ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

LEVERAGING LLMS AS NOISY LABEL GENERATORS FOR NAMED ENTITY RECOGNITION

CANDIDATE	SUPERVISOR
Antonio Lopez	Prof. Paolo Torroni
	CO-SUPERVISOR
	Andrea Zugarini

Academic year 2023-2024

Session III

Contents

List of Figures

List of Tables

Abstract

Large Language Models (LLMs) have been proven effective on various tasks due to their adaptability. Thanks to their reasoning [15] and in-context learning ability [7] several Natural Language Processing (NLP) tasks could now be taken into account by these models reaching good results without any training. One of the most famous and important tasks in NLP is Named Entity Recognition (NER), a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories. LLMs can effectively address this task when provided with a clear description and relevant examples [21] [24]. However, despite their strong performance, running these models requires substantial computational resources. In this case, smaller encoder-only models like GLiNER [27] or fine-tuned BERT-like [6] can ensure better performance at a lower cost. The main limitation of these models is that they are usually bound to the data seen in training while LLMs can easily adapt to different scenarios. In this work, a distillation pipeline is proposed to leverage the ability of large language models to act as label generators, creating synthetic data from unsupervised sources. This synthetic data is then used to distill smaller models capable of effectively replacing their teacher. The task addressed is Named Entity Recognition (NER) on the BUSTER dataset, which contains manually annotated financial transactions.

Introduction

Natural language processing is a wide field that covers different language tasks, from natural language understanding to question answering and entity recognition. The advent of transformer-based models has represented a breakthrough. In recent years LLMs like Llama[23], Mistral[11], Falcon[1] have grown in popularity and nowadays are used to solve several tasks. The main reason for their success is the ability of this model to successfully solve novel tasks without, in many cases, necessitating any fine-tuning. LLMs could be adapted to different scenarios by carefully engineering their prompt [15]. Specifically, methods like In Context Learning (ICL)[7] where task examples are shown as part of the prompt to enhance the model's ability by providing a limited number of instances of the problem. Thanks to this flexibility LLMs could be used as generators of noisy labels for unseen tasks and unlabeled datasets. Human labeling activity is slow and costly and deep learning models are data-hungry, labeling thousands of data samples requires a huge investment of time and resources. Here LLMs can provide valuable help since they are cheaper and faster than a human although the quality of their predictions may not match the same accuracy. To enhance this generation's ability it is also possible to limit the manual labeling to a small set and then use it to guide the model by prompting techniques like the already mentioned ICL. This labeling activity produces a novel synthetic dataset in which other models can be trained, at this point, LLMs can be replaced with smaller task-focused models (like BERT style models) at a fraction of their computational cost. This type of process can also be seen as a specific case of model distillation where the LLM is the teacher and the students are the models trained on its predictions. This work proposes a distillation pipeline for a specific study case. Specifically, LLMs would be used to train BERT-like models (<1B) by generating synthetic training data. The goal is to develop smaller models with comparable performance while significantly reducing computational costs. The addressed task is NER on BUSTER [29] a manually annotated dataset of financial transactions. Part of these annotations would be used to guide the generation process and to evaluate the quality of the generated labels and the distilled models.

This thesis aims to address the following research questions:

- **Large Language Models as label generators for NER in business transactions**: can large language models effectively produce high-quality annotations for named entity recognition in the technical domain of business transactions?
- **Optimizing annotation process using passage retrieval**: although LLM-generated annotations are faster and more cost-effective than human annotations, they can still require significant time. How can the annotation process be further accelerated without compromising quality using passage retrieval?
- **Distilling smaller models with noisy synthetic data**: how does the performance of distilled models trained on synthetic, noisy data compare to that of their teachers?
- **Impact of data quality on distillation**: how does the quality of training data influence the effectiveness of the distillation process and the performance of distilled models?

The work is organized into three main chapters:

- **Chapter 1**: introduces the necessary background, including the methods, models, and dataset used in the study.

- **Chapter 2**: focuses on the generation pipeline for synthetic annotations, detailing its design and implementation.
- **Chapter 3**: describes the training process for the student models, evaluates their performance, and compares the results with those of the teacher models, emphasizing the effectiveness of the distillation approach.

