UNCOVERING INCONSISTENCIES AND CONTRADICTIONS IN FINANCIAL REPORTS USING LARGE LANGUAGE MODELS

Tobias Deußer*[1,2], David Leonhard [1,2], Lars Hillebrand [2], Armin Berger [2], Mohamed Khaled [3], Sarah Heiden [3], Tim Dilmaghani [3], Bernd Kliem [3], Rüdiger Loitz [3], Christian Bauckhage [1,2], and Rafet Sifa [1,2]
[1]University of Bonn, Bonn, Germany
[2]Fraunhofer IAIS, Sankt Augustin, Germany
[3]PricewaterhouseCoopers GmbH, Düsseldorf, Germany

ABSTRACT

Correct identification and correction of contradictions and inconsistencies within financial reports constitute a fundamental component of the audit process. To streamline and automate this critical task, we introduce a novel approach leveraging large language models and an embedding-based paragraph clustering methodology. This paper assesses our approach across three distinct datasets, including two annotated datasets and one unannotated dataset, all within a zero-shot framework. Our findings reveal highly promising results that significantly enhance the effectiveness and efficiency of the auditing process, ultimately reducing the time required for a thorough and reliable financial report audit.

Keywords Contradiction Detection · Natural Language Processing · Large Language Models · Financial Reports · Machine Learning

1 Introduction

In the landscape of global finance and accounting, the precision and reliability of financial reports are of paramount importance. These reports serve as the bedrock upon which critical decisions are made by investors, analysts, regulators, and stakeholders. However, the financial reporting landscape is intricate and multifaceted, characterized by a labyrinthine web of data sources[1] and reporting standards, like the International Financial Reporting Standards (IFRS), the German Commercial Code ("Handelsgesetzbuch", HGB) or Japan's Modified International Standards (JMIS). In such a complex environment, contradictions, inconsistencies, and even ambiguities in financial reports can have profound implications, ranging from skewed investment decisions to regulatory compliance challenges, ultimately impacting market stability and investor confidence.

This paper details various approaches to automatically detect such artifacts by harnessing the potential of large language models (LLMs), such as GPT-4 [1] or Llama [2]. Our approach is based on the following key insights:

- LLMs can be trained to understand the complex relationships between different financial relationships [3, 4] and metrics [5].
- The LLM can evaluate these relationships and metrics to identify inconsistencies and contradictions, as seen in Section 4.

*tdeusser@uni-bonn.de, ORCID-ID: 0000-0003-4685-0847

[1]Examples for financial report data sources are the EDGAR database by the U.S. Security Exchange Commission (https://www.sec.gov/edgar/search-and-access), the SEDAR+ database by the Canadian Securities Administrators (https://www.sedarplus.ca), or the DART database by the South Korean Financial Supervisory Service (https://englishdart.fss.or.kr/)

We evaluate our approach on three datasets of real-world financial reports. Our results show that we can identify inconsistencies and contradictions in financial reports effectively, and by deploying our findings and models, we can drastically lighten the workload of financial auditors who are tasked with finding and eradicating such contradictions.

We believe that our work has the potential to improve the accuracy and reliability of financial reporting significantly. By automating the identification of inconsistencies and contradictions, LLMs can help businesses and auditors produce more accurate and reliable financial reports and help investors and creditors make more informed decisions.

The rest of this paper is organized as follows. Section 2 provides a background on machine learning for financial reporting and contradiction detection. Section 3 describes our proposed approaches for uncovering inconsistencies and contradictions in financial reports. Section 4 presents our experimental results. Section 5 concludes our work and discusses limitations and directions for future research.

2 Related Work

The task of contradiction detection, especially within the realm of natural language processing (NLP), has garnered significant attention in recent years due to its applicability in various domains like Finance and Law. In this section, we discuss relevant literature related to both contradiction detection (CD) and the utilization of machine learning for analyzing financial textual data.

Pre-dating contradiction detection is the field of natural language inference (NLI), which is a more general approach to this task and in which it is necessary to ascertain whether a provided hypothesis can be deduced from a given premise. It was first popularized in a challenge in [6], which attracted 17 submissions from different participants. Modern approaches in NLI leverage the power of deep pre-trained transformer models and achieve remarkable performances with these setups [7, 8, 9, 10, 11, 12, 13].

Initial methods for CD usually relied on rule-based systems and lexical matching to identify contradictory statements. [14] aimed to detect inconsistencies by utilizing three distinct forms of linguistic cues: negation, antonymy, and semantic and pragmatic details linked to discourse relationships and [15] combined shallow semantic representations derived from semantic role labeling with binary relations extracted from sentences in a rule-based framework.

More recently, CD was based upon transformer models like BERT [16], RoBERTa [17], or StructBERT [18]. Leveraging said models, [19] presented a contradiction detection approach that considers linguistic semantic features and uncertainty indicators to address the challenge of detecting contradictions in uncertain statements and [20] identified contradictions within sentence pairs, with potentially real-world applications across fields like law enforcement and historical analysis. The specific field of financial contradictions has been studied in [21], who leveraged a RoBERTa model and added specific pre-training.

In addition to English, the task of contradiction detection has been extended to other languages such as Spanish [22, 23], Japanese [24], Persian [25], Arabic [26], Chinese [27], and German [28, 29, 30].

Within the wider scope of automating the audit procedures for financial documents, into which our contradiction detection for financial reports approach fits, [31] introduced a recommender-driven tool aimed at streamlining and partially automating the audit of financial documents, which was subsequently enhanced in [32], [33], and [3]. [4] introduced the concept of artificial intelligence co-piloted auditing, highlighting the collaborative potential between auditors and foundation models like LaMDA [34], DALL-E [35], and GPT-4 [1] in enhancing various audit tasks. [36], [37], [38], and [39] attempted to jointly extract named entities and their relations from financial documents in English, German, French, and Chinese, respectively. Similarly, [40] recognized named entities in financial documents to then anonymize them for further processing. [41] proposed a distributed learning framework that trains audit models from accounting data without revealing sensitive data. [42] used NLP techniques to study the results of government financial statement audits in South Africa.

