KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

Retrieval-augmented slot filling for European bond prospectuses

A comparative analysis of retrieval techniques and language models

Martin HALSBERGHE
R0892373
Thomas MORIAUX
R0890486

Thesis submitted to obtain the degree of
MASTER OF BUSINESS ADMINISTRATION
Financial Analysis

Promoter: Prof. Dr. Tarik Roukny
Work Leader: Yanis El Omari
Academic year 2024-2025

Copyright information:

Student paper as part of an academic education and examination.

No correction was made to the paper after examination.

KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

Retrieval-augmented slot filling for European bond prospectuses

A comparative analysis of retrieval techniques and language models

This thesis develops and evaluates retrieval-augmented slot filling (RASF) models for automating template filling of key information from European bond prospectuses. Three objectives are addressed: identifying the most effective information retrieval models, evaluating the most effective slot filling models, and assessing the combined system's ability to satisfy investors' information needs. Five retrieval models were compared: BM25, a dense vector model, and three fusion models that combine the BM25 and dense relevance scores with varying weighting schemes. For slot filling, Gemini 2.5 Flash and Llama 3.1 8B were evaluated in zero-shot, few-shot, and LoRA-tuned settings. Evaluation was conducted on a manually annotated dataset of 20 English-language bond prospectuses. Results show that a fusion model, equally weighting the BM25 and dense relevance scores, achieved the highest overall performance (nDCG@5 of 0.64; F1@5 of 0.47). However, performance varied depending on the type of information need. BM25-dominant models outperform on precisely phrased information needs (such as ISINs), while dense-dominant models excel on context-rich ones (such as ESG commitments). For slot filling, a LoRA-tuned Gemini 2.5 Flash model achieved the highest performance across all information needs (exact match accuracy of 78%; slot error rate of 0.18), outperforming both its zero-shot and few-shot equivalents and the Llama 3.1 8B model. The complete RASF model, combining the best retrieval model per information need with the best slot filling model, achieved an exact match of 64% and a slot error rate of 0.22. Of the total slot errors made, 38% is attributable to the retrieval pipeline, mainly due to query misalignment. In comparison, 62% is caused by the slot filling pipeline, with template complexity being a key contributor. Despite encouraging results, further refinement of both pipelines is needed to fully automate prospectus processing.

Acknowledgements

This thesis used generative AI to write code, brainstorm ideas, find academic studies, and improve the readability of this thesis. The authors have critically evaluated every idea and content before being adopted. Work generated by AI is clearly labelled.

Table of Contents

Introduction

Prospectuses are legally mandated documents issued during public offerings. By offering extensive information about the issued security and issuer, they are aimed at guiding investment decisions (European Commission, n.d.). As a result, prospectuses can be seen as a tool to reduce information asymmetry between potential investors and the issuer. In the European Union, content requirements are defined by the Prospectus Regulation (Regulation (EU) 2017/1129, 2017).

Despite these regulatory requirements, prospectuses remain lengthy and unstandardized. An analysis of the ESMA (2022) found these documents to have an average of 200 pages, with outliers exceeding 1000 pages. Additionally, issuers maintain great freedom in the structure and formulation of phrases. Both factors hinder an average investor from efficiently extracting key information. Consequently, information asymmetry still exists.

This thesis aims to develop and evaluate various retrieval-augmented slot filling models for automated template filling of European prospectuses. Specifically, it aims to answer three questions: (i) which information retrieval models perform best at retrieving relevant passages in prospectuses, (ii) which slot filling models perform best in populating a predefined template, and (iii) whether the combined system satisfies the information needs of investors. The scope is limited to English-language bond prospectuses involving the issuance of a single bond.

To achieve this, multiple information retrieval and slot filling models are developed using established techniques, benchmarked against each other and tested for statistically significant differences. The retrieval-augmented slot filling model is constructed using a two-stage pipeline consisting of an information retrieval model, retrieving relevant passages, and a slot filling model that populates a predefined template based on these passages. This model is then evaluated to the extent it satisfies an information need. An attribution analysis will be performed to determine whether performance limitations arise from the retrieval or slot filling stage.

With this research, this thesis provides an overview of different retrieval-augmented slot filling approaches and develops a model capable of transforming legal-financial documents into accessible, structured data. By doing so, it contributes to bridging the gap between dense, unstandardized prospectuses and investors' needs.

1 Literature review

This chapter reviews the literature on information retrieval and language models used for template filling of prospectuses. First, core concepts of template filling are introduced. This is followed by a comparison of information retrieval techniques and language models for template filling. Finally, metrics are presented to assess the performance of both information retrieval and slot filling models.

1.1 Core concepts of template filling

Natural language processing (NLP) is the use of algorithms to process human language, like found in a prospectus (Jurafsky & Martin, 2025). Information extraction (IE) is an NLP task that converts unstructured information into structured data. One of the applications of IE is template or slot filling.

However, information extraction is computationally expensive. Consequently, when employed on large documents like prospectuses, its application must be limited to the most relevant sections. Information retrieval is the process of identifying these relevant passages. In the following two sections, information retrieval and slot filling will be covered in great depth.

1.2 Information retrieval

Information retrieval (IR) is formally defined as the process of finding all documents that satisfy an information need from a specific collection (Manning et al., 2008). Based on this definition, three components can be identified: an information need, a collection of documents, and a process for finding documents to satisfy this information need. The first component must be represented as a set of terms called a "query" (Jurafsky & Martin, 2025). In IR, "documents" are units of text treated as a single entity. This term must not be confused with the everyday meaning of the word. Following this definition, a prospectus, split into multiple units of text, consists of multiple documents. All documents subject to retrieval are collectively referred to as the "collection". Ultimately, a process is required to retrieve documents that satisfy a given query. Before this can be done, however, text from the collection and query must first be preprocessed.

1.2.1 Text preprocessing and chunking

Text preprocessing is the conversion of a raw text into linguistically meaningful units in a machine-readable way (Indurkhya & Damerau, 2010). It encompasses a broad set of tasks, of which tokenization is most important. Tokenization refers to breaking down text into smaller units, known as tokens. Advanced NLP models, such as BERT, utilize sub-word tokenizers, breaking down text into units beyond the word level, while simpler models, like BM25, work best with word-level tokenizers (Song et al., 2021).

Following tokenization, the tokenized text must be segmented into documents or chunks. This process, referred to as chunking, involves two decisions: the chunk size and strategy. Based on limited research, the optimal size seems to depend on the nature of the data (Bhat et al., 2025; Juvekar & Purwar, 2024; Yepes et al., 2024). Bhat et al. (2025) found short chunks (64-128 tokens) to perform better for documents with brief factual information, whereas larger chunks (512-1024 tokens) were superior when broader conceptual understanding was needed. Juvekar and Purwar (2024) found that larger chunks to work best for legal documents.

The second decision refers to using fixed-length or semantic chunking. Qu et al. (2024) and Yepes et al. (2024) show the latter to outperform the former. While preferred, however, semantic chunking is not always feasible to implement. To mitigate the risk of fixed-length chunking splitting coherent texts mid-paragraph, overlap between chunks can be added (Thattantavida & Sarkar, 2025).

1.2.2 Model paradigms

Within IR, different model paradigms exist, each with its own strengths and weaknesses. This review discusses the lexical, semantic, and hybrid model types.

1.2.2.1 Lexical models

Lexical or keyword-based models rank document relevance by explicitly matching document terms with the query terms (Jurafsky & Martin, 2025). An essential requirement is the exact match between terms. This leads to a vocabulary mismatch problem where semantically identical, but lexically different terms will not be considered a match (such as "automobile" and "car") (Furnas et al., 1987). The most used algorithm is Okapi BM25 (Robertson & Zaragoza, 2009). Introduced by Robertson et al. (1995), BM25 is a ranking function estimating a document's relevance based on (1) how often query terms occur in a document (term frequency), (2) how rare the query terms are in the collection (inverse document frequency), and (3) the length of the document. Documents are ranked higher if they have a high term frequency, especially if they include uncommon terms and are not excessively long.

Thakur et al. (2021) demonstrate that BM25 outperformed 75% of the semantic ranking functions evaluated in their study on nDCG@10[1] when no fine-tuning is performed. In addition, BM25 is computationally more efficient than its semantic counterpart. Only the reranking model, which combines BM25 with a semantic model, showed an 11% outperformance in nDCG@10 compared to BM25.

1.2.2.2 Semantic models

Semantic models solve the vocabulary mismatch problem by capturing semantically related words. This is implemented by representing documents and queries as vectors, a process called embedding (Jurafsky & Martin, 2025). To illustrate this, consider a document vector d and a query vector q with z unique terms occurring in the document and query.

[1] nDCG@10 measures the relevance and rank quality of top 10 retrieved documents by an IR model. It lies between 1 (perfect ranking) and 0 (no relevant documents). A complete discussion of performance metrics can be found in Section 1.4.

This results in the vectors below with w_z the weight of term z, such as the frequency count. A document's relevance is evaluated based on a distance measure, such as cosine similarity, between its vector and the query vector.

$$d_n = (w_{1,n}, w_{2,n}, ..., w_{z,n}) \tag{1.1}$$

$$q = (w_{1,q}, w_{2,q}, ..., w_{z,q}) \tag{1.2}$$

With the advent of the Transformer architecture, context-aware embedding (such as BERT-based embedding), as opposed to static embedding (such as Word2Vec), further spurred semantic understanding by representing a term based on the surrounding terms (Vaswani et al., 2017).

1.2.2.3 Hybrid models

The final paradigm combines the previous two paradigms. Hybrid models exist in two forms: reranking models and fusion models. Reranking models create an initial ranking using an efficient retriever (like BM25) and then apply a sophisticated retriever (like a BERT-based retriever) to a subset of the top-ranked documents (Nogueira et al., 2019). Fusion models, on the other hand, combine the scores or ranks of both models (Bruch et al., 2023). As Thakur et al. (2021) comparison highlighted, reranking models show an outperformance compared to both models on a standalone basis. Also, fusion models confirm their superior performance in the studies of Kuzi et al. (2020) and Bruch et al. (2023) with an nDCG@1000 of 0.454 compared to 0.309 for BM25 and 0.441 for the semantic model on the MS MARCO benchmark. These results demonstrate that combining the strength of both lexical and semantic retrieval can lead to promising outcomes.

1.3 Slot filling using LMs

Texts frequently describe stereotypical situations. Template or slot filling (SF) is the task of identifying a set of predefined attributes for such a situation by extracting their values from a text (Jurafsky & Martin, 2025). Each attribute is considered a "slot" and is filled with its corresponding value(s) derived from the text. For example, in the case of a bond's coupon rate structure (stereotypical situation), the slots could include the coupon rate, the type of coupon rate, and the day count convention.

The field of information extraction is limited to extracting values verbatim from an input text (Mehri & Eskenazi, 2021). However, due to the complexity of prospectuses, it can sometimes still be helpful to generate new values to fill slots. As a result, this thesis will adopt a hybrid approach that combines both information extraction and classification to populate template slots.

1.3.1 Transformer-based LMs

The most advanced SF models are built on Transformer-based language models. Language models (LMs) aim to predict the probability of future or missing tokens in a sequence of tokens (Zhao et al., 2025). LMs must be trained on large datasets and have a high number of parameters to achieve significant performance improvements. Large language models (LLMs) refer to LMs pre-trained on massive amounts of data with billions of parameters.

In practice, LLMs will always be favored over self-trained models on limited data due to their superior performance (Jurafsky & Martin, 2025). To accommodate task-specific needs, LLMs can be further trained with a limited amount of task-specific data, a process called fine-tuning.

The advent of the Transformer architecture spurred the creation of the most advanced LLMs such as BERT, GPT, and T5. Transformer-based LLMs show a remarkable performance in IE tasks, even without fine-tuning. Introduced by Vaswani et al. (2017) in their landmark paper, the Transformer architecture relies on the self-attention mechanism. This mechanism represents and generates tokens in a sequence based on its surrounding and preceding tokens. Self-attention, however, was not unique to Transformers. Instead, the innovation came from "eschewing recurrence [a mechanism used by an earlier architecture] and instead relying *entirely* on an attention mechanism".

The original Transformer architecture consists of two components: an encoder and a decoder (Vaswani et al., 2017). The encoder processes and understands the input sequence. The decoder generates an output sequence using this representation. This results in three LM types: encoder-only, decoder-only, and encoder-decoder. Their architectures and performance in SF will be discussed in the following two sections.