Chapter 1

Background

1.1 Named entity recognition

Named entity recognition is a natural language processing method that aims to extract and categorize information within a text [2]. This information is identified as named entities and represents key subjects in texts such as names, locations, dates, events, and topics. The set of named entities depends on the contexts and they are usually defined in advance. The main approaches for addressing these tasks can be categorized into three categories:

- **Features-based**: uses machine learning and statistical models. Supervised machine-learning approaches use algorithms like SVM [14] [17].
- **Rule-based**: information is extracted by using specific grammatical and statistical rules [20][19]. This approaches usually guarantee high precision due to the high human intervention required to tune and create the rules. This system has usually low recall due to the presence of rules.
- **Deep learning approach**: the use of neural networks, such as recurrent neural networks [5] and transformer architectures [22][13], to examine the syntax and semantics of sentence structures.

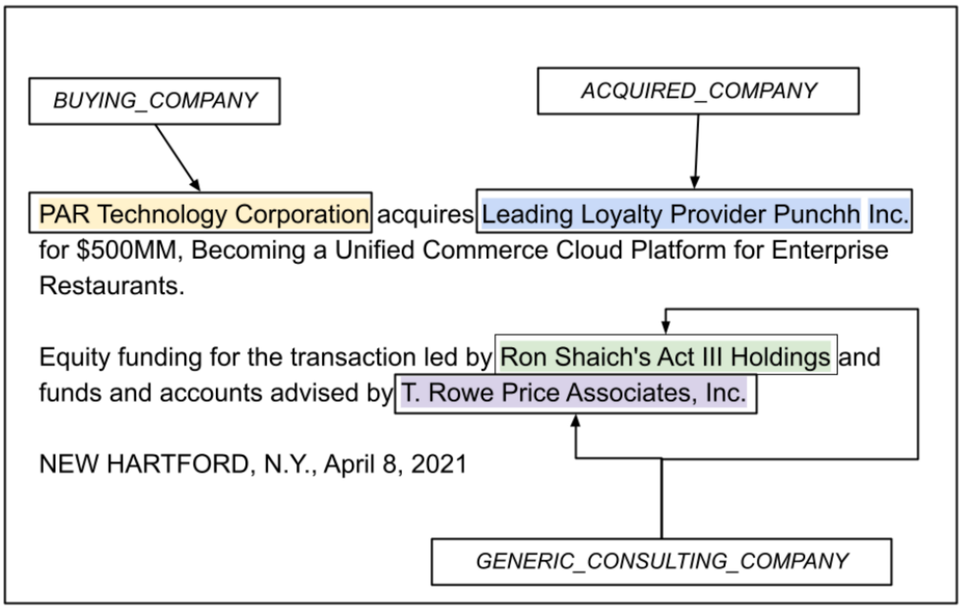
BUYING_COMPANY
ACQUIRED_COMPANY
PAR Technology Corporation acquires Leading Loyalty Provider Punchh Inc. for $500MM, Becoming a Unified Commerce Cloud Platform for Enterprise Restaurants.
Equity funding for the transaction led by Ron Shaich's Act III Holdings and funds and accounts advised by T. Rowe Price Associates, Inc.
NEW HARTFORD, N.Y., April 8, 2021
GENERIC_CONSULTING_COMPANY

Figure 1.1: An example of NER from BUSTER.

Named Entity Recognition is widely used in many other NLP tasks. Entity extractions provide valuable information that can be used for ChatBots, sentiment analysis, and information retrieval.

While large language models (LLMs) have made impressive strides, they still lag behind state-of-the-art supervised systems especially when there are abundant annotated examples [25][16]. However, for the vast majority of entity types, there is little annotated data. New entity types constantly emerge, and it is expensive and time-consuming to generate annotated examples, especially in high-value domains such as biomedicine where specialized expertise is required for annotation. To address these challenges recent methods such as UniversalNER[28], PromptNER [3], and Gollie [21] seek to leverage the reasoning ability of the LLMs to generalize on open-domain data using few-shot learning and carefully engineering the prompt.

1.2 Knowledge distillation

Knowledge distillation, introduced by Hinton et al. in 2015 [10], is a technique for transferring knowledge from a large, complex model (teacher) to a smaller, simpler model (student) while maintaining performance. They proposed using soft targets-probability distributions output by the teacher model over the class labels as additional supervision for the student model. These soft targets, generated by applying a temperature scaling factor to the teacher's logits, capture richer information about inter-class relationships than hard labels alone. By training the student to mimic the teacher's softened output, the student model can generalize better, even with fewer parameters. Knowledge distillation has since become a foundational method for reducing computational costs and deploying efficient models in resource-constrained environments.

In recent years, the rise of generative AI has led to new possibilities, particularly in generating large quantities of data at a relatively low cost. This capability has paved the way for a novel approach to knowledge distillation, where