Despite the advancements in both contradiction detection and financial data analysis using NLP, there exists a gap in the literature regarding the application of large language models for detecting contradictions within financial texts. This paper aims to address this gap by proposing a framework that leverages the capabilities of state-of-the-art large language models for accurately identifying contradictions in financial documents.

3 Methodology

This section aims to describe the motivation for and the general ideas of the relative information extraction of the financial reports up until the obtainment of the results by processing queries with a large language model (LLM). We evaluate three separate datasets in this context: An annotated, sentence-pair-level dataset introduced in [21], an

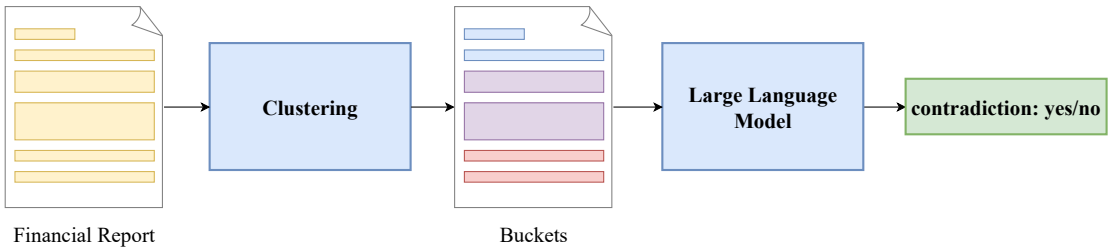

Figure 1: The workflow of our document-level contradiction detection approach. The clustering step is further detailed in the sections 3.2.1, 3.2.2.

annotated, document-level dataset, and a not annotated, document-level dataset. Each of these datasets requires a different approach or evaluation, which we elaborate on in the following sections.

3.1 Sentence-Pair Data

The methodology for finding contradictions between two sentences is a rather simple one: Query the LLM and parse its output to cast the answer into the binary decision of it being a contradiction or not. The prompt we used for this is:

> Given the following paragraph, check if there are any contradictions in it. Your answer has to start with "yes" or "no". Explain your answer.
> paragraph: "{paragraph}"

3.2 Document Data

When we consider data on a document level, we have to implement a further "Bucketing" step into our approach. This is called the combination level, where sub-document buckets of paragraphs are created. Within each of these buckets, contradictions are searched for using a large language model (LLM). A query is sent to the LLM, containing a prompt text with processing and output instructions and the paragraph collection, including paragraph ids. For each query, the LLM outputs either "yes" or "no" and a brief explanation if a contradiction is found. This process is outlined in Fig. 1. The prompt, translated to German if the document language is German, is the following:

> Check the following document to see if it contains any contradictions. Justify your answer. Based on the justification: end your answer with "Yes" if the document contains contradictions, otherwise "No". If the document contains contradictions, cite the respective paragraph numbers in your justification.
> Document: "{document}"

Contradictions are generally expected to become apparent from within a single sentence or two; the whole context of the surrounding paragraph might not be necessary to deduce a contradictory meaning within one or between two or more paragraphs. Thus, a query containing multiple paragraphs contains not only possibly contradictory information but also information irrelevant to the context of the possible contradiction, which should be regarded as noise. Accordingly, increasing the number of paragraphs contained in a query of the LLM could reduce the performance and output quality, as the information relevant to the possible contradiction takes up a smaller amount of the information contained in the prompt and is, therefore, less likely to be subject to the primary attention of the LLM (See Section 4.2.2 for an empirical analysis of this phenomenon). Additionally, increasing the length or number of queries also increases the computational time and costs for both external pay-by-query (e.g., GPT-4 [1]) and self-hosted open-source models (e.g., Llama [2]).

The simplest way to process the paragraphs of the financial report would be to match every paragraph with every other paragraph for a query of the LLM. However, according to simple combinatorics, the number of queries would then be roughly proportional to the number of paragraphs squared:

$$\begin{aligned} N_{\text{queries, paragraph-to-paragraph}} &= \frac{N_{\text{paragraphs}} \cdot (N_{\text{paragraphs}} - 1)}{2} \\ &\propto N^2_{\text{paragraphs}}. \end{aligned} \tag{1}$$

As each financial report we analyze holds between 200 and 1200 paragraphs (see Section 4.3.1), we would have to query any LLM an absurd amount of times and either run into exorbitant costs or computational complexity. To reduce the

number of queries, insight into the content of the paragraphs is needed. This is done by creating paragraph embeddings and clustering paragraphs based on them.

Embeddings transfer the meaning conveyed in natural language into a vector representation that machines can process. While [3] showed that fine-tuned encoders significantly outperformed general models on specific tasks on financial documents, a broader approach that does not require fine-tuning using the `text-embedding-ada-002` embeddings by OpenAI, introduced by [43], was used. With the assumption that paragraphs containing contradicting information concern similar topics, a pre-filtering can be conducted for all paragraphs in the financial report by creating a collection for each paragraph containing paragraphs with high similarities in the embedding space. Possible measurements for these similarities are, for example, the Euclidean distance (low values for high similarities) or cosine similarity (values close to, but below 1 for high similarities). The latter was implemented within this work. With the obtained embeddings, so-called buckets – collections of similar paragraphs – can be created. In the following, we describe three different approaches one could take to accomplish this bucketing process.

3.2.1 Fixed-Length Buckets

To create buckets of fixed length $k + 1 = \lambda$, the k nearest paragraphs are selected according to the cosine similarity inferred from the embeddings. This reduces the number of queries to $N_{\text{paragraphs}}$, compared to the baseline approach with paragraph-to-paragraph comparisons. For $\lambda < \frac{N_{\text{paragraphs}}}{2}$, this also results in less combined input into the LLM. For an λ of order 10, this significantly reduces the costs for commercial LLMs, such as OpenAI's GPT-4, where the API call cost is based on input and completion tokens.

