despite achieving solid scores (considering the complexity of this task) that outperform vision-based approaches (Scharfenberger & Funk, 2023) and can serve as baseline for future work, the overall performance still indicates the *need of human-in-the-loop solutions for construct relation extraction from scientific articles*.

### 5.2. Impact of PDF Parsing

Although there are repositories that provide access to the articles’ raw texts (e.g., Latex files on arxiv or XMLs via PubMed’s BioC API), a substantial number of papers are only available as PDF files. Our evaluation suggests that parsing the PDF files does not necessarily impact the extraction performance in a negative way. When selecting a suitable context length, our pipeline with parsed inputs can achieve better scores than the baseline that uses the paper’s paragraphs as inputs. On our dataset we find that context sizes of 300 to 600 words perform well in terms of  $F_1$  score. In contrast, the relation extraction quality for parsed inputs decreases noticeably when using the entire text as inputs.

### 5.3. Analysis of sections contribution

When optimizing the construct relation extraction for speed and inference costs, excluding sections that do not positively contribute to the construct relation quality can be beneficial. Our findings align with the intuitive assumptions on the importance of the result sections. Excluding other sections (e.g., abstract, methods) has a lesser impact on the overall performance.

Besides the result section, we investigate the assumption that tables compactly summarize findings from the path analysis and can be harnessed for construct relation extraction. Our findings suggest that specialized approaches are necessary to correctly understand the table structure and leverage its contents for the relation extraction. If naive attempts are used, excluding tables from the extraction can yield better  $F_1$  scores while slightly decreasing the recall. It is worth mentioning that our findings are based on the predicted extractions and do not consider from which section relations were extracted in the manual annotation.

### 5.4. Limitations

Our study evaluates the construct relation extraction on a dataset of 72 randomly selected SEM articles from PubMed published in 2022 or later. While this dataset comprises papers from various disciplines, it is mostly limited applications in biomedical, life, and social sciences. Since our data is limited to recently published articles, our findings for parsed PDF files might not hold for a small fraction of old SEM articles. More precisely, parsing scanned PDF articles could reduce the text input quality and thus have a higher impact on the extraction quality.

The field of LLM research is rapidly evolving and new models are frequently proposed. Our results for two state-of-the-art commercial (GPT-4o) and open-source (Phi-3) LLMs can serve as a benchmark scores for construct relation extraction. More recent models are likely to perform even better on this task. However, we do not anticipate that newer models will significantly alter the impact of PDF parsing or the section importance, but they may become better at natively handling parsed tables.

Our study only focuses on analyzing the articles' text bodies and tables. Images which are often provided to compactly summarize the SEM results in a graph structure could further enhance the extraction performance. However, dedicated computer vision solutions or multi-modal models have to be used to extract relations from figures (e.g., Huettemann et al., 2023; Scharfenberger and Funk, 2023).

## 6. Conclusion

IS scholars frequently encounter the jingle-jangle fallacy (Larsen & Bong, 2016) when reviewing literature. Approaches tackling this problem heavily rely on manual effort to extract the required data from papers. To facilitate the data collection, we propose and thoroughly evaluate an LLM-based approach to extract construct relations from scholarly articles. Thereby, we contribute to existing literature in a theoretical and practical way. First, our study marks an important step towards the augmentation of the theorists' work (Scharfenberger et al., 2021). We hope to spark theoretical discussions on the future of theory building through SEM research. Second, we provide a manually annotated dataset to ease and standardize the evaluation of future approaches in construct relation extraction. Additionally, our research questions and evaluation suggest promising fields for future work, which are outlined in the following paragraph.

Our study suggests four propitious research areas for future work. First, our evaluation highlights the need for exploring methods to tackle the low precision scores of the relation extractions, such as an improved acronym mapping or grouping similarly named constructs. Second, the overall scores suggest that it is worthwhile to fine-tune the Phi-3 model or other open-source LLMs for this task. Third, incorporating non-textual knowledge still remains challenging. Currently, including tables does not increase the quality of the extraction. Furthermore, combining the LLM extractions with extractions obtained from SEM figures is important, as some articles only present path coefficients in the articles' figures. Finally, to measure the usefulness of tools like ours, a field study should be conducted to quantify the time savings when integrating the tool into a human-in-the-loop solution. The practical benefits of analyzing the SEM figures can also be evaluated in this context.