1.3.1.1 Different types

Encoder-only LMs consist of only the encoder of the Transformer. A Transformer-based encoder represents each token of an input sequence based on all the surrounding tokens (full self-attention). The core strength of such models lies in their deep understanding of input text due to this mechanism (Devlin et al., 2019). The most well-known encoder-only LM is Bidirectional Encoder Representations from Transformers (BERT).

Decoder-only LMs utilize only the decoder component. A Transformer-based decoder generates an output sequence by autoregressively generating each token based on the previously generated tokens, a process known as masked self-attention (Jurafsky & Martin, 2025). The initial token sequence provided as input to a decoder is referred to as the prompt. Without an encoder, input text, such as a document, is appended to the prompt. Decoder-only LMs, like the Generative Pre-trained Transformer (GPT), excel in generating new coherent text (Radford et al., 2018).

Encoder-decoder LMs, like the original Transformer, combine both encoder and decoder components. This allows both a deep conceptual understanding of tokens and the ability to generate new text based on this understanding (Vaswani et al., 2017). After encoding, the decoder generates an output sequence by simultaneously considering the previously generated tokens (masked self-attention) and the encoder's representation of the input. This latter process is known as cross-attention. Due to the combination of both components, these models excel in tasks like language translation. A prominent family of models in this field is T5 (Raffel et al., 2020).

1.3.1.2 Performance comparison

Encoder-only LMs generally lead to superior performance in pure extractive tasks where a deep understanding of the input text is required (Lu & Huo, 2025; Sharkey & Treleaven, 2024). Indeed, encoder-only LLMs like BERT were trained with this explicit goal in mind (Devlin et al., 2019). Aguirre et al. (2025) found, however, that a small decoder-only

LLM (Llama-3-8B) fine-tuned with only 10% of the training data outperformed a fully fine-tuned BERT model on SF. The most significant limitation of encoder-only LMs for this thesis is their inability to generate new text due to the absence of a decoder. This limitation, in addition to requiring substantial training before being used, makes them unsuitable for prospectus SF.

When a high amount of training data is available, encoder-decoder models outperform decoder-only models in SF tasks (Andreas et al., 2021; Fatemi & Hu, 2023). This finding is consistent with the architectural advantage of an encoder, which allows for a deeper understanding of input text, a capability that a decoder-only model lacks. However, due to these two components, encoder-decoder models require more computational resources to operate. Additionally, like the encoder-only model, a vast amount of training data is required for fine-tuning the encoder part. Without fine-tuning, decoder-only models outperform encoder-decoder models (Wang et al., 2022).

1.3.2 Decoder-only LLMs and fine-tuning

Given the limited amount of labelled prospectuses data and the need to generate new tokens, the literature suggests decoder-only LLMs are most appropriate for prospectus SF. Decoder-only LLMs are adapted for the task of SF by providing, within the prompt, both (i) explicit instructions on the task and (ii) the relevant prospectus text on which to perform the task.

Decoder-only LLMs applied in SF can vary in two aspects: their scale and post-training regime. The scale is usually measured using the parameter count of a model (Zhao et al., 2025). While a higher parameter count usually leads to a better performance, it also necessitates higher computational resources to both train and run the pre-trained model.

The post-training regime refers to the extent to which fine-tuning is performed. This allows a pre-trained LLM to be optimized for a specific task, like the SF of a prospectus. It ranges from full fine-tuning, where all model parameters are updated with additional training data, to zero-shot learning, where an LLM is used without additional training. Between these two extremes lie in-context learning and LoRA fine-tuning.

In-context learning (ICL) refers to giving multiple examples within a prompt on how an LLM should answer (Dong et al., 2024). Unlike fine-tuning, it does not involve changing model parameters, making it an easy and cost-effective implementation. Brown et al. (2020) show that ICL improves its performance on specific tasks (e.g., completion tasks and question-answering) compared to a zero-shot LLM, while on others, it achieves on-par or even slightly worse results (such as language inference).

While ICL can show its merit, Mosbach et al. (2023), show that generally fine-tuned models with limited data are still preferred over ICL LLMs. However, full fine-tuning is computationally expensive. This is where LoRA becomes valuable. Low-Rank Adaptation (LoRA) is a parameter efficient fine-tuning method that freezes the existing LM weights and only trains a small number of added low-rank parameters (Hu et al., 2021). The original study showed that a LoRA-fine-tuned GPT-3 version resulted in on par performance with a thoroughly fine-tuned GPT-3 on various NLP tasks. This, while needing 3 times lower GPU memory. It also discovered that with limited training examples, as few as 100, LoRA remains useful and showed superior performance compared to full fine-tuning. Consequently, LoRA is particularly suitable when fine-tuning an LLM using limited prospectus data.

1.4 Performance metrics

IR metrics. Recall and Precision are the most fundamental metrics looking at the relevance of a retrieval result given an information need (Manning et al., 2008). Recall is the fraction of relevant documents that are retrieved. Precision is the fraction of retrieved documents that are relevant. These two metrics can be combined by taking the weighted harmonic mean of both (F-measure). The relevance of a metric depends on the use case of the retrieved documents. In this study, they serve as input for a slot filling model where the number of relevant documents is more important than their ranking. As a result, the Mean Reciprocal Rank (MRR), which only looks at the ranking of the first relevant document, is considered less important.

The above-stated metrics only consider the relevance of all documents without considering their ranking (Recall, Precision and F-measure) or only the ranking of the first relevant document (MRR). Moreover, they only consider the binary relevance of retrieved documents. In practice, however, there are multiple levels of relevance for retrieved documents. The normalized Discounted Cumulative Gain (nDCG) solves both limitations by considering both the multi-level relevance and ranking of all retrieved documents. It does this by computing the logarithmically rank-discounted sum of the relevance score of all retrieved documents (DCG) and normalizing this sum by the maximum obtainable score (ideal DCG) (Manning et al., 2008).

Both Recall and nDCG require knowing the relevance of all documents in a collection. An assessment of a whole collection is, however, often infeasible. Pooling addresses this by only considering a subset of top-ranked retrieved documents from multiple retrieval models (Manning et al., 2008). All relevant documents within this subset are then assumed to be the total pool of relevant documents. While pooling leads to an underestimation of the true Recall and nDCG, it still allows for a fair comparison between retrieval models.

SF metrics. In SF, *predicted* slot-value pairs are compared against *expected* slot-value pairs (golden slots). Each pair is exclusively categorized as either: correct if the pair is present in both sets with matching values, substitution error if present in both sets but with mismatching values, insertion error if only present in the predictions set or deletion error if only present in the golden set (Makhoul et al., 1999). Precision is the fraction of predicted slot-value pairs that are correct. Recall is the fraction of golden slots that are correctly predicted. Makhoul et al. (1999) show the F1 measure within SF overweigh substitution errors compared to insertion and deletion errors. Assuming all errors to be equal, using F1 leads to an unfair view. They propose the Slot Error Rate (SER) as an alternative, which is the fraction of golden slots that are filled erroneously (regardless of their type). The formulas for all these metrics are outlined in Section 2.5.4.

Statistical significance. Urbano et al. (2019) show that for testing the significance of IR metrics, the Student's t-test and permutation test have Type I and Type II error rates that most closely align with reality. They observed this by controlling the ground truth, thereby knowing if the null or alternative hypothesis held. Indeed, Smucker et al. (2007) also confirm these two tests as being the most robust. Theoretically, however, the t-test violates the normality assumption. Many IR measures are asymmetrically distributed or bounded between 0 and 1. In contrast, the permutation test makes no distributional assumptions. It is thus preferred on both empirical and theoretical grounds. The sole assumption of the test is the exchangeability of the observed data under the null hypothesis. Given that the normality assumption can also be questioned for SF metrics, the permutation test seems also here to be the most appropriate.

2 Methodology

This chapter discusses the data collection, preprocessing of prospectuses, the definition of information needs of investors, and construction and performance evaluation of all models. All operations were performed in Python.

2.1 Data collection

Prospectus collection. A total of 40 prospectuses were downloaded from the ESMA prospectus register. The earliest available at the time were selected. Twenty prospectuses were allocated to the evaluation of both the IR and SF models. This ensures sufficient diversity to assess performance while balancing the high cost of manual annotation. The remaining twenty were used for fine-tuning the SF model and optimizing the IR query. Preliminary testing revealed an increase in complexity for prospectuses that introduce multiple bonds within a single prospectus. As this study aims to establish a proof of concept, it is limited to single-bond prospectuses, focusing on the models' baseline performance.

Retrieval relevance. The relevance of each top k retrieved document, given an information need, was evaluated on a discrete scale of 0, 1, or 2. Documents with no useful information were given a grade of zero. Two was assigned when a document directly addressed the information need. Partial credit was given if it did not directly address the need but still contained some useful information. For example, some documents addressed only the used reference rate for the floating coupon (score 1), while others also included the added spread (score 2).

Golden slots. For evaluating SF models, expected slot-value pairs were constructed by human judgment of the provided chunks to the SF model. These serve to judge the performance of SF models in isolation. Additionally, the golden slots based on the entire prospectus were compiled. These will form the basis to judge whether a two-stage pipeline, combining the IR and SF model, satisfies an investor's information need.

2.2 Text preprocessing

The text preprocessing was done in multiple steps. First, the PDF prospectus was converted into a string using the library PDFPlumber to make it machine-readable. Second, the string was split up using a fixed-token limit. This limit is set at 512 tokens, as the literature has shown that larger chunks are superior for complex, legally dense documents like prospectuses. Ideally, semantic chunking would be used. However, this is not feasible, as the PDF structure is lost during conversion to a string and due to the variability in structure across different prospectuses. To mitigate context loss arising from the fixed-sized chunks, a 100-token overlap is used between chunks.

For keyword-based IR model, tokenization of the collection and query is performed on a word level using spaCy's `en_core_web_sm` model (https://spacy.io/models/en#en_core_web_sm). The model's performance is closely aligned with other word-level tokenizers and is easy to implement in Python (Meaney et al., 2023). For the semantic IR model, the built-in tokenizer and embedding model of the chosen IR model are used, which will be covered in Section 2.4.1. For the LMs, the built-in tokenizer is also used.

2.3 Defining information needs

Based on the analysis of 20 fine-tuning prospectuses, 17 recurring features were observed, categorized in three groups (see Table 2.1). For evaluation, six information needs will be considered. While these six information needs do not cover all critical bond features, the goal is to assess the overall performance of IR and SF models. The evaluated features were chosen based on diversity for both tasks. For example, for *presence of a put option*, while relevant for an investment decision, is closely related to the *presence of a call option* and thus excluded. These six information needs will then be translated into queries (IR) and template schemas (SF). This is covered in the following sections. The complete list of features can be found in Appendix A.

Table 2.1
Selection of six recurring features in bond prospectuses

Category	Description	Feature
Explicit features	Features being explicitly stated	ISIN
		Coupon rate
Inference features	Features being implicitly stated	Coupon periodicity
		Presence call option
Summary features	Features requiring summarization and understanding of longer lines of text	ESG commitment
		Use of proceeds

2.4 Information retrieval

2.4.1 Construction models

In total, five different models are constructed, of which three are fusion models. The complete source code for these is available at: https://github.com/martinhlss/RASF.

Lexical model. The first model uses Okapi BM25 as a ranking function. This model is considered the baseline due to its robust performance compared to embedding models. Keyword-based models focus exclusively on explicit term matches. Nevertheless, as prospectuses often use explicit terms, like "issue price", this approach is still useful. The resulting BM25 scores of each document represent its relevance to a given query.

Semantic model. To allow semantically similar but lexically different terms to still be matched, a semantic model is also considered. This model is comprised of an embedding model and a vector similarity calculation. The embedding is performed using the BGE-M3 model (https://huggingface.co/BAAI/bge-m3). BGE-M3 was used due to its strong performance with limited memory usage (Hugging Face, n.d.-b). Ideally, a token-level embedding model would be used, such as BERT, as opposed to a document-level model using BGE-M3. However, due to computational limits, this was not feasible. Given the vectorized documents and query, a retrieval score is determined by calculating the cosine similarity between each document's vector and the query's vector. Part of this model implementation is based on code from Nguyen (2024).

Fusion model. Finally, a combination of the previous two models is created. This is implemented by taking a weighted sum of the min-max normalized scores of the previous

two models. The rationale for this model lies in the combination of the strengths of both models and its outperformance of the previous two models, according to the literature review. Three variants are considered: equally weighted scores ("Fusion-EQ"), 75% embedding and 25% BM25 ("Fusion-EM"), and vice versa ("Fusion-BM").