3.2.2 Variable-Length Buckets

As the fixed-length buckets can contain very similar paragraphs, the variable-length buckets approach tries to further reduce the number of queries necessary. The term variable-length buckets describes the method of creating buckets not by a fixed length of k nearest paragraphs but by setting an embedding similarity threshold ζ below which paragraphs will not be considered for the bucket. With the `text-embedding-ada-002` embeddings from OpenAI that were used, similarity scores calculated with the cosine-similarity function are generally close to one. The macro-average is at 0.79 for all documents. To make sure that the bucket does not grow too large, with respect to the number of different semantically close paragraphs it holds, a maximum bucket size is set, as well as a minimum size, to rule out cases where no paragraphs exceed the embedding similarity threshold. In the following, these will be denoted with Λ and λ for maximum and minimum, respectively. Table 1 shows that the average bucket size can be reduced significantly by setting an embedding similarity threshold of 0.9. However, contradicting paragraphs might be close in the embedding space but could sometimes not exceed the embedding similarity threshold. Thus, this method could lead to some contradictions being filtered out.

3.2.3 Merging Fixed-Length Buckets

The basis for this approach is a collection of buckets created with fixed length. As became apparent during the implementation and with the first test results of the Fixed-Length Bucketing method, the collection of buckets obtained for one financial document often contains multiple buckets that are very similar with respect to the paragraphs they hold. In our data model, paragraphs are identified by a paragraph identifier, a unique integer that is assigned to a paragraph in the document parsing and processing context. This is called the *paragraph id*. If two buckets of fixed length k contain equal paragraph ids, they can be merged to create one bucket with a size $< 2k$. This bucket can then be used to create the query text for both the two initial paragraphs. Two parameters define this method: the bucket similarity threshold, denoted η, which refers to the similarity between buckets based on their stored paragraph identifiers – not based on embeddings but on the overlap of paragraph ids – and the maximum number of paragraph identifiers, denoted as Λ, a bucket can store (including the buckets own id). A bucket b can be thought of as a set of integers, e.g., their ids. Then, the similarity between buckets can be calculated as

$$\text{Similarity}(b_1, b_2) = \frac{|b_1 \cap b_2|}{|b_{1,2}|},$$

whereas $|b_{1,2}|$ is the same for all buckets, corresponding to the fixed-length bucketing done before. A theoretical third hyper-parameter is also present, similar to the variable bucketing, which is the minimum number of paragraphs in a bucket. This is, in our terminology, $\lambda = k + 1$, because the merging takes place after fixed buckets have been calculated. The $k + 1$ comes from the fact that k refers to the nearest neighbors of a paragraph, and the resulting query thus features $k + 1$ paragraphs. A simple example explaining the merging process is shown in Fig. 2, while the overall process is depicted in Fig. 3. The parameter Λ is important because, as mentioned, indefinitely large buckets and, thus, indefinitely