## References

- Abdin, M., Jacobs, S. A., Awan, A. A., ..., & Zhou, X. (2024). Phi-3 technical report: A highly capable language model locally on your phone. <https://arxiv.org/abs/2404.14219>
- Buehler, M. J. (2024). Accelerating scientific discovery with generative knowledge extraction, graph-based representation, and multimodal intelligent graph reasoning. <https://arxiv.org/abs/2403.11996>
- Comeau, D. C., Wei, C. H., Doğan, R. I., & Lu, Z. (2019). Pmc text mining subset in bioc:

- About 3 million full text articles and growing. *Bioinformatics*.
- Dagdelen, J., Dunn, A., Lee, S., et al. (2024). Structured information extraction from scientific text with large language models. *Nature Communications*, *15*, 1418.
- Dann, D., Maedche, A., Teubner, T., Mueller, B., Meske, C., & Funk, B. (2019). Disknet – a platform for the systematic accumulation of knowledge in research. *ICIS 2019 Proceedings*, (1).
- Ebert, L., Huettemann, S., & Mueller, R. M. (2023). Hey article, what are you about? question answering for information systems articles through transformer models for long sequences. *56th Proceedings of the Hawaii International Conference on System Sciences*.
- Evermann, J., & Tate, M. (2009). Building theory from quantitative studies, or, how to fit sem models. *ICIS 2009 Proceedings*, (192).
- Evermann, J., & Rönkkö, M. (2023). Recent developments in pls. *Communications of the Association for Information Systems*, *52*, 663–667.
- Henriksen, A., & Pedersen, P. (2007). The application of structural equation modelling in information systems research. *ECIS 2007 Proceedings*, (35).
- Huettemann, S., Mueller, R. M., Larsen, K. R., Dinter, B., & Chiny, J. C. (2023). How best to hunt a mammoth - toward automated knowledge extraction from graphical research models. *Wirtschaftsinformatik 2023 Proceedings*, (87).
- Larsen, K. R., & Bong, C. H. (2016). A tool for addressing construct identity in literature reviews and meta-analyses. *MIS Quarterly*, *40*(3), 529–552.
- Li, J., & Larsen, K. (2011). Establishing nomological networks for behavioral science: A natural language processing based approach. *ICIS 2011 Proceedings*, (24).
- Li, J., Larsen, K., & Abbasi, A. (2020). Theoryon: A design framework and system for unlocking behavioral knowledge through ontology learning. *MIS Quarterly*, *44*(4), 1733–1772.
- Ludwig, S., Funk, B., & Mueller, B. (2020). Using natural language processing techniques to tackle the construct identity problem in information systems research. *53rd Hawaii International Conference on System Sciences*.
- Mueller, R. M., & Abdullaev, S. (2019). Deepcause: Hypothesis extraction from information systems papers with deep learning for theory ontology learning. *52nd Proceedings of the Hawaii International Conference on System Sciences*.
- Mueller, R. M., & Huettemann, S. (2018). Extracting causal claims from information systems papers with natural language processing for theory ontology learning. *51st Proceedings of the Hawaii International Conference on System Sciences*.
- Polak, M., & Morgan, D. (2024). Extracting accurate materials data from research papers with conversational language models and prompt engineering. *Nature Communications*, *15*, 1569.
- PyMuPDF Developers. (2024). *Pymupdf documentation* [Version 1.24.4]. <https://pymupdf.readthedocs.io/en/latest/>
- Scharfenberger, J., & Funk, B. (2023). Parsing causal models – an instance segmentation approach. *Intelligent Information Systems. CAiSE 2023. Lecture Notes in Business Information Processing*, 477.
- Scharfenberger, J., Funk, B., & Mueller, B. (2021). The augmented theorist - toward automated knowledge extraction from conceptual models. *ICIS 2021 Proceedings*, (6).
- Schoelch, L., Steinhäuser, J., Beichter, M., ..., & Stiefelhagen, R. (2022). Towards automatic parsing of structured visual content through the use of synthetic data. *2022 26th International Conference on Pattern Recognition (ICPR)*, 1607–1613.
- Song, Y., Watson, R. T., & Zhao, X. (2021). Literature reviewing: Addressing the jingle and jangle fallacies and jungle conundrum using graph theory and nlp. *ICIS 2021 Proceedings*.
- Urbach, N., & Ahlemann, F. (2010). Structural equation modeling in information systems research using partial least squares. *Journal of Information Technology Theory and Application (JITTA)*, *11*(2).
- Watson, R. T., & Webster, J. (2020). Analysing the past to prepare for the future: Writing a literature review a roadmap for release 2.0. *Journal of Decision Systems*, *29*(3), 129–147.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ..., & Rush, A. M. (2020). Huggingface's transformers: State-of-the-art natural language processing. <https://arxiv.org/abs/1910.03771>
- Xiao, S., Liu, Z., Zhang, P., & Muennighoff, N. (2023). C-pack: Packaged resources to advance general chinese embedding. <https://arxiv.org/abs/2309.07597>