2.4.2 Query formulation

To translate each information need into the query that best represents it, limited testing will be performed to determine the most effective formulation. This will be done by comparing the Precision@3 of various queries based on a subset of five prospectuses of the fine-tuning data for both the lexical and semantic models. Only Precision will be considered for simplicity, as it does not require knowing the set of all relevant documents in the collection.

2.4.3 Performance evaluation

nDCG as preferred measure. For comparing the performance of the different IR models, nDCG will be focused on as it accounts for multi-level relevance and considers both the ranking and relevance of each document. The F1 score is considered the second most important metric as it considers both Recall and Precision equally. While MRR will also be reported, it is considered less important as it only reflects the ranking of the first relevant document, which is not critical for slot filling.

Top 5. Only the top 5 retrieved documents will be considered. While modern language model prompts allow a substantially higher number of documents to be provided, the time required for human grading necessitates limiting this number to five. Indeed, with five models, six information needs, and twenty collections, this results in 3,000 retrieved documents to be evaluated for this section alone.

Pooling. Using pooling, an estimation will be made of the set of relevant documents for a given prospectus (collection) and information need. This is achieved by considering all evaluated documents of all evaluated models. Consider IR model A, which has the following binary relevance assessment of its top 3 retrieved documents: [1, 1, 0]. Assuming IR model B retrieved two other documents that were also relevant, the total relevance pool equals 4 documents. Consequently, the Recall@3 of model A would be 2/4.

Statistical test. To test for differences in nDCG between models, the permutation test with 10,000 random shuffles will be used. Initially, an analysis will determine whether at least one model exhibits a different mean nDCG across all information needs (overall outperformance) and within each information needs (information need-specific outperformance). This is done using a one-way ANOVA test. If the alternative hypothesis is corroborated, pairwise permutation tests will be performed to consider the statistical outperformance of specific models.

Exchangeability assumption. A permutation test constructs the null distribution of the test statistic by swapping the observations (nDCG scores) across groups (IR models). Within this analysis, it would also involve swapping nDCG scores across information needs and prospectuses. However, this would violate the exchangeability assumptions as the nDCG scores are inherently different across information needs and prospectuses regardless of the used IR model. Indeed, some prospectuses and information needs are more complex than others. Therefore, the permutations are restricted to only occur within the same information needs and prospectus.

Adjustment p-values. As multiple hypotheses are tested on the same dataset, p-values are adjusted to account for the inflated probability of making a Type I error. This is achieved by maintaining a false discovery rate (FDR) of 5% using the Benjamini-Hochberg (BH) procedure. The FDR is kept at 5% rather than the family-wise error rate (FWER) as some Type I errors are accepted to reduce Type II errors. The BH procedure will first be applied to all p-values resulting from the ANOVA test. Afterwards, the BH procedure will be applied to all p-values resulting from the pairwise testing for all information needs where at least one statistically significant IR model was identified.

2.5 Slot filling

2.5.1 Template construction

To perform SF, a clear template must first be defined for each information need. For each need, this is achieved by identifying the relevant slots based on the most commonly occurring features identified through analysis of the 20 fine-tuning prospectuses. Based on this, a unique schema is constructed that satisfies three criteria. First, the schema captures all relevant facets, avoiding oversimplifying the information needed. Second, it is be able to accommodate every possible scenario in a prospectus. Third, the template is unambiguously formulated such that it is applied consistently in the same scenarios.

For *coupon rate*, this resulted in the schema shown in Table 2.2. The template allows multiple entries, repeating the same slot, if the bond has a switchable-rate structure (e.g., fixed-to-floating). Each entry represents a group of slot-value pairs corresponding to one phase in the rate structure. The schemas for the other information needs can be found in Appendix B.

Table 2.2
Template schema of 'coupon rate'

Slot name	Slot value
Type	– Fixed: fixed coupon rate parts – Floating: floating coupon rate parts – Other: non-standard coupon rate parts
Value	– Fixed: <X.XXX> (e.g., 3.500) – Floating: <reference rate name>, <spread> (e.g., 1-month LIBOR, 2.000) – null: other rate types
Phase	– Constant: unchanged coupon structure – Initial: first phase in changing coupon structure – Follow-up: subsequent phase in changing coupon structure – Unspecified: coupon structure unclear or missing
Exceptions	– Unspecified coupon rates: not clearly stated – Incomplete coupon rates: underspecified or essential details omitted

It must be recognized that the schemas for most information needs are considered complex for the domain of SF. Indeed, SF, as part of IE, is strictly speaking limited to extracting verbatim information from text. These schemes, conversely, often require generating new values for classification (such as coupon type). However, the complexity is guided by the information needs of investors. As a result, the complexity is a prerequisite to satisfy this need.

2.5.2 Construction models

SF models are constructed using a decoder-only LM that gets provided (i) the context on which to apply slot filling and (ii) instructions on how to do this slot filling. The first component will be the top 5 retrieved documents from the best IR model for each information need. The second component will be the translation of the template schema into prompts, discussed in the following section.

To evaluate the best decoder-only LM, both a large-scale and a small-scale LLM are chosen. For the large-scale LLM, Gemini 2.5 Flash is chosen as it allows for fine-tuning using LoRA and offers a free allowance. Llama 3.1 8B Instruct is used as a small-scale LLM. Compared to other open-source models of a similar parameter count, it performs best in executing rules (Sun et al., 2024). All models will be evaluated with a temperature fixed at zero to ensure a deterministic output.

Both LLMs are evaluated in three settings: zero-shot, few-shot using in-context learning, and LoRA fine-tuned. A zero-shot LLM is provided with instructions without any input-output examples. Few-shot LLMs contain five input-output examples that are a short fictional prospectus snippet, along with the desired output for challenging situations.

Finally, LoRA fine-tuned LLMs are constructed using 20 fine-tuning prospectuses for six information needs, resulting in 120 examples. Of these, 102 (85%) are used for training, and 18 (15%) are used for validation. To determine the optimal number of training cycles (epochs), the cross-entropy loss on the validation dataset is used as evaluation metric. LoRA fine-tuning of Llama is done using Hugging Face's Python libraries. For Gemini, this is done via the Vertex AI Platform. Further details on the parameters used for fine-tuning can be found in Appendix C. The SF model based on Llama is available at https://github.com/martinhlss/RASF.

2.5.3 Prompt construction

Decoder-only models will be provided instructions on how to perform slot filling (system prompt) and the context on which this must be performed (user prompt). The latter is achieved by concatenating the top 5 retrieved documents with three dots (. . .) to highlight that they come from the same prospectus.

Instructions are the translation of the template schema into the output requirements of the LLM. To optimize these instructions, different prompt components, provided by the documentation from Google (n.d.), are used. These components include a persona, objective, output format, and few-shot examples. The implementation for *ISIN* is shown in Figure 2.1. The prompts were constructed using generative AI based on the template schema and further refined by hand. All complete system prompts can be found in Appendix B.

During preliminary testing, it was found that adding a constraint prevented the LLM from returning ISINs not belonging to the newly issued bond. Additionally, it was clarified that the same triplet can never be repeated, as was sometimes done by the Llama LLM.

Figure 2.1

Example of system prompt for ISIN information need

```
<PERSONA>
You are an information extraction assistant.
</PERSONA>

<OBJECTIVE>
Find all ISIN values that . . . identify the newly issued bond(s) described
in the context.
</OBJECTIVE>

<CONSTRAINT>
Exclude ISINs of any bonds mentioned only as references . . .
</CONSTRAINT>

<OUTPUT FORMAT>
Return each ISIN in the format:
VALUE1;VALUE2;VALUE3

Where:
   - VALUE1 is the ISIN code (e.g., US1234567890)
   - VALUE2 is one of: "Rule 144A", "Regulation S" or "Unspecified"
            Use "Rule 144A" or "Regulation S" only if clearly indicated.
            Use "Unspecified" in all other cases.
   - VALUE3 is one of: "Permanent", "Temporarily" or "Unspecified"
            Use "Permanent" or "Temporarily" if the ISIN is described as such
            Use "Unspecified" if above stated information is not stated

List one VALUE1;VALUE2;VALUE3 triplet per line.
No explanations, comments, or headers.

Never repeat the same triplet.
If no ISIN . . . can be found, return: Unspecified;null;null
</OUTPUT FORMAT>

<FEW-SHOT EXAMPLES>
Example 1:
Context: The ISIN for the newly issued bond is XS1234567890.
Output:
XS1234567890;Unspecified;Unspecified
. . .
</FEW-SHOT EXAMPLES>
```

Note. Prompt largely generated using ChatGPT from OpenAI (2025).

2.5.4 Performance evaluation

Entry matching. Each information need template has one or multiple slots. Multiple entries may occur within one observation (prospectus). Each entry consists of this fixed set of interdependent slots (e.g., coupon rate, type, and phase) that together express a complete unit of information (e.g., phase in coupon rate structure). To account for multiple entries during evaluation, each *predicted* entry is matched to a *golden* entry. This is done based on the value of the chosen entry identifier slot (e.g., coupon rate). For each matched predicted entry with a golden entry, the corresponding slot-value pairs are then compared.

Triplet evaluation. Each slot-value pair in a matched entry is then categorized as: **C**orrect if the pair is present in both the predicted and golden set with matching values, **S**ubstitution error if present in both sets but with mismatching values, **I**nsertion error if only present in the prediction set, or **D**eletion if only present in the golden set. If multiple predicted entries are matched to the same golden entry (e.g., the same ISIN predicted twice), the predicted entry with the highest overlap with other pairs is considered matched. Remaining unmatched predicted entries lead to an Insertion error per pair. Remaining unmatched golden entries lead to one Deletion error per pair. Examples of this categorization can be found in Appendix D.

SER as preferred measure. To evaluate the performance across SF models, the Slot Error Rate (SER) is used due to its equal penalization of all error types. Recall and Precision are also considered. While F1 will be reported for completeness, this metric is considered inferior compared to the SER due to the discrimination in weighing different error types, despite these error types being considered equally important. The calculation of these metrics is presented in Table 2.3.

Table 2.3
Formulas of evaluation metrics

Metric	Formula	Interpretation
SER_{SF}	$\frac{S+D+I}{C+S+D}$	$\frac{\#\text{ slot errors}}{\#\text{ golden slots}}$
Precision	$\frac{C}{C+S+I}$	$\frac{\#\text{ correct slot predictions}}{\#\text{ slot predictions}}$
Recall	$\frac{C}{C+S+D}$	$\frac{\#\text{ correct slot predictions}}{\#\text{ golden slots}}$
1-F_1	$\frac{S+\frac{1}{2}(D+I)}{C+S+\frac{1}{2}(I+D)}$	-

Note. Based on Makhoul et al. (1999)

Metric aggregation. To compute these metrics at the aggregate level, both micro and macro-averages are considered. Micro-averaging sums the count of each category (C/S/D/I) across observations before calculating the metric. This gives greater weight to prospectuses with more entries and slots with more frequent golden values. Macro-averaging, on the other hand, treats each observation equally by computing the average of the per-observation metric. As statistical tests require per-observation metric values, the macro-averaged results will be focused on.

Exact match accuracy. Using the (slot, value, group) triple as an evaluation unit prevents penalizing models that have partly solved a template for a given prospectus. However, a partly solved template for a given prospectus does not satisfy an information need. Therefore, exact match accuracy is also considered as the fraction of total observations in which the template is predicted completely correct.

Statistical significance. Following the statistical testing procedure of IR models, a permutation test will be conducted on the average SER of models across information needs and within each information need. The permutation will be restricted within the same information need to satisfy the exchangeability assumption. P-values will be adjusted in the same manner as explained in Section 2.4.3.

2.6 Retrieval-augmented slot filling model

Investors are ultimately concerned with whether both the IR and SF models combined deliver a complete answer. To assess whether the combined retrieval-augmented slot filling (RASF) model can satisfy an investor's information need, the filled templates produced by the RASF model will be compared to the golden slots derived from the full prospectus, as opposed to the top 5 retrieved documents. This evaluation will be done using the best-performing IR-SF model combination for each information need.