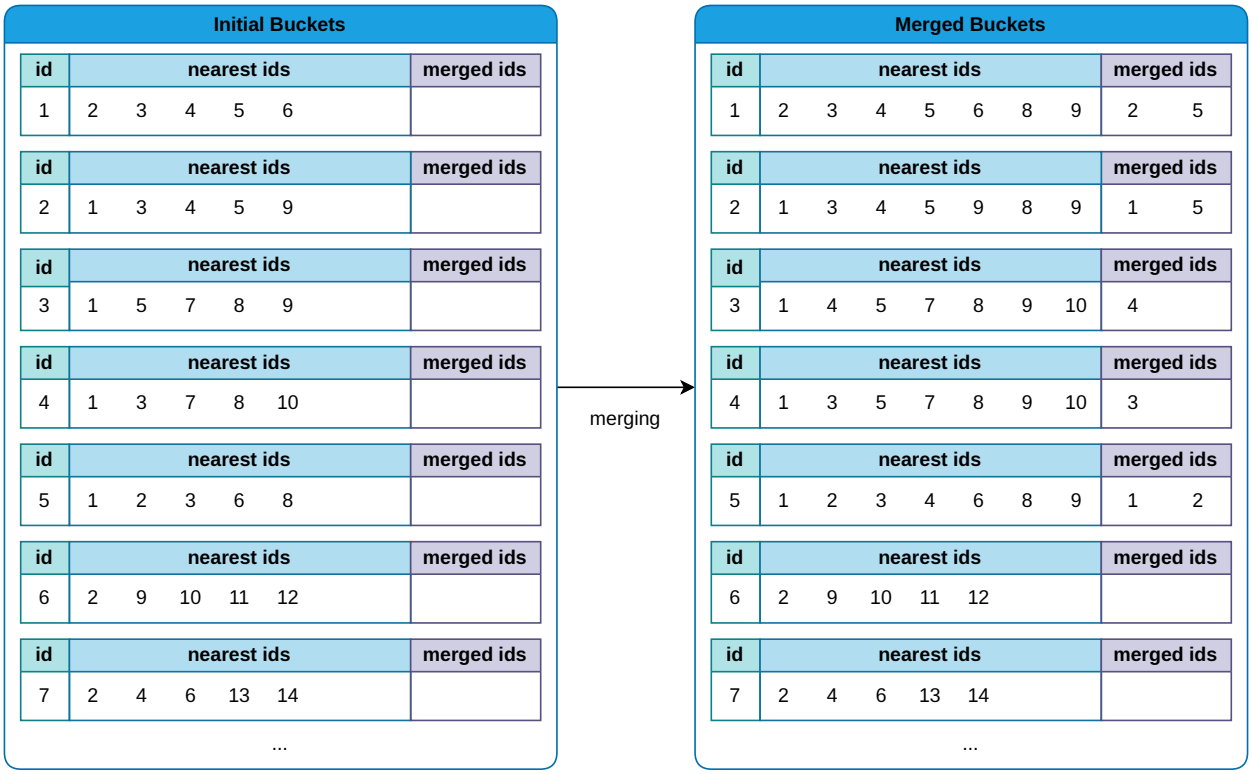

Figure 2: This fictitious example aims to demonstrate the idea of merging. Suppose there is a list of 7+ initial buckets with varying similarities where the `id` is a unique paragraph identifier of each paragraph in a document. Suppose further that the merging hyper-parameters are $\lambda = 6$, $\Lambda = 9$ and $\eta = 0.6$. With the `id` of each bucket plus the `nearest ids`, there are then, after merging, a maximum of $\Lambda = 9$ paragraphs contained in a query for each bucket. Initially, there are 6. Clustering the buckets according to their similarity within their `nearest ids` shows that we group the first bucket with the second, the fifth, and the seventh. However, merging the second, fifth, and seventh all into the first bucket would exceed the maximum number of paragraphs Λ. Thus, it will be only merged with the second and the fifth. The same goes for the second and the fifth buckets. This way, the data model can store the `related ids` and `merged ids` from each bucket. When the list of paragraphs is then iterated later to create the queries, a prompt can be built from the paragraph `id` and the `related ids`. However, if there are `merged ids` indicating that this query has already been made with the same content, the model does not need to be required. In the above example, the merging reduction would lead to four queries instead of the initial seven.

long LLM prompts are undesirable due to the expected loss in accuracy and usability of the LLM's output, as well as the limitation in context size.

Choosing the parameters for merging fixed-length buckets is not a trivial task. The most important constraint is that we are, for the sake of reduced model output and query cost, interested in merging paragraphs as often as possible. On the other hand, we want our queries to be as short as possible to reduce the complexity and noise in the model input. We choose the fixed bucketing method with a $k = 10$ as a base. Then, the number of additional paragraphs a bucket can hold is increased from 0 to 9 in steps of one, such that $\Lambda \in [11, 20]$. Secondly, η is iterated from 0.1 to 1.0 in steps of 0.1, such that buckets for a total of 100 combinations of Λ and η are calculated. The total reduction in the number of queries in percent for all 30 documents can be seen in Fig. 4. As is apparent, the biggest reduction can be achieved for a similarity $\eta = 0.5$ and a $\Lambda = 20$, whereas the latter is to be expected, as a larger maximum number of paragraphs to be stored in a bucket trivially allows for more mergers.

However, for our final evaluation, we chose to set the parameters to $\Lambda = 17$ and $\eta = 0.6$ for two reasons. First, this was considered a compromise to achieve a large cost and output size reduction while simultaneously keeping the LLM prompts comparatively short. Second, a higher similarity η might positively affect the model performance, as merged buckets with a higher similarity might be contextually closer.

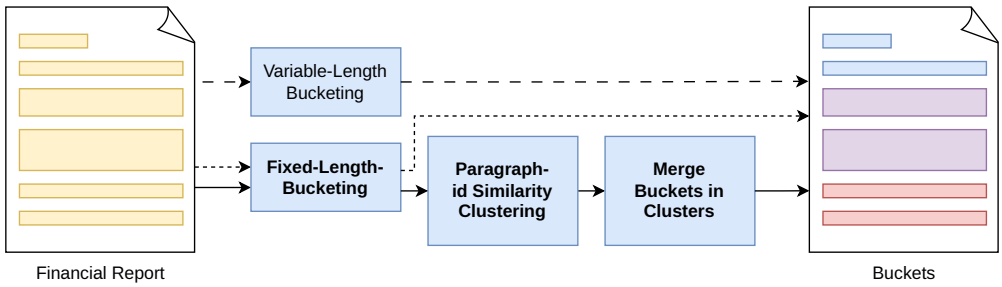

Figure 3: This figure displays the individual steps of the clustering in Fig. 1, if the method of merging buckets is chosen. Additionally, the alternative methods, variable-length bucketing (dashed line) and fixed-length bucketing only (dotted line), are displayed.

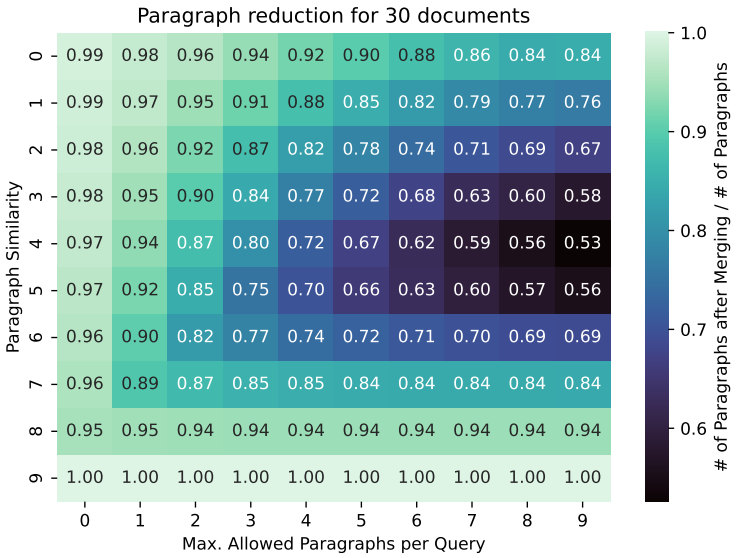

Figure 4: Heatmap to display the effectiveness of Merging Fixed-Length Buckets (see section 3.2.3).

3.2.4 Query Length and Cost Comparison

For a corpus of 30 documents, a cost and query length comparison, in terms of the number of paragraphs featured in the query with a fixed prompt text containing the assignment description, is shown in Table 1. For auditors wanting to use this methodology as a tool for contradiction detection, the subsequent cost to let the LLM calculate its output is, when it comes to commercial models such as GPT, not trivial. Keeping the cost at an affordable and appropriate level is fundamentally important for the respective use case. Thus, we created a comparison of the different methods presented previously with respect to two configurations for the variable and the merging approach. As can be seen, the baseline approach of comparing each two paragraphs in a document is not feasible. An overview of the cost and paragraphs merged can be seen in Fig. 4. The greatest cost reduction can be achieved by merging buckets with parameters $\Lambda = 20$ and $\eta = 0.5$. The second lowest cost is achieved by $\Lambda = 17$ and $\eta = 0.6$, which were chosen as the configurations for the later conducted experiment.

4 Experiments

This section describes the experiments conducted to evaluate the performance of the approaches described in the previous section. The experiments were conducted on three contradiction detection datasets, detailed in the following

Bucketing	Bucketing Parameters	Cost in US$		Query Length
		GPT-3.5	GPT-4	
None	None	2539.28	68471.06	2
Fixed	$k = 10$	29.09	582.01	11
Variable	$\Lambda = 11, \lambda = 3$ $\zeta = 0.85$	26.84	544.51	9.44
Variable	$\Lambda = 11, \lambda = 3$ $\zeta = 0.9$	16.67	403.51	4.97
Merge	$\Lambda = 17, \lambda = 11$ $\eta = 0.6$	**21.13**	**393.38**	**13.61**
Merge	$\Lambda = 20, \lambda = 11$ $\eta = 0.5$	20.02	348.37	15.96

Table 1: Query length and cost comparison for the different paragraph bucketing methods. The bucketing keywords refer to the different sections 3.2.1, 3.2.2, and 3.2.3. There, the bucketing parameters are also explained. The *None* Bucketing refers to no bucketing method being used: each paragraph in a document is paired for a query with every other paragraph in the document. The cost is the estimated maximum cost for the dataset of 30 documents. The LLMs for which the theoretical maximum costs were calculated are GPT-3.5-turbo, denoted GPT-3.5, and GPT-4, using the values from `https://openai.com/pricing`, last accessed 27 October, 2023. The actual query costs can be expected to be a bit lower, as the maximum number of input and output tokens was usually not reached in our experiments. The column *Query Length* displays the macro average of the number of paragraphs that were assigned to each query of the LLM through different bucketing methods. The bold configuration is the one used in the experiment.

subsection. They are an annotated, sentence-pair-level dataset introduced in [21], an annotated, document-level dataset, and an unannotated, document-level dataset. The two latter ones are novel datasets[2] introduced in this work.

The results show that the proposed zero-shot approach achieves outstanding results on all three datasets and is able to surpass a competing, supervised approach introduced in [21] by a considerable margin.

4.1 Dataset 1: Annotated sentence-pair data

4.1.1 Data

The first dataset consists of 640 annotated sentence pairs found in published English financial documents (annual reports). Out of these 640 sentence pairs, 126 belong to the test set and are thus the only part we use for this study, as our approach does not require any dataset-specific fine-tuning. For more information and how this dataset was collected, we refer to the paper [21] introducing it.

4.1.2 Results

In the presented results in Table 2, we compare the performance of three different models on the sentence-level dataset introduced in [21], focusing on precision, recall, and the F1 score. The models evaluated in this study are a fine-tuned XLM-RoBERTa-large [17], the best-performing approach from [21] and, thus, our baseline, GPT-3.5 Turbo, and GPT-4. The models were trained under different conditions, with XLM-RoBERTa-large being supervised on the training and validation set and both GPT-3.5 Turbo and GPT-4 being evaluated in a zero-shot learning scenario. The reported F_1 score reveals that GPT-4 achieves the highest score at 93.24%, emphasizing its overall effectiveness in the task. XLM-RoBERTa-large scores 89.55%, while GPT-3.5 Turbo scores 86.36%.

The results demonstrate that GPT-4, operating in a zero-shot learning setting, outperforms the other models, including the supervised model XLM-RoBERTa-large, in terms of precision, recall, and F1 score. These findings suggest that GPT-4 is a promising candidate for tasks where accurate positive predictions and a high recall rate are critical, showcasing its potential in real-world applications and beating competing models, even if they are fine-tuned on the specific task.

[2]Since the datasets and Python code were developed and used for a confidential project, we cannot release them to the public. This is especially true for the annotated contradictions and evaluation by professional auditors, which are highly sensitive information.

Model	Training	Precision in %	Recall in %	F_1 in %
RoBERTa [21]	Supervised	88.24	90.91	89.55
GPT-3.5 Turbo	Zero-Shot	86.12	86.59	86.36
GPT-4	Zero-Shot	**93.01**	**93.47**	**93.24**

Table 2: Sentence-level results

Dataset	Documents	Contradictions		
		Redundancy-based	Numerical	Other
IFRS	5	12	13	8
HGB	5	6	16	13

Table 3: Inserted Contradictions in Total

4.2 Dataset 2: Annotated document data

4.2.1 Data

The basis for this data is from published reports in the International Financial Reporting Standard (IFRS) as well as the German Commercial Code ("Handelsgesetzbuch", HGB). Five reports were chosen with the criterion that much of the information conveyed in the document is in the form of paragraphs, not tables. The reports were parsed into machine-readable format, assigning each paragraph a paragraph id as a unique identifier. The first method of introducing contradictions into the reports was on the basis of redundancies. If the annotators found redundant information, one of the redundancies was edited to create a contradiction, e.g., "John Doe is Chairman of company C" was edited in one place to read then "John Doe is Janitor of company C". Thus, comparing two paragraphs that before featured redundancies should then yield a contradiction. The second method was based on numerical contradictions, e.g., "The revenue comes from products A (60%) and B (50%)", which is a contradictory statement because percentages should add up to 100%. Numerical contradictions furthermore include, among other things, announcing an enumeration of multiple elements but naming less or more than that, announcing an increase/decrease of values while stating opposite numbers or trends, and errors in simple mathematical calculations. The third kind of contradiction, later denoted as *other*, theoretically requires some degree of common or world knowledge to be detected. Examples include but are not limited to named cities being assigned to the wrong countries or contradictory statements about sustainability, like promoting an increase in energy consumption.

The altered paragraphs and their contradictory counterparts were noted and could thus be directly extracted to present them to the respective LLMs with or without noise.

The Number of contradictions, distributed over the five IFRS and HGB reports each, was about 30, respectively. A detailed overview can be found in Tab. 3.

4.2.2 Results

We use this dataset to investigate the effect of noise on the performance of an LLM, namely GPT-4. Noise, in this case, is defined as adding context to the query irrelevant to the investigated contradiction. To simulate this, we add "paragraph neighbors", i.e., paragraphs appearing right next to the contradiction, to our query. The results of this can be seen in Figure 5.