The same evaluation metrics used for SF models are applied here. For incorrect predictions, errors are further broken down into errors attributed to the retrieval or slot filling phase. Deletion errors are attributed to slot filling if the slot is not predicted when the golden slot is present in the retrieved documents. Deletion errors are attributed to the retrieval phase if the golden slot is present in the prospectus, but not in the retrieved documents. For substitution errors, they are attributed to the slot filling phase if the golden slot is present in the retrieved documents but the model predicted another, incorrect value. This error is attributed to the retrieval phase if the prediction matches based on the retrieved chunks, but not the full prospectus. Insertion errors are exclusively attributed to slot filling, as they involve creating slots that did not occur in either the retrieved documents or the prospectus. An illustrative example is provided in Appendix D.

Table 2.4 shows how the SER of the RASF model is calculated, along with the attribution of this metric to both the retrieval and slot filling part. The total number of golden slots based on the full prospectus is defined as $N = C + D_{IR} + D_{SF} + S_{IR} + S_{SF}$.

Table 2.4
Formulas for SER of the RASF model and attribution to its components

Metric	Formula	Interpretation
SER_{RASF}	$\frac{D_{\text{IR}} + D_{\text{SF}} + S_{\text{IR}} + S_{\text{SF}} + I_{\text{SF}}}{N}$	Number of slot errors divided by number of golden slots.
$\text{SER}_{\text{IR,RASF}}$	$\frac{D_{\text{IR}} + S_{\text{IR}}}{N}$	Part of the SER of the RASF model attributed to the retrieval part.
$\text{SER}_{\text{SF,RASF}}$	$\frac{D_{\text{SF}} + S_{\text{SF}} + I_{\text{SF}}}{N}$	Part of the SER of the RASF model attributed to the slot filling part.

For evaluation, the micro SER will be focused on as it reflects its real-world performance by giving greater weight to prospectuses with more entries and slots with more frequent golden values. Exact match will also be considered. This reflects the proportion of prospectuses for which it returned a filled template that aligns with the complete prospectus (no errors from IR nor SF).

3 Results

In this chapter, performance results are presented for the IR and SF models across information needs. Additionally, it covers the query formulation, fine-tuning results, and performance of the RASF model, along with the error attribution.

3.1 Information retrieval

3.1.1 Query formulation

Of the six information needs considered, four needs were found to be referenced in various wording across different prospectuses. *Use of proceeds* and *call option* were always referred to in the same explicit wording, offering no benefit of testing different queries. Table 3.1 presents the Precision@3 for various queries related to the evaluated information needs. The highest score for the BM25 and embedding model is marked in bold. The final chosen query used for subsequent evaluation is also marked in bold. To allow for fair comparison, the same query is selected for both models. In two cases, this required picking a query that led to an inferior Precision for the embedding model.

Table 3.1
Precision@3 for various queries representing an information need

Information need	Query	BM25	Embedding
ISIN	**ISIN**	**0.73**	**0.40**
	ISIN issued bond	0.53	0.33
Coupon rate	coupon rate	0.70	0.33
	interest rate	0.13	0.33
	coupon interest rate	0.20	**0.53**
	coupon interest rate per cent	**0.40**	0.40
Coupon periodicity	payment frequency	0.20	0.20
	payment periodicity	0.20	0.40
	coupon interest periodicity	**0.70**	**0.67**
	interest coupon payment periodicity	0.27	0.60
ESG commitment	ESG commitment	0.27	0.33
	ESG strategy	**0.40**	**0.40**
	ESG sustainability strategy	**0.40**	**0.40**
Use of proceeds	use of proceeds	Not tested	
Call option	issuer call option	Not tested	

Sometimes prospectuses refer to ISINs of existing bonds, as opposed to the issued bond. Therefore, it was tested whether adding "issued bond" to the query improved scores. Prospectuses both refer to their coupon rate using "interest rate" and "coupon rate" and were thus also tested. As "coupon rate" and "periodicity" are often accompanied by "per cent" and "payment" respectively, queries with these were also tested. Finally, "ESG commitment" was often referred to as "ESG strategy".

3.1.2 Performance models

The ANOVA test confirms differences between at least two models across information needs and within all information needs, with an adjusted p-value below 0.01 except for *coupon rate*, as seen in Table 3.2.

Table 3.2
ANOVA test for all information needs

Information need	F-statistic	Adjusted p-value
ISIN	11.54	0.00
Coupon rate	0.40	0.72
Coupon periodicity	1.42	< 0.01
Call option	11.89	0.00
ESG commitment	1.35	0.00
Use of proceeds	6.24	0.00
Aggregate level	4.90	0.00

Table 3.3 presents the model's performance on an aggregate level. Fusion models seem to outperform non-hybrid models. Fusion-EQ achieved the highest nDCG@5 (0.64) and F1@5 (0.47) versus the embedding model (ΔnDCG $= 0.16$, adj. $p < 0.01$) and the BM25 model (ΔnDCG $= 0.03$, adj. $p = 0.13$). Despite being considered the most advanced method, the embedding model underperforms compared to all other models in a statistically significant way according to the pairwise t-tests (highest adj. $p < 0.01$). Except for the embedding model, no other model statistically outperforms another when limiting the Type I error rate to 5%. The full results of the pairwise tests can be found in Appendix E.

Table 3.3
Aggregate performance of all IR models

Model	nDCG@5	F1@5	Recall@5	Precision@5	MRR@5
BM25	0.60	0.46	0.69	0.38	0.59
Embedding	0.48	0.34	0.50	0.28	0.54
Fusion-EQ	**0.64**	**0.47**	0.69	0.39	0.65
Fusion-BM	0.63	**0.47**	0.70	0.39	0.61
Fusion-EM	0.61	0.44	0.65	0.36	0.66
Average	0.59	0.43	0.65	0.36	0.61

Across different information needs, there is no single model considered best across all needs (Table 3.4). Semantic-dominant models show superior performance compared to lexical models on information needs prone to the vocabulary mismatch problem (e.g., ΔnDCG $= 0.24$, adj. $p < 0.01$ on *ESG commitment*). Conversely, for queries involving precisely mentioned terminology (*ISIN*, *call option*, and *use of proceeds*), lexical-dominant models take the lead (e.g., ΔnDCG $= 0.43$, adj. $p < 0.01$ on *ISIN*). Despite its strong performance on an aggregate level, Hybrid-EQ never outperformed on any specific information need. Full pairwise testing results are provided in Appendix E.

Table 3.4
Information need-specific winners and losers based on nDCG@5

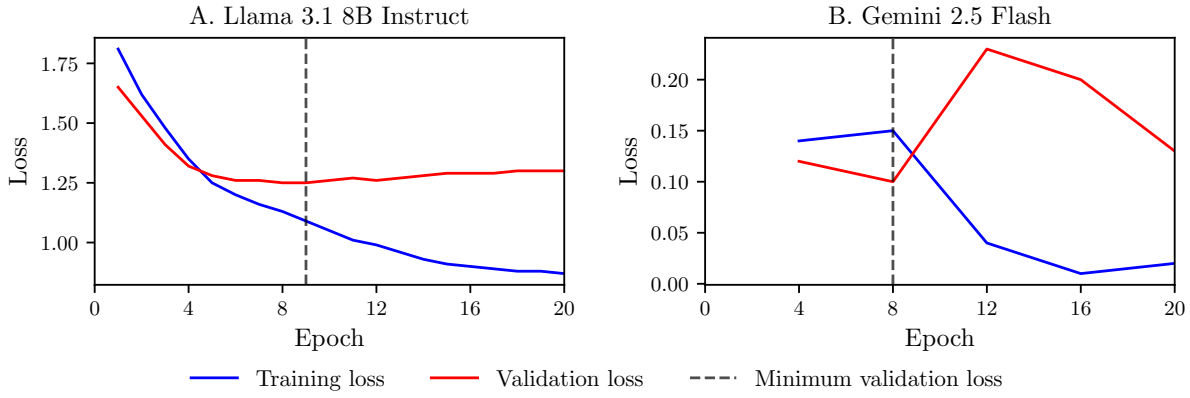
Information need	Winner (score)	Loser (score)	Diff. (adj. p-value)
ISIN	BM25 (0.85)	Embedding (0.42)	0.43 (< 0.01)
Call option	BM25 (0.83)	Embedding (0.43)	0.40 (< 0.01)
Use of proceeds	Fusion-BM (0.83)	Embedding (0.47)	0.36 (< 0.01)
Coupon periodicity	Fusion-EM (0.69)	BM25 (0.46)	0.22 (< 0.01)
ESG commitment	Embedding (0.44)	BM25 (0.19)	0.24 (< 0.01)
Coupon rate	Fusion-EM (0.58)	BM25 (0.50)	0.08 (insign. F-stat)

3.2 Slot filling

3.2.1 Model fine-tuning

Llama 3.1 8B Instruct and Gemini 2.5 Flash fine-tuned with nine and eight iterations, respectively, lead to a minimum validation loss (1.25 and 0.098) as seen in Figure 3.1. These are thus chosen as the final LoRA fine-tuned versions for evaluation. While the loss based on the training dataset continues to decline, this is likely due to overfitting. As a result, further epochs are not considered. At epoch 28, Gemini achieved a marginally lower validation loss (0.096 compared to 0.098). However, the lower epoch version is preferred to avoid overfitting.

Figure 3.1
Loss over epochs during LoRA fine-tuning



3.2.2 Performance models

The ANOVA test was significant (adj. $p < 0.01$) for all information needs and for all information needs combined. The complete results of the test can be found in Appendix F.

Across information needs, the LoRA-fine-tuned Gemini 2.5 Flash model is the best performer on all metrics, including the SER (0.18) and exact match (78%), as shown in Table 3.5. Gemini's outperformance of Llama is statistically significant across all the fine-tuning variants with adjusted p-values below 0.01. The LoRA fine-tuned Llama 3.1 8B Instruct model did not outperform the zero-shot Gemini 2.5 Flash (ΔSER = 0.76, adj. $p < 0.01$).

LoRA's outperformance compared to few-shot and zero-shot, as well as few-shot compared to zero-shot within the same models, is confirmed for both models with an adjusted p-value below 0.01. Two exceptions to this rule are the few-shot versus zero-shot Gemini model ($\Delta\text{SER} = -0.01$) and LoRA versus few-shot Llama ($\Delta\text{SER} = 0.00$). All pairwise test results and micro-averaged SERs are presented in Appendix F.

Table 3.5
Performance macro-averaged metrics for SF models

Model	Fine-tuning	SER_{SF}	Exact Match	Precision	Recall	F1
Llama 3.1 8B Instruct	Zero-shot	1.54	8%	0.43	0.62	0.47
	Few-shot	**1.05**	**35%**	0.57	0.71	0.61
	LoRA	**1.05**	**35%**	**0.56**	**0.64**	**0.58**
Gemini 2.5 Flash	Zero-shot	0.29	70%	0.86	0.86	0.85
	Few-shot	0.28	72%	0.87	0.86	0.86
	LoRA	**0.18**	**78%**	**0.91**	**0.91**	**0.91**

Table 3.6 lists the various error types that occur. Llama models hallucinate proportionally more often than Gemini (insertion error). While Gemini forgets to predict golden slots (deletion error) more in relative terms compared to Llama, the number of absolute errors made is still lower. For the predicted entries made by the model matched with the golden slots, relatively few values are filled with a value that differs from the golden slot value (substitution error).

Table 3.6
Error types as percentage of the total number of errors and total count

Model	Fine-tuning	Insertions	Deletions	Substitutions
Llama 3.1 8B Instruct	Zero-shot	70% (309)	20% (89)	10% (44)
	Few-shot	69% (195)	25% (72)	6% (17)
	LoRA	61% (150)	34% (85)	5% (12)
Gemini 2.5 Flash	Zero-shot	47% (34)	47% (34)	7% (5)
	Few-shot	43% (29)	51% (34)	6% (4)
	LoRA	43% (19)	43% (19)	14% (6)

Note. Total count is displayed in brackets and based on 120 observations.

When looking at the performance within each information need, the outperformance of Gemini compared to Llama can be confirmed across all fine-tuning variants for all information needs, except *ISIN* (only LoRA and ZS Gemini compared to Llama) and *use of proceeds* (mixed results), with a statistical significance of 5%. Table 3.7 shows this for the LoRA version of both models.

Significant differences across fine-tuning regimes cannot be observed for Gemini models within information needs. For Llama models, this is limited to the outperformance of LoRA and few-shot models compared to its zero-shot counterpart for most information needs at a significance level of 5%. This divergence between aggregate and information need-specific results is possibly due to the difference in sample size.