A first observation is that the Precision of GPT-4 in finding contradictions is astounding; every contradiction found is, in fact, a contradiction. The recall, on the other hand, is more worrisome. Nevertheless, without any noise added, we achieved a remarkable F_1 score of 46.51% and 55.67% on the IFRS and HGB datasets, respectively. Another observation confirmed our expectation: the more we add noise to our query, the more the performance suffers. This affects our choice of the maximum allowed paragraphs during our investigation of the third dataset, as described in Section 3.

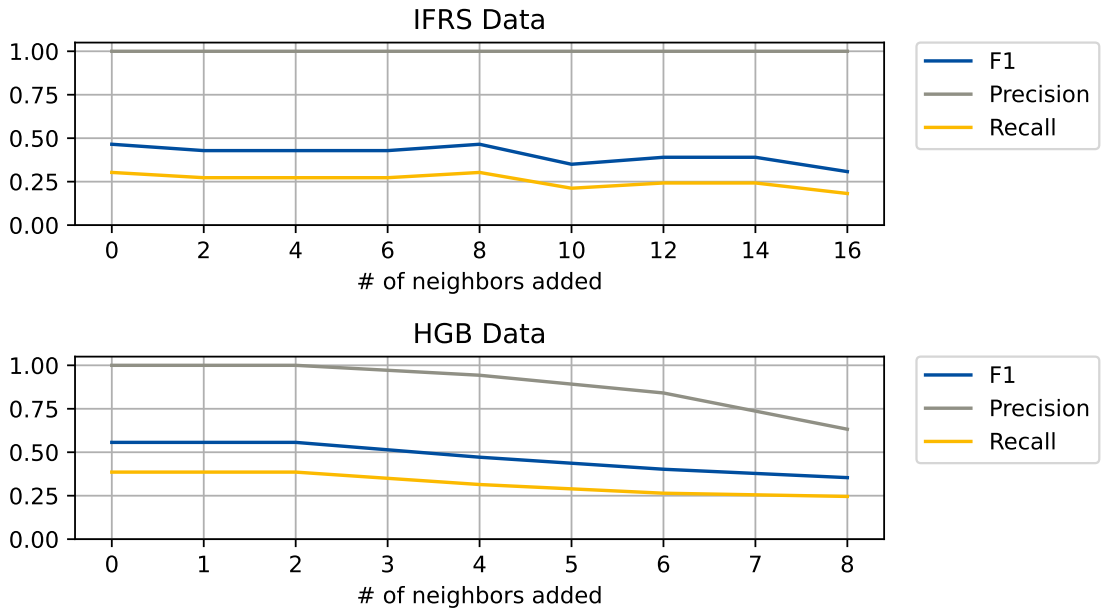

Figure 5: Plot of Contradiction F_1, Precision, and Recall of GPT-4 on Dataset 2 with an increasing number of paragraph neighbors added to the prompt.

Score	Description
1	There is a contradiction in the paragraphs provided in the prompt.
2	There is no contradiction in the paragraphs provided in the prompt, but the formulation in the text is ambiguous and could be interpreted as a contradiction. Ideally, the formulation will be corrected to clear up any ambiguities. The artifact found is therefore relevant for the review.
3	There is no contradiction in the paragraphs provided in the prompt. However, this becomes apparent only in the context of missing information. Therefore, there is no contradiction, but it is justified to report the existence of a contradiction based on the information contained in the prompt.
4	There is no contradiction in the paragraphs provided in the prompt. The existing justification is technically incorrect, but this can be attributed to a lack of specific subject knowledge or faulty interpretations of terms on the part of the LLM.
5	There is no contradiction in the paragraphs provided in the prompt. The justification is also generally unusable, as it makes no sense, is not machine-readable because the output formatting specifications were not followed, or it is missing completely.

Table 4: The ranking system employed by financial auditors when evaluating the output of the language model. This system is only applicable if no annotations exist, i.e., in the case of dataset 3.

4.3 Dataset 3: Unannotated Document data

4.3.1 Data

The third dataset comprises 30 already published and audited German financial reports in the *International Financial Reporting Standard* (IFRS) format. They are unannotated, and thus, to evaluate the results on this dataset, the model outputs were presented to expert financial auditors tasked to rank them with respect to how helpful they are in analyzing the financial report and finding potential contradictions. The ranking system is described in Table 4. The 30 reports in the dataset had, after parsing[3], an average and median of 261.13 and 190.5 of paragraphs per document respectively. Additionally, the number of words in each paragraph is, at the macro-average and -median, 46.16 and 35, respectively.

[3]Among others, individual words and headings were omitted, since in most cases it could not be assumed that contradictions were to be identifiable on the basis of paragraphs with such limited content and context.

Score	Frequency (absolute)	Frequency (in %)
1	23	8.84
2	75	28.85
3	70	26.92
4	50	19.23
5	42	16.15

Table 5: GPT-4 output ranked by professional auditors.

4.3.2 Results

Generally speaking, published and audited financial documents should not contain any contradiction, so finding contradictions within the third dataset should not be possible. Therefore, following that reasoning and looking at Table 4, every contradiction that the model, GPT-4, reported should have been rated a **2** or higher by expert auditors.

However, Table 5 proves that contradictions, in fact, still exist in professionally audited and published financial reports. Surprisingly, almost 9% of contradictions found turned out to be correct hits, and a significant amount (28.85%) is at least ambiguous and could be interpreted as a contradiction.

Furthermore, the auditors tasked with rating the results were overwhelmingly pleased with the provided analysis and are eager to use such a framework in their day-to-day auditing process, which is more of a "soft" metric but nevertheless bears witness to the success of our approach.

One should note that we can only measure the precision, i.e., if a found contradiction is actually a contradiction, and not the recall, i.e., how many contradictions of the total amount of contradictions are found, of our approach. This is due to the unannotated nature of the dataset considered during this task.

5 Conclusion

This paper has demonstrated the potential of using large language models (LLMs) to uncover inconsistencies and contradictions in financial reports. We have proposed a novel approach that leverages the ability of LLMs to learn complex relationships between different parts of a text to identify potential problems and how to solve the combinatory issue of checking a whole document for contradictions (described Section 3.2 and formalized in Equation (1)) by creating an intelligent bucketing system. This system was able to reduce the necessary number of LLM queries by up to 47%. Our approach was evaluated on a real-world dataset of financial reports, and it was shown to be effective in detecting various inconsistencies and contradictions.

The findings of this paper have several important implications for the field of financial reporting. First, they suggest that LLMs can be used to develop new and more effective tools for detecting contradictions and other irregularities. Second, they highlight the importance of considering the potential for inconsistencies and contradictions when interpreting financial reports. Third, they raise questions about the current state of financial reporting practices, as we have multiple contradictions in already published reports, as shown in Section 4.3.2.

We believe this paper represents a significant step forward in developing new and innovative methods for auditing financial reports. We encourage future research to explore the potential of LLMs in this area further and to develop new and improved approaches for detecting and preventing inconsistencies and contradictions in financial reporting. A logical next step would be to test various open-source language models like Llama 2 [44] or Bloom [45]. One could also combine these models with the findings in [46], which stated that injecting noise during the training of transformer models might improve the model's later performance. A clear candidate for this noise would be adding "unrelated" paragraphs to the input as already encountered during our experiments (see Section 4.2.2). Exploring different clustering strategies, such as the application of HDBSCAN [47] or Deep Gaussian mixture models [48], could be undertaken to assess their potential impact on enhancing the efficacy of the introduced bucketing process.

In conclusion, we are convinced that LLMs have the potential to fundamentally change the way financial reporting is done. LLMs can help financial auditors detect contradictions and other irregularities more effectively, lighten the workload of said auditors immensely, and can also help to improve the transparency and accuracy of financial reporting in general. As a consequence, we plan to implement this approach into the financial auditing process of PricewaterhouseCoopers Deutschland, the German arm of one of the largest financial auditing companies in the world.

Acknowledgments

This research has been partially funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr Institute for Machine Learning and Artificial Intelligence.

References

[1] OpenAI. GPT-4 technical report, 2023.

[2] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models, 2023.

[3] Lars Hillebrand, Armin Berger, Tobias Deußer, Tim Dilmaghani, Mohamed Khaled, Bernd Kliem, Rüdiger Loitz, Maren Pielka, David Leonhard, Christian Bauckhage, and Rafet Sifa. Improving zero-shot text matching for financial auditing with large language models. In *Proc. DocEng*, 2023.

[4] Hanchi Gu, Marco Schreyer, Kevin C. Moffitt, and Miklos A. Vasarhelyi. Artificial intelligence co-piloted auditing. *SSRN Electronic Journal*, 2023.

[5] Tobias Deußer, Lars Hillebrand, Christian Bauckhage, and Rafet Sifa. Informed named entity recognition decoding for generative language models, 2023.

[6] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190, 2006.

[7] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proc. ACL*, July 2020. doi:10.18653/v1/2020.acl-main.197.

[8] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *Proc. ICLR*, 2020.

[9] Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. Muppet: Massive multi-task representations with pre-finetuning. In *Proc. EMNLP*, pages 5799–5811, 2021. doi:10.18653/v1/2021.emnlp-main.468.

[10] Abdul Wahab and Rafet Sifa. DIBERT: Dependency injected bidirectional encoder representations from transformers. In *Proc. SSCI*, pages 1–8, 2021. doi:10.1109/SSCI50451.2021.9659898.

[11] Jonathan Pilault, Amine El hattami, and Christopher Pal. Conditionally adaptive multi-task learning: Improving transfer learning in NLP using fewer parameters & less data. In *Proc. ICLR*, 2021.

[12] Maren Pielka, Svetlana Schmidt, Lisa Pucknat, and Rafet Sifa. Towards linguistically informed multi-objective transformer pre-training for natural language inference. In *Proc. ECIR*, pages 553–561, 2023.

[13] Reto Gubelmann, Aikaterini-lida Kalouli, Christina Niklaus, and Siegfried Handschuh. When truth matters - addressing pragmatic categories in natural language inference (NLI) by large language models (LLMs). In *Proc. *SEM*, pages 24–39, 2023.

[14] Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. Negation, contrast and contradiction in text processing. In *Proc. AAAI*, volume 6, pages 755–762, 2006.

[15] Minh Quang Nhat Pham, Minh Le Nguyen, and Akira Shimazu. Using shallow semantic parsing and relation extraction for finding contradiction in text. In *Proc. IJCNLP*, pages 1017–1021, 2013.

[16] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*, 2019. doi:10.18653/v1/N19-1423.

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692*, 2019. doi:10.48550/arXiv.1907.11692.

[18] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. Structbert: Incorporating language structures into pre-training for deep language understanding. In *Proc. ICLR*, 2020.

[19] Ala Eddine Kharrat, Lobna Hlaoua, and Lotfi Ben Romdhane. Contradiction detection approach based on semantic relations and evidence of uncertainty. In *Proc. ICCCI*, 2022.

[20] Shivam Sharma, Mirdul Swarup, Tanushri Mahajan, and Zeel Dilipkumar Patel. Detecting anomalies, contradictions, and contextual analysis through nlp in text. In *Proc. ICICT*, pages 1–5, 2022.

[21] Tobias Deußer, Maren Pielka, Lisa Pucknat, Basil Jacob, Tim Dilmaghani, Mahdis Nourimand, Bernd Kliem, Rüdiger Loitz, Christian Bauckhage, and Rafet Sifa. Contradiction detection in financial reports. In *Proc. NLDL*, 2023. doi:10.7557/18.6799.

[22] Robiert Sepúlveda-Torres, Alba Bonet-Jover, and Estela Saquete. "Here are the rules: Ignore all rules": Automatic contradiction detection in spanish. *Applied Sciences*, 11(7):3060, 2021. doi:10.3390/app11073060.

[23] Robiert Sepúlveda-Torres, Alba Bonet-Jover, and Estela Saquete. Detecting misleading headlines through the automatic recognition of contradiction in spanish. *IEEE Access*, 11:72007–72026, 2023.