Table 3.7
Comparison of macro-averaged SER across models per information need

Information need	Llama FS vs. ZS	Llama LoRA vs. FS	Gemini vs. Llama (LoRA)
ISIN	0.07	0.27	0.05
Coupon rate	1.21*	0.21	1.17*
Coupon periodicity	0.55*	−0.05	0.68*
Call option	−0.28	0.12	1.21*
ESG commitment	0.58*	−0.81	1.78*
Use of proceeds	0.84*	1.42*	0.32*
Aggregate level	0.50*	0.00	0.87*

Note. *Significant result at adj. $p < .05$; FS = Few-shot, ZS = Zero-shot

Table 3.8 shows that SER and exact match are similar across information needs. The small observed differences between information needs can be attributed to the varying difficulty of the template schemas (e.g., *ISIN* vs. *use of proceeds*).

Table 3.8
Performance for LoRA models across information needs

Information need	Model (LoRA)	Macro-avg. SER	Exact match	Error distribution (I/D/S)
ISIN	Gemini	0.03	90%	0/0/2
	Llama	0.08	75%	0/0/8
Coupon rate	Gemini	0.34	75%	6/4/3
	Llama	1.51	20%	49/37/4
Coupon periodicity	Gemini	0.08	95%	1/2/0
	Llama	0.75	65%	14/13/0
Call option	Gemini	0.16	70%	2/6/0
	Llama	1.37	10%	49/6/0
ESG commitment	Gemini	0.08	80%	2/1/1
	Llama	1.86	5%	23/23/0
Use of proceeds	Gemini	0.38	60%	8/6/0
	Llama	0.70	35%	15/6/0

3.3 Performance RASF model

The RASF model achieved an average SER of 0.22 across all information needs (Table 3.9). Of this, 38% of errors are attributable to IR and 62% to SF. On average, the model achieved a Precision of 0.88 and a Recall of 0.84, meaning most predicted slots were correct and the majority of golden slots were successfully retrieved. Exact match accuracy is 64%, indicating that nearly two-thirds of prospectuses were predicted without any errors. The performance, however, differs across information needs. SER was lowest for *ISIN* (0.15), while *use of proceeds* (0.37) displayed the highest error rate.

Importantly, the proportion of errors attributed to IR versus SF varies considerably between information needs. Nearly all errors for *ISIN* (91%) were IR-driven, whereas errors in *use of proceeds* were predominantly SF-related (82%). *Coupon periodicity* achieved a high exact match accuracy (0.85) despite an SER of 0.20. This indicates that errors were typically confined to a limited number of prospectuses where many errors were being made. In contrast, *ISIN* has the lowest SER (0.15) along with the lowest exact match accuracy (50%). This indicates that while slot errors were limited, they often resulted in minor errors, preventing the return of an entirely correctly filled template 50% of the time.

Table 3.9
Performance RASF model across information needs (micro-averages)

Information need	**SER_{RASF}**	**SER attrib. to IR**	**SER attrib. to SF**	**Exact match**	**Precision**	**Recall**
ISIN	0.15	91%	9%	50%	0.85	0.85
Coupon rate	0.25	22%	78%	65%	0.85	0.84
Coupon periodicity	0.20	38%	63%	85%	0.92	0.88
Call option	0.17	22%	78%	65%	0.96	0.87
ESG commitment	0.24	63%	38%	65%	0.93	0.82
Use of proceeds	0.37	18%	82%	55%	0.82	0.80
Aggregate level	0.22	38%	62%	64%	0.88	0.84

4 Discussion

This chapter interprets the results of the IR and SF models, focusing on performance differences between the models and information needs. It further analyses the causes of errors in the RASF pipeline, covers study limitations, and provides guidance for future work.

4.1 Interpretation

4.1.1 Information retrieval

Query formulation. Results show that queries expanded with synonyms and related words (e.g., "coupon" with "interest") lead to better Precision. Conceptually different, but words often accompanied with the information need (e.g., "per cent" with "interest rate" and "issued bond" with "ISIN") benefit the keyword-based model in one of the two cases (*coupon rate*). For the embedding model, it always seems to distort its performance, likely due to adding unnecessary noise. For *ESG commitment*, using "strategy" as opposed to "commitment" improved its results. It should be noted, however, that these results are merely exploratory as no significance testing was performed due to the small sample size.

Model comparison. In line with the literature, fusion models outperformed non-hybrid models on an aggregate level. This view changes, however, when looking at information need-specific performance. Here, semantic-dominant models are better at handling the vocabulary mismatch problem due to their ability to represent words based on their similarity to others. Lexical models, on the other hand, excel when information needs are explicitly stated using the query terms. The fact that Hybrid-EQ never outperformed on any specific information need, despite achieving strong aggregate scores, suggests that model choice should be best contingent on the linguistic characteristics of the query. This challenges the existing belief that Hybrid-EQ is universally superior. One explanation for this is that fusion models benefit from diversification, while the BM25 and embedding models rely on a single approach.

Information need performance differences. In general, information needs with a higher nDCG score tend to utilize keyword-dominant models, while the opposite is true for semantic-dominant models. This difference, however, is attributable to the more abstract and narrative phrasing used to address these information needs, rather than the IR architecture employed.

4.1.2 Slot filling

Model comparison. The LoRA Gemini 2.5 Flash outperformed all other evaluated models, confirming prior findings that larger model size and more sophisticated fine-tuning lead to superior results. Interestingly, the LoRA Llama 3.1 8B Instruct model did not surpass the zero-shot Gemini 2.5 Flash model in prospectus slot filling. The indifference in performance between the zero-shot and few-shot Gemini model suggests that powerful models can generalize template-based tasks well, thereby not needing any additional

examples. For Llama, LoRA fine-tuning shows no performance gains on SER compared to in-context learning. However, LoRA fine-tuning was done on a limited set of examples (102 examples). As a result, using a higher number of examples, either human-annotated or synthetically generated, may change this view.

Information need performance differences. Differences in the SER across information needs can be attributed to the difficulty in classification (*use of proceeds* and *call option*) and template complexity (*coupon rate*). For example, *use of proceeds*, is often challenged with domain-specific terminology (e.g., "strengthen Tier 2 capital ratios"). *Coupon rate* requires identifying the linkage between slots when faced with multi-faced rate structures.

4.1.3 RASF model and error attribution

With an exact match accuracy of 64% and SER of 22%, the RASF model demonstrates promising results. At the same time, it highlights that more than one in five slots remain incorrectly filled. To support fully automated prospectus processing, further refinement of both the retrieval and slot filling component is required.

Across information needs, two factors largely dictate the RASF performance: the alignment of the query with all the required information for slot filling and the intrinsic complexity of the slot filling task. The latter is demonstrated by the positive correlation between SER_{RASF} attributed to SF (RASF model) with the SER_{SF} (SF model). This is the case for *coupon rate*, *call option*, and *use of proceeds* with both the highest SER_{RASF} attribution to SF and the highest isolated SER for SF compared to other information needs.

While the high SER attribution to IR for *coupon periodicity* and *ESG commitment* aligns with the weaker performance observed during general retrieval testing for these information needs, this explanation does not hold for *ISIN*. The retrieval part of the RASF model struggles with *ISIN* due to the misalignment between the template schema slots and the focus of the IR model. The IR model focuses exclusively on retrieving documents related to the ISIN code. At the same time, the *ISIN* template also includes a slot regarding the ISIN regulation (Rule 144A, Regulation S, or Unspecified). As this regulation is often located in a separate, unrelated section, it is frequently overlooked, resulting in a high SER attribution to the IR part.

4.2 Limitations

Limitations arise from the design and evaluation of the IR and SF models. Regarding IR, a fixed-size chunking strategy was adopted, which often splits chunks at semantically inappropriate boundaries and neglects document structure, as prior studies have shown to lead to better retrieval results. In addition, the evaluation of IR models did not consider diversity in retrieved documents to cover all parts of the template schema, despite being beneficial for the slot filling model. Specifically, for the *ISIN*, only the presence of the ISIN code was considered, not the presence of the associated regulatory information. This is because the template schema was constructed after the IR models were developed and evaluated.

For SF, only decoder-only LMs were considered due to the limited availability of training data, despite encoder-decoder models showing superior performance in the literature. Moreover, the SF model was evaluated only with the top five retrieval documents, although using more documents would likely lead to better results. Finally, the reliance on single bond prospectuses and a small sample size may limit the generalizability of the study's findings.

4.3 Future work

Future studies should address these limitations by considering semantic-based chunking strategies and adopting a RASF model which includes a feedback loop where queries are further refined based on additional data required by the SF pipeline. The construction of an encoder-decoder model, once enough training data can be assembled, would also be valuable. Finally, this study focused on characteristics that can be represented in a template schema. Future research could, however, also investigate the ability of summarization of nuanced, more complex topics that occur in prospectuses, such as the involved risks.

Conclusion

This thesis develops and evaluates retrieval-augmented slot filling models for automating template filling of key information from European bond prospectuses. It does so by addressing three research questions. First, it investigates which information retrieval system performs best. A fusion model that equally weights the lexical and semantic relevance scores achieves the highest performance, with an nDCG@5 of 0.64 and F1@5 of 0.47 on aggregate. Conversely, lexical-dominant models excel at precisely phrased information needs like *ISIN* and *use of proceeds*, while semantic-dominant models perform better with context-rich information needs like *ESG commitment*.

The second focus lies on evaluating the most effective slot filling models for populating a predefined template. A LoRA fine-tuned Gemini 2.5 Flash model outperformed all its counterparts, including its zero-shot and few-shot equivalent, and all variants of the Llama 3.1 8B Instruct model. It achieved an exact match accuracy of 78% and a slot error rate of 0.18. These results confirm that fine-tuning a large-scale language model is effective, despite the high computational cost compared to zero-shot and few-shot approaches.

The final objective evaluates whether the combination of the retrieval and slot filling pipeline can satisfy investors' information needs. When evaluated against a prospectus, the retrieval-augmented slot filling model attains an exact match accuracy of 64% and a slot error rate of 0.22. The SER decomposition showed that 38% of errors originated from the retrieval pipeline and 62% from slot filling. Further investigation revealed template complexity and query misalignment to be the two main error causes. While these results are promising, they highlight that further refinement is needed to fully automate prospectus processing.

By benchmarking both the retrieval and slot filling pipeline and attributing errors, this thesis sheds light on the available techniques to translate legal-financial documents into accessible, structured data. It shows that retrieval model choice is contingent on the information need and large-scale language models consistently outperform small-scale language models, even if LoRA-finetuned. With this research, an initial contribution is made toward developing retrieval-augmented slot filling models that reduce information asymmetry and promote better investment decisions.

Bibliography

Aguirre, M., Méndez, A., Del Pozo, A., Torres, M. I., & Torralbo, M. (2025). Fine-tuning medium-scale LLMs for joint intent classification and slot filling: A data-efficient and cost-effective solution for SMEs. *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, 251–262.

Andreas, V. M., Winata, G. I., & Purwarianti, A. (2021). A comparative study on language models for task-oriented dialogue systems. *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, 1–5. https://doi.org/10.1109/ICAICTA53211.2021.9640249

Bhat, S. R., Rudat, M., Spiekermann, J., & Flores-Herr, N. (2025). Rethinking chunk size for long-document retrieval: A multi-dataset analysis. https://doi.org/10.48550/arXiv.2505.21700

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. https://doi.org/10.48550/arXiv.2005.14165

Bruch, S., Gai, S., & Ingber, A. (2023). An analysis of fusion functions for hybrid retrieval. *ACM Trans. Inf. Syst.*, *42*(1), 20:1–20:35. https://doi.org/10.1145/3596512

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Liu, T., Chang, B., Sun, X., Li, L., & Sui, Z. (2024). A survey on in-context learning. https://doi.org/10.48550/arXiv.2301.00234

ESMA. (2022). *Parsing prospectuses: A text mining approach.* Publications Office. LU. Retrieved July 22, 2025, from https://data.europa.eu/doi/10.2856/03284

European Commission. (n.d.). *Securities prospectus*. Retrieved July 22, 2025, from https://finance.ec.europa.eu/capital-markets-union-and-financial-markets/financial-markets/securities-markets/securities-prospectus_en

Fatemi, S., & Hu, Y. (2023). A comparative analysis of fine-tuned LLMs and few-shot learning of LLMs for financial sentiment analysis [version: 1]. https://doi.org/10.48550/arXiv.2312.08725

Furnas, G. W., Landauer, T. K., Gomez, L. M., & Dumais, S. T. (1987). The vocabulary problem in human-system communication. *Commun. ACM*, *30*(11), 964–971. https://doi.org/10.1145/32206.32212

Google. (n.d.). *Overview of prompting strategies* [Google cloud]. Retrieved August 4, 2025, from https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/prompt-design-strategies

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-rank adaptation of large language models. https://doi.org/10.48550/arXiv.2106.09685

Hugging Face. (n.d.-a). *LoRA*. Retrieved August 2, 2025, from https://huggingface.co/docs/diffusers/training/lora

Hugging Face. (n.d.-b). *MTEB leaderboard - a hugging face space by mteb*. Retrieved July 22, 2025, from https://huggingface.co/spaces/mteb/leaderboard

Indurkhya, N., & Damerau, F. J. (Eds.). (2010). *Handbook of natural language processing*. Chapman; Hall/CRC.

Jurafsky, D., & Martin, J. H. (2025). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition with language models* (3rd ed.). https://web.stanford.edu/~jurafsky/slp3

Juvekar, K., & Purwar, A. (2024). Introducing a new hyper-parameter for RAG: Context window utilization. https://doi.org/10.48550/arXiv.2407.19794

Kuzi, S., Zhang, M., Li, C., Bendersky, M., & Najork, M. (2020). Leveraging semantic and lexical matching to improve the recall of document retrieval systems: A hybrid approach. https://doi.org/10.48550/arXiv.2010.01195

Lu, Y.-T., & Huo, Y. (2025). Financial named entity recognition: How far can LLM go? In C.-C. Chen, A. Moreno-Sandoval, J. Huang, Q. Xie, S. Ananiadou, & H.-H. Chen (Eds.), *Proceedings of the joint workshop of the 9th financial technology and natural language processing (FinNLP), the 6th financial narrative processing (FNP), and the 1st workshop on large language models for finance and legal (LLMFinLegal)* (pp. 164–168). Association for Computational Linguistics. Retrieved July 22, 2025, from https://aclanthology.org/2025.finnlp-1.15/

Makhoul, J., Kubala, F., Schwartz, R., Weischedel, R., et al. (1999). Performance measures for information extraction. *Proceedings of DARPA broadcast news workshop*, *249*, 252.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Meaney, C., Stukel, T. A., Austin, P. C., & Escobar, M. (2023). Comparing variation in tokenizer outputs using a series of problematic and challenging biomedical sentences. https://doi.org/10.48550/arXiv.2305.08787

Mehri, S., & Eskenazi, M. (2021). GenSF: Simultaneous adaptation of generative pre-trained models and slot filling. https://doi.org/10.48550/arXiv.2106.07055

Mosbach, M., Pimentel, T., Ravfogel, S., Klakow, D., & Elazar, Y. (2023). Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In A. Rogers, J. Boyd-Graber, & N. Okazaki (Eds.), *Findings of the association for computational linguistics: ACL 2023* (pp. 12284–12314). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-acl.779

Nguyen, X.-S. (2024). *Ngxson/demo_simple_rag_py* [Hugging face]. Retrieved August 4, 2025, from https://huggingface.co/ngxson/demo_simple_rag_py/tree/main

Nogueira, R., Yang, W., Cho, K., & Lin, J. (2019). Multi-stage document ranking with BERT. https://doi.org/10.48550/arXiv.1910.14424

OpenAI. (2025). *ChatGPT* (Version 4o). https://chat.openai.com/chat

Qu, R., Tu, R., & Bao, F. (2024). Is semantic chunking worth the computational cost? https://doi.org/10.48550/arXiv.2410.13070

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training [Publisher: San Francisco, CA, USA].

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1–67. Retrieved July 22, 2025, from http://jmlr.org/papers/v21/20-074.html

Regulation (EU) 2017/1129 (2017). Retrieved July 22, 2025, from https://eur-lex.europa.eu/eli/reg/2017/1129/oj/eng

Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. (1995). *Okapi at TREC-3*. British Library Research; Development Department.

Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, *3*(4), 333–389. https://doi.org/10.1561/1500000019

Sharkey, E., & Treleaven, P. (2024). BERT vs GPT for financial engineering. https://doi.org/10.48550/arXiv.2405.12990

Smucker, M. D., Allan, J., & Carterette, B. (2007). A comparison of statistical significance tests for information retrieval evaluation. *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 623–632. https://doi.org/10.1145/1321440.1321528

Song, X., Salcianu, A., Song, Y., Dopson, D., & Zhou, D. (2021). Fast WordPiece tokenization. https://doi.org/10.48550/arXiv.2012.15524

Sun, W., Zhang, C., Zhang, X., Yu, X., Huang, Z., Chen, P., Xu, H., He, S., Zhao, J., & Liu, K. (2024). Beyond instruction following: Evaluating inferential rule following of large language models. https://doi.org/10.48550/arXiv.2407.08440

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021, October 21). BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. https://doi.org/10.48550/arXiv.2104.08663

Thattantavida, S. M., & Sarkar, G. (2025). *Enhancing RAG performance with smart chunking strategies* [IBM developer]. Retrieved July 22, 2025, from https://developer.ibm.com/articles/awb-enhancing-rag-performance-chunking-strategies/

Urbano, J., Lima, H., & Hanjalic, A. (2019). Statistical significance testing in information retrieval: An empirical analysis of type i, type II and type III errors, 505–514. https://doi.org/10.1145/3331184.3331259

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wang, T., Roberts, A., Hesslow, D., Scao, T. L., Chung, H. W., Beltagy, I., Launay, J., & Raffel, C. (2022). What language model architecture and pretraining objective work best for zero-shot generalization? https://doi.org/10.48550/arXiv.2204.05832

Yepes, A. J., You, Y., Milczek, J., Laverde, S., & Li, R. (2024). Financial report chunking for effective retrieval augmented generation. https://doi.org/10.48550/arXiv.2402.05131

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., ... Wen, J.-R. (2025). A survey of large language models. https://doi.org/10.48550/arXiv.2303.18223

List of Figures

List of Tables

Appendix A

Categorization all bond prospectus features

A. Explicit features

Assumed to be explicitly stated in the bond prospectus.

1. ISIN
2. Name of issuer
3. Denomination
4. Currency
5. Issue price
6. Issue date
7. Coupon rate (interest rate)
8. Maturity date

B. Inference features

Assumed to be needed to be derived implicitly.

1. Coupon type
2. Coupon periodicity
3. Issuer call option type
4. Bondholder put option type
5. Seniority
6. Securitization
7. Credit rating bond

C. Summary features

Assumed to require summarization and understanding of broader lines of text.

1. ESG commitment
2. Use of proceeds

This results in a total of 17 identified features.

Appendix B

Template schemas and optimized system prompts

Every template schema is a de facto part of the system prompt. As a result, it can be derived from these prompts. Few-shot examples are only included when evaluating the few-shot LLM. All prompts were primarily generated using ChatGPT from OpenAI (2025).

Overview of all prompts:

1. **ISIN**: Figure B.1
2. **Coupon rate**: Figure B.2
3. **Coupon periodicity**: Figure B.3
4. **Call option**: Figure B.4
5. **ESG commitment**: Figure B.5
6. **Use of proceeds**: Figure B.6

Figure B.1
System prompt "ISIN"

```
You are an information extraction assistant.

Find all ISIN values that specifically identify the newly issued bond(s)
described in the context.
Exclude ISINs of any bonds mentioned only as references, collateral,
or predecessors, even if related to the issuance, unless they are explicitly
described as the newly issued bond.

Return each ISIN in the format:
VALUE1;VALUE2;VALUE3

Where:
- VALUE1 is the ISIN code (e.g. US1234567890)
- VALUE2 is one of: "Rule 144A", "Regulation S" or "Unspecified"
* Use "Rule 144A" or "Regulation S" only if clearly indicated.
* Use "Unspecified" in all other cases.
- VALUE3 is one of: "Permanent", "Temporarily" or "Unspecified"
* Use "Permanent" or "Temporarily" if the ISIN is described as such
* Use "Unspecified" if above stated information is not stated

List one VALUE1;VALUE2;VALUE3 triplet per line.
No explanations, comments, or headers.
Never repeat the same triplet.

If no ISIN for the newly issued bond can be found, return:
Unspecified;null;null
```

```
Example 1:
Context: The ISIN for the newly issued bond is XS1234567890.
Output:
XS1234567890;Unspecified;Unspecified

Example 2:
Context: The Regulation S Notes will be issued under ISIN XS2222222222
and the Rule 144A Notes under ISIN US2222222222.
Output:
XS2222222222;Regulation S;Unspecified
US2222222222;Rule 144A;Unspecified

Example 3:
Context: The temporarily ISIN for the Notes is XS3333333333.
Output:
XS3333333333;Unspecified;Temporarily

Example 4:
Context: The ISIN of the existing bonds issued in 2021 is XS4444444444.
Output:
Unspecified;null;null

Example 5:
Context: No ISIN has yet been assigned to the newly issued bonds.
Output:
Unspecified;null;null
```

Note. Prompt largely generated using ChatGPT from OpenAI (2025).

Figure B.2

System prompt "coupon rate"

```
You are an information extraction assistant.

Find all interest/coupon rate values that describe the newly issued bond
in the context.
Exclude any interest/coupon rates mentioned only as references, collateral,
predecessors, or related bonds, even if related to the issuance, unless they
are explicitly described as part of the newly issued bond.

Return each interest/coupon rate entry in the format:
VALUE1;VALUE2;VALUE3

Where:
- VALUE1 (rate type) is one of: "Fixed", "Floating", "Range", "Other"
  or "Unspecified"
* "Fixed": for fixed coupon rate parts
* "Floating": for floating coupon rate parts
* "Other": if a known but non-standard or unusual coupon rate type
* For step-up or multi-phase: return one entry per change
```

```
- VALUE2 depends on the rate type:
* Fixed rates: decimal percentage with three decimals and dot notation,
  no symbol (e.g. "2.2 percent" becomes "2.200")
* For floating rates:
    - Provide both the reference rate name and spread separated by a comma
      as a single entry (e.g. "1-month LIBOR,0.150")
    - Reference rate name: apply the following format strictly regardless of
      how it is written in the context:
        Must be "[number]-[periodicity] [REFERENCE NAME]" in this exact order
        (e.g. "1-month LIBOR")
        or [REFERENCE NAME] alone if the tenor is not given (e.g. "LIBOR")
        Only use official names of the reference rates (e.g. LIBOR) otherwise
        consider it not specified
        If the reference rate name is not specified, use "Unspecified" for it
        (e.g. "Unspecified,0.200")
    - Spread:
        Apply the same format as for the fixed rate
        If the spread is missing, use "Unspecified" for it
        (e.g. "1-month LIBOR,Unspecified")
* For any other rate type return "null" as VALUE2

- VALUE3 (coupon phase) is one of: "Constant", "Initial", "Follow-up"
  or "Unspecified"
* "Constant": for single, unchanged coupon structures
* "Initial": first phase in a rate switch
* "Follow-up": subsequent phase in a rate switch
* "Unspecified": if phase information is missing or unclear

List one VALUE1;VALUE2;VALUE3 pair per line.
No explanations, comments, or headers.
Never repeat the same triplet.

Return:
* "Unspecified;null;null" if the coupon rate is not clearly stated
* "Incomplete;null;null" if the rate structure appears underspecified or
  only partially described leading to essential details being omitted

Example 1:
Context: The newly issued bond bears an interest equal to the LIBOR on 3 months
basis with a spread of 0.15 per cent, after 1 year this changes to a fixed rate
of 2.2 per cent.
Output:
Floating; 3-month LIBOR,0.150; Initial
Fixed; 2.200; Follow-up

Example 2:
Context: The newly issued bond interest rate will be linked to EURIBOR.
The bond issued in in 2020 has a fixed rate of 10%.
Output:
Floating;EURIBOR,Unspecified;Constant
```

```
Example 3:
Context: The newly issued bond interest rate will vary and be contigent
on the occurance of a natural disaster and range between 10-50%.
After 2 years it will be fixed to 10%.
Output:
Other;null;Initial
Fixed;10.000;Follow-up

Example 4:
Context: The bond starts at 1% for the first 5 years, then steps up to 1.75%.
After this the coupon will depend on the occurance of a a tornado.
Output:
Fixed;1.000;Initial
Fixed;1.750;Follow-up
Other;null;Follow-up

Example 5:
Context: This bond is considered high quality and will be offered to
professional investors only.
Output:
Unspecified;null;null
```

Note. Prompt largely generated using ChatGPT from OpenAI (2025).

Figure B.3

System prompt "coupon periodicity"

```
You are an information extraction assistant.

Find the periodicity of interest/coupon payments of the newly issued bond
described in the context.
Exclude any periodicity information related to other bonds mentioned only as
references, collateral, or predecessors, even if related to the issuance,
unless they are explicitly described as part of the newly issued bond.

Return each periodicity in the format:
VALUE1;VALUE2

Where:
- VALUE1 is one of the following frequencies: "Monthly", "Quarterly",
  "Semi-annually" or "Annually"
* Use multiple lines if the periodicity changes due to the bond's structure
  (e.g. fixed-to-floating)
* If the periodicity is different than the allowed values use "Complex" as value
- VALUE2 is one of: "Initial", "Follow-up", "Constant" or "Unspecified"
* "Constant": if only one periodicity applies
* "Initial": for the first phase in a structural change (e.g. fixed-to-floating)
* "Follow-up": for any subsequent phase
* "Unspecified": if it is unclear or not specified
```

```
List one VALUE1;VALUE2 pair per line.
No explanations, comments, or headers.
Never repeat the same triplet.

If the periodicity is not clearly stated, return:
Unspecified;null

Example 1:
Context: The Notes will bear interest at a rate of 4.250% per annum,
payable each year on 11 March, commencing on 11 March 2025.
Output:
Annually;Constant

Example 2:
Context: The bonds will pay interest every quarter for the first 3 years,
and then semi-annually thereafter until maturity.
Output:
Quarterly;Initial
Semi-annually;Follow-up

Example 3:
Context: The bond will be fixed until 2026 with coupons paid each year on
February 1. After this it will switch to the LIBOR with a spread of 5%.
Output:
Annually;Unspecified

Example 4:
Context: The coupon will be paid bi-weekly for the first year.
After this it will be paid every quarter.
Output:
Complex;Initial
Quarterly;Follow-up

Example 5:
Context: The bond is of high quality and has a coupon of 5%.
Output:
Unspecified;null
```

Note. Prompt largely generated using ChatGPT from OpenAI (2025).

Figure B.4

System prompt "call option"

```
You are an information extraction assistant.

Find all unique call option types (i.e. an option for the issuer to redeem
the bond before maturity) that are specifically related to the
newly issued bond in the context.
Exclude any call options that relate to other bonds (e.g. references,
collateral, predecessors), unless they are clearly part of the
newly issued bond's terms.
```

```
Exclude any other options (e.g. put option).

Use the following possible values:
- "Standard": Optional redemption by issuer at or after a specific date
  (e.g. first call date)
- "Make-whole": Redemption based on present value of remaining payments
- "Clean-up": Redemption triggered when a large portion (e.g. 75%) of the notes
  have been redeemed
- "Tax": Redemption due to adverse changes in tax laws
- "Regulatory": Redemption due to changes in regulation or
  account classfication affecting the issuer's capital or compliance status
- "Other": If a clearly described call option is present but does not fit
  any of the above
- "Unknown": If the call option is present but the type is not clearly specified

Output one line per uniquely identified call option.
Do not add comments, explanations, or headers.
Never repeat the same line.

If it explicitly states there is no call options present return
"No call option"
If there is no information about presence or absence of call option return
"Unspecified"


Example 1:
Context: The Issuer may, at any time from and including 11 December 2029 to but
excluding the Maturity Date, redeem the bonds in whole, but not in part, at
their outstanding principal amount plus accrued interest.
In addition, if 70% or more of the notes have been redeemed or purchased
and cancelled, the Issuer may redeem the remaining notes in full.
Prior to 11 December 2029, the bonds may also be redeemed at a
make-whole amount calculated by the calculation agent.

Output:
Standard
Clean-up
Make-whole

Example 2:
Context: The bonds may be redeemed early at the Issuer's option in the event
that changes to tax law result in withholding tax obligations.
The Issuer may also redeem the bonds if regulatory changes negatively impact
their capital treatment.

Output:
Tax
Regulatory

Example 3:
```

```
Context:
If 75% of the aggregate principal amount of the bonds has been purchased or
redeemed, the Issuer may redeem the remaining bonds in whole.
The prospectus also includes a provision allowing early redemption at a
make-whole amount prior to 2029.
Furthermore, bonds may be redeemed at their principal amount from
11 December 2029 to the maturity date.

Output:
Clean-up
Make-whole
Standard

Example 4:
Context: The bonds shall not be subject to redemption prior to
the maturity date under any circumstances.

Output:
No call option

Example 5:
Context: This bond is considered high quality and will be offered to
professional investors only.

Output:
Unspecified
```

Note. Prompt largely generated using ChatGPT from OpenAI (2025).

Figure B.5

System prompt "ESG commitment"

```
You are an information extraction assistant.

Find every distinct ESG/sustainability commitment of the newly issued bond or
bond issuer described in the context.
Exclude any ESG/sustainability commitments mentioned only as references,
collateral, predecessors, or related bonds.
Exclude any ESG/sustainability talk considered as risk factor
or descriptive analysis.

Return each line in the format:
VALUE1;VALUE2

Where:
- VALUE1 is one of the following commitment types:
* "Strategy integration": if explicit embedding ESG/Sustainability into the
  issuer's corporate strategy
* "Performance linking": if explicitly states linking bond financial terms
  (e.g. coupon rate) to sustainability KPIs
* "Environmental initiatives": if setting explicit, quantitative targets
```

```
  directly related to reducing greenhouse gas emissions
* "Designated green financing": if explictely states proceeds of
  the newly issued bond will be allocated to sustainability projects

- VALUE2 is the structured numeric ESG target
* Only populate this for VALUE = "Environmental initiatives",
  otherwise return "null"
* Return this in the following format:
  Metric=<metric>, Value=<number><unit>, By=<YYYY>
    <metric>: the thing being measured (e.g. GHG emissions)
    <number><unit>: the target amount (e.g. -40%, -10000tCO2)
    <YYYY>: the target year (e.g. 2030)
* If no clear quantitative goal is mentioned, return "Unspecified"
* If multiple distinct targets appear, separate them with the pipe character
* Only return the target if you can extract all three components
  (metric, value, and year); otherwise return "Unspecified" for that target
* E.g. Environmental initiatives;Metric=GHG Scope 1,Value=-40%,By=2030
  |Metric=GHG Scope 2,Value=-30%,By=2030|Unspecified

List one VALUE1;VALUE2 pair per line.
Return one line per commitment type. Each commitment type can never be
again repeated.
No explanations, comments, or headers.
Never repeat the same triplet.

If no ESG/sustainability commitment is stated or fits into the stated
categories, return:
Unspecified;null

Example 1:
Context: The issuer has incorporated sustainability goals into its
corporate strategy. The issuer has committed to reduce Scope 1 and 2
GHG emissions by 40% and 30% respectively by 2030.
Output:
Strategy integration;null
Environmental initiatives;Metric=GHG Scope 1,Value=-40%,By=2030
|Metric=GHG Scope 2,Value=-30%,By=2030

Example 2:
Context: The bond's coupon rate will step down by 25 basis points if the issuer
fails to meet its renewable energy production KPI.
Output:
Performance linking;null

Example 3:
Context: The issuer is commited to the environment and will do its best to
reduce its impact. The proceeds of the bond issued back in 2020 were used to
sustainability projects.
Output:
Unspecified;null
```

```
Example 4:
Context: The proceeds of the newly issued bond will be allocated to
renewable energy, clean transportation, and energy efficiency projects.
Output:
Designated green financing;null

Example 5:
Context: This document does not include any ESG commitments related to
the newly issued bond.
Output:
Unspecified;null
```

Note. Prompt largely generated using ChatGPT from OpenAI (2025).

Figure B.6

System prompt "use of proceeds"

```
You are an information extraction assistant.

Find all stated use of proceeds purposes for the newly issued bond described
in the context.
Exclude any use of proceeds information related to other bonds mentioned only
as references, collateral, or predecessors, even if related to the issuance,
unless they are explicitly described as the newly issued bond.

Use the following possible values:
- "General corporate purposes"
- "Refinancing existing debt"
- "Investments and acquisitions"
- "Regulatory capital purposes": to meet specific regulatory requirements
  for financial institutions
- "Other": if a clearly described purpose does not fit above categories

Output one line per unique use of proceeds.
Do not add comments, explanations, or headers.
Never repeat the same line.

If no clear use of proceeds are stated return "Unspecified"

Example 1:
Context: The proceeds will be used to pay off our current outstanding debt
expiring soon and for general corporate purposes.
Output:
Refinancing Existing Debt
General corporate purposes

Example 2:
Context: Funds will support strategic acquisitions and general corporate
purposes.
Output:
```

```
Investments and Acquisitions
General corporate purposes

Example 3:
Context: This bond issuance strengthens the bank's Tier 2 capital base.
Output:
Regulatory capital purposes


Example 4:
Context: The issuer stated the proceeds will be allocated to a new headquarters
project.
Output:
Other

Example 5:
Context: No explicit use of proceeds purpose is stated.
Output:
Unspecified
```

Note. Prompt largely generated using ChatGPT from OpenAI (2025).

Appendix C

Parameters of LoRA fine-tuning

Vertex AI's platform to fine-tune Gemini 2.5 Flash allows only the learning rate multiplier and adapter size to be adapted. These parameters were, however, left unspecified to have Vertex AI choose the optimal, but unspecified, value.

Llama 2.5 8B Instruct was fine-tuned using Hugging Face's Transformer Reinforcement Learning, Transformer, PEFT, and Datasets libraries. The chosen parameter values can be found in Table C.1. Most values were chosen based on commonly set defaults, to prevent overfitting given the limited training data and to accommodate computational constraints. A systematic refinement of parameters was not performed, with the exception of training parameters to accommodate GPU constraints. The model was trained using an A100 40GB GPU via Google Colab.

Table C.1

Parameter values of LoRA and training configuration

Parameter	**Value**	**Motivation**
Base model	Llama 3.1 8B Instruct	Best in class in executing rules (Sun et al., 2024)
Precision type	Full precision	-
	LoRA configuration	
Rank	8	Default values and due to GPU constraints (Hugging Face, n.d.-a)
Alpha	8	
Dropout	0.05	
Target modules	q_proj, k_proj, v_proj, o_proj	-
Bias terms adaptation	No	Default value (Hugging Face, n.d.-a)
	Training parameters	
Learning rate	2×10^{-4}	Slightly higher than average to allow for faster convergence
Number of considered epochs	20	Any higher number of training cycles would lead to overfitting
Warmup ratio	0.03	-
Precision type	Mixed - bfloat16	Due to GPU constraints full precision was not feasible
Evaluation strategy	Per epoch	The metric is calculated and evaluated per epoch
Selection metric	Evaluation loss	Loss on validation data is used to choose best model

Appendix D

Categorization examples

The *ISIN* template has three slots: ISIN code, distribution regulation (Rule 144A, Regulation S, or Unspecified) and issuance status (Permanent, Temporarily, and Unspecified). If no ISIN is found, the ISIN code slot value must be "Unspecified".

Overview of all examples:

1. **Example 1**: Two unlinked entries
2. **Example 2**: Two predicted pairs match the same golden answer
3. **Example 3**: Matching pairs with different slot values
4. **Example 4**: RASF model evaluation with error attribution

Example 1: Two unlinked entries

Entry	**Code**	**Regulation**	**Status**
	Predicted pairs		
1	BE4433221100	Rule 144A	Temporarily
2	FR2233445566	Unspecified	Unspecified
	Golden pairs		
A	FR2233445566	Unspecified	Unspecified
B	XS4949920392	Rule 144A	Permanent

Only entry 2 and A can be matched. Given all pairs match within this entry, it results in three correct answers. Entry 1 of the prediction set cannot be matched, resulting in 3 insertion errors. Entry B was not found predicted and thus results in 3 deletion errors. The SER for this observation is $6/6 = 1$.

Example 2: Two predicted pairs match the same golden answer

Entry	**Code**	**Regulation**	**Status**
	Predicted pairs		
1	BE4433221100	Rule 144A	Temporarily
2	BE4433221100	Unspecified	Unspecified
	Golden pairs		
A	BE4433221100	Rule 144A	Temporarily

Two entries with the same value for the identifier slot. The entry matching best across all pairs is entry 1 and is thus matched to entry A. This results in three correct answers and 3 insertion errors. The SER is $3/3 = 1$.

Example 3: Matching pairs with different slot values

Entry	Code	Regulation	Status
	Predicted pairs		
1	Unspecified	Rule 144A	null
	Golden pairs		
A	Unspecified	null	null

Two entries can be matched based on the code. The predicted pair, however, returned one extra value, while it should not. It has one value pair correct and made one insertion error. Note that the matching status pair is not given credit for empty slots. The SER is 1/1.

Example 4: RASF model evaluation with error attribution

Entry	Code	Regulation	Status
	Predicted pairs		
1	XS4949920392	Rule 144A	Temporarily
	Chunk golden answer		
I	XS4949920392	Rule 144A	Permanent
	Prospectus golden answer		
A	XS4949920392	Rule 144A	Permanent
B	BE4433221100	Rule 144A	Temporarily

Since the RASF evaluation considers both IR and SF errors, the predicted pairs are matched against the prospectus golden answers. Two entries can be matched: entry 1 and entry A. Within this match, two slot values are correct, while the status slot differs, resulting in one substitution error attributed to the SF model, as this mismatch is also present in the chunk golden answer. Additionally, prospectus golden entry B is not retrieved in the chunk-level golden answers, resulting in three deletion errors attributed to the IR model. The SER is 4/6

Appendix E

Additional information on statistical tests for IR models

The one-way ANOVA test for all information needs combined used the nDCGs for each model on each information need for each prospectus as observations (120 observations per model). All other tests were performed on the models within a certain information with the nDCGs for each prospectus within this information need as observations (20 observations per model).

Overview of all figures and tables:

1. **Table E.1**: ANOVA test results
2. **Figure E.1**: Approximated distribution of F-statistic
3. **Figure E.2**: Approximated distribution of mean differences
4. **Table E.2**: Results pairwise T-test for *ISIN*
5. **Table E.3**: Results pairwise T-test for *coupon rate*
6. **Table E.4**: Results pairwise T-test for *coupon periodicity*
7. **Table E.5**: Results pairwise T-test for *call option*
8. **Table E.6**: Results pairwise T-test for *ESG commitment*
9. **Table E.7**: Results pairwise T-test for *use of proceeds*

Table E.1

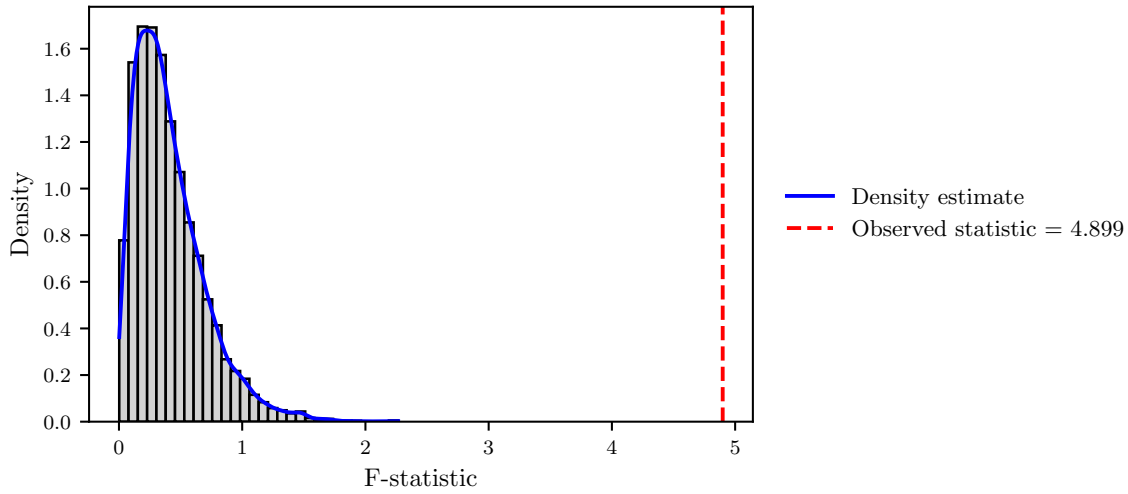
One-way ANOVA results for model comparison on nDCG@5

Information need	F-statistic	Raw p-value	Adjusted p-value
Aggregate level	4.90	0.00	0.00
ISIN	11.54	0.00	0.00
Coupon rate	0.40	0.72	0.72
Coupon periodicity	1.42	< 0.01	< 0.01
Call option	11.89	0.00	0.00
Use of proceeds	6.24	0.00	0.00
ESG commitment	1.35	0.00	0.00

Note. P-values adjusted using the Benjamini-Hochberg procedure with an FDR of 5%.

Figure E.1

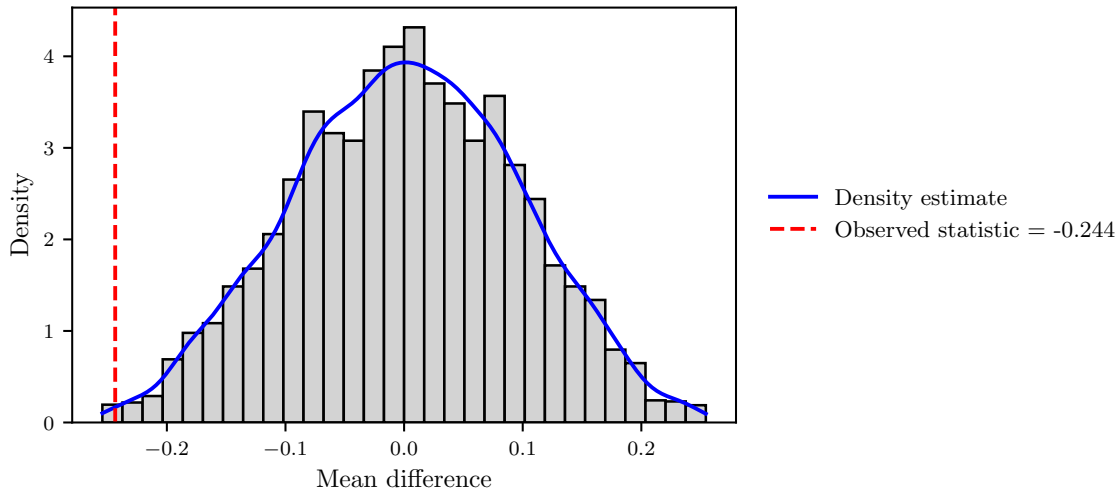
Approximated distribution F-statistic for aggregate level



Note. This serves as an illustration as one of the six ANOVA tests.

Figure E.2

Approximated distribution mean nDCG@5 differences BM25-embedding for "ESG commitment"



Note. This serves as an illustration as one of the sixty pairwise T-tests.

Table E.2

Mean difference nDCG@5 across models for "ISIN"

	Embedding	Fusion-EQ	Fusion-BM	Fusion-EM
BM25 (0.85)	0.43 (0.00)	0.08 (0.07)	0.07 (0.03)	0.16 (0.03)
Embedding (0.42)	–	-0.35 (0.00)	-0.36 (0.00)	-0.27 (0.00)
Fusion-EQ (0.77)		–	-0.01 (0.62)	0.08 (0.06)
Fusion-BM (0.78)			–	0.09 (0.07)
Fusion-EM (0.69)				–

Note. nDCG@5 in brackets in the first column, adj. p-value in brackets behind difference

Table E.3

Mean difference nDCG@5 across models for "coupon rate"

	Embedding	Fusion-EQ	Fusion-BM	Fusion-EM
BM25 (0.50)	-0.01	-0.07	-0.04	-0.08
Embedding (0.51)	–	-0.06	-0.03	-0.07
Fusion-EQ (0.57)		–	0.03	-0.01
Fusion-BM (0.54)			–	-0.04
Fusion-EM (0.58)				–

Note. nDCG@5 in brackets in the first column, insignificant F-statistic

Table E.4

Mean difference nDCG@5 across models for "coupon periodicity"

	Embedding	Fusion-EQ	Fusion-BM	Fusion-EM
BM25 (0.46)	-0.15 (0.07)	-0.12 (0.04)	-0.08 (0.06)	-0.22 (0.00)
Embedding (0.61)	–	0.03 (0.66)	0.07 (0.28)	-0.07 (0.09)
Fusion-EQ (0.59)		–	0.04 (0.18)	-0.10 (0.03)
Fusion-BM (0.54)			–	-0.14 (0.01)
Fusion-EM (0.69)				–

Note. nDCG@5 in brackets in the first column, adj. p-value in brackets behind difference

Table E.5

Mean difference nDCG@5 across models for "call option"

	Embedding	Fusion-EQ	Fusion-BM	Fusion-EM
BM25 (0.83)	0.40 (0.00)	0.06 (0.07)	0.03 (0.02)	0.15 (0.01)
Embedding (0.43)	–	-0.34 (0.00)	-0.37 (0.00)	-0.25 (0.00)
Fusion-EQ (0.77)		–	-0.03 (0.25)	0.09 (0.02)
Fusion-BM (0.80)			–	0.12 (0.02)
Fusion-EM (0.67)				–

Note. nDCG@5 in brackets in the first column, adj. p-value in brackets behind difference

Table E.6

Mean difference nDCG@5 across models for "ESG commitment"

	Embedding	Fusion-EQ	Fusion-BM	Fusion-EM
BM25 (0.19)	-0.24 (0.01)	-0.11 (0.02)	-0.09 (0.03)	-0.19 (0.01)
Embedding (0.44)	–	0.13 (0.03)	0.15 (0.01)	0.05 (0.16)
Fusion-EQ (0.31)		–	0.03 (0.28)	-0.08 (0.03)
Fusion-BM (0.28)			–	-0.10 (0.01)
Fusion-EM (0.38)				–

Note. nDCG@5 in brackets in the first column, adj. p-value in brackets behind difference

Table E.7

Mean difference nDCG@5 across models for "use of proceeds"

	Embedding	Fusion-EQ	Fusion-BM	Fusion-EM
BM25 (0.78)	0.31 (0.00)	-0.04 (0.25)	-0.05 (0.07)	0.12 (0.11)
Embedding (0.47)	–	-0.35 (0.00)	-0.36 (0.00)	-0.19 (0.00)
Fusion-EQ (0.82)		–	-0.01 (0.75)	0.16 (0.01)
Fusion-BM (0.83)			–	0.17 (0.01)
Fusion-EM (0.66)				–

Note. nDCG@5 in brackets in the first column, adj. p-value in brackets behind difference

Appendix F

Additional information on statistical tests for SF models

Table F.1
One-way ANOVA results for model comparison on SER

Information need	F-statistic	Raw p-value	Adjusted p-value
Aggregate level	39.53	0.00	0.00
ISIN	3.54	< 0.01	< 0.01
Coupon rate	11.51	0.00	0.00
Coupon periodicity	7.21	0.00	0.00
Call option	11.35	0.00	0.00
Use of proceeds	9.11	0.00	0.00
ESG commitment	11.75	0.00	0.00

Note. P-values adjusted using the Benjamini-Hochberg procedure with an FDR of 5%.

Table F.2
Micro-averaged metrics for all SF models

Model	Fine-tuning	SER	Precision	Recall	F1
Llama 3.1 8B Instruct	Zero-shot	1.45	0.33	0.56	0.41
	Few-shot	0.93	0.50	0.71	0.59
	LoRA	0.81	0.56	0.68	0.62
Gemini 2.5 Flash	Zero-shot	0.24	0.87	0.87	0.87
	Few-shot	0.22	0.89	0.88	0.88
	LoRA	0.14	0.92	0.92	0.92

Table F.3
Macro-average difference SER across models across information needs

	FS Gemini	FS Llama	LoRA Gemini	LoRA Llama	ZS Gemini	ZS Llama
FS Gemini	–	-0.77 (0.00)	0.10 (0.00)	-0.77 (0.00)	-0.02 (0.52)	-1.26 (0.00)
FS Llama		–	0.87 (0.00)	0.00 (1.00)	0.75 (0.00)	-0.50 (0.00)
LoRA Gemini			–	-0.87 (0.00)	-0.12 (0.00)	-1.36 (0.00)
LoRA Llama				–	0.75 (0.00)	-0.50 (0.00)
ZS Gemini					–	–
ZS Llama						–

Note. ZS = Zero-shot, FS = Few-shot

FACULTY OF BUSINESS AND ECONOMICS
Warmoesberg 26
1000 BRUSSEL, België
feb.leuven@kuleuven.be
www.feb.kuleuven.be

LID VAN ASSOCIATIE KU LEUVEN