[24] Yu Takabatake, Hajime Morita, Daisuke Kawahara, Sadao Kurohashi, Ryuichiro Higashinaka, and Yoshihiro Matsuo. Classification and acquisition of contradictory event pairs using crowdsourcing. In *Proc. Workshop on EVENTS at NAACL-HLT*, pages 99–107, 2015. doi:10.3115/v1/W15-0813.

[25] Zeinab Rahimi and Mehrnoush ShamsFard. Contradiction detection in persian text. *arXiv preprint arXiv:2107.01987*, 2021. doi:10.48550/ARXIV.2107.01987.

[26] Khloud Al Jallad and Nada Ghneim. ArNLI: Arabic natural language inference for entailment and contradiction detection. *Computer Science*, 24(2), 2023.

[27] Chujie Zheng, Jinfeng Zhou, Yinhe Zheng, Libiao Peng, Zhen Guo, Wenquan Wu, Zheng-Yu Niu, Hua Wu, and Minlie Huang. CDConv: A benchmark for contradiction detection in Chinese conversations. In *Proc. EMNLP*, pages 18–29, December 2022.

[28] Rafet Sifa, Maren Pielka, Rajkumar Ramamurthy, Anna Ladi, Lars Hillebrand, and Christian Bauckhage. Towards contradiction detection in german: a translation-driven approach. In *Proc. SSCI*, pages 2497–2505. IEEE, 2019. doi:10.1109/SSCI44817.2019.9003090.

[29] Maren Pielka, Rafet Sifa, Lars Patrick Hillebrand, David Biesner, Rajkumar Ramamurthy, Anna Ladi, and Christian Bauckhage. Tackling contradiction detection in german using machine translation and end-to-end recurrent neural networks. In *Proc. ICPR*, pages 6696–6701, 2021. doi:10.1109/ICPR48806.2021.9413257.

[30] Lisa Pucknat, Maren Pielka, and Rafet Sifa. Detecting contradictions in german text: A comparative study. In *Proc. SSCI*, pages 01–07, 2021. doi:10.1109/SSCI50451.2021.9659881.

[31] Rafet Sifa, Anna Ladi, Maren Pielka, Rajkumar Ramamurthy, Lars Hillebrand, Birgit Kirsch, David Biesner, Robin Stenzel, Thiago Bell, Max Lübbering, et al. Towards automated auditing with machine learning. In *Proc. DocEng*, 2019. doi:10.1145/3342558.3345421.

[32] Rajkumar Ramamurthy, Maren Pielka, Robin Stenzel, Christian Bauckhage, Rafet Sifa, Tim Dilmaghani Khameneh, Ulrich Warning, Bernd Kliem, and Rüdiger Loitz. ALiBERT: improved automated list inspection (ALI) with BERT. In *Proc. DocEng*, pages 1–4, 2021. doi:10.1145/3469096.3474928.

[33] David Biesner, Maren Pielka, Rajkumar Ramamurthy, Tim Dilmaghani, Bernd Kliem, Rüdiger Loitz, and Rafet Sifa. Zero-shot text matching for automated auditing using sentence transformers. In *Proc. ICMLA*, 2022.

[34] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications, 2022.

[35] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[36] Tobias Deußer, Syed Musharraf Ali, Lars Hillebrand, Desiana Nurchalifah, Basil Jacob, Christian Bauckhage, and Rafet Sifa. KPI-EDGAR: A novel dataset and accompanying metric for relation extraction from financial documents. In *Proc. ICMLA*, pages 1654–1659, 2022.

[37] Lars Hillebrand, Tobias Deußer, Tim Dilmaghani, Bernd Kliem, Rüdiger Loitz, Christian Bauckhage, and Rafet Sifa. KPI-BERT: A joint named entity recognition and relation extraction model for financial reports. In *Proc. ICPR*, 2022.

[38] Ali Jabbari, Olivier Sauvage, Hamada Zeine, and Hamza Chergui. A French corpus and annotation schema for named entity recognition and relation extraction of financial news. In *Proc. LREC*, pages 2293–2299, 2020.

[39] Yixuan Cao, Hongwei Li, Ping Luo, and Jiaquan Yao. Towards automatic numerical cross-checking: Extracting formulas from text. In *Proc. WWW*, 2018. doi:10.1145/3178876.3186166.

[40] David Biesner, Rajkumar Ramamurthy, Robin Stenzel, Max Lübbering, Lars Hillebrand, Anna Ladi, Maren Pielka, Robin Stenzel, Rüdiger Loitz, Christian Bauckhage, and Rafet Sifa. Anonymization of german financial documents using neural network-based language models with contextual word representations. *Springer International Journal of Data Science and Analytics*, 2021. doi:10.1007/s41060-021-00285-x.

[41] Marco Schreyer, Timur Sattarov, and Damian Borth. Federated and privacy-preserving learning of accounting data in financial statement audits. In *Proc. ICAIF*, page 105–113, 2022.

[42] Keletso Mabelane, Wilson Tsakane Mongwe, Rendani Mbuvha, and Tshilidzi Marwala. An analysis of local government financial statement audit outcomes in a developing economy using machine learning. *Sustainability*, 15, 2023.

[43] Ryan Greene, Ted Sanders, Lilian Weng, and Arvind Neelakantan. New and improved embedding model, 12 2022. URL `https://openai.com/blog/new-and-improved-embedding-model`. Accessed 27.09.2023.

[44] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

[45] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. BLOOM: A 176b-parameter open-access multilingual language model, 2023.

[46] Tobias Deußer, Cong Zhao, Wolfgang Krämer, David Leonhard, Christian Bauckhage, and Rafet Sifa. Controlled randomness improves the performance of transformer models, 2023.

[47] Ricardo Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In *Proc. PAKDD*, pages 160–172, 2013.

[48] Cinzia Viroli and Geoffrey J. McLachlan. Deep gaussian mixture models. *Statistics and Computing*, 29:43 – 51, 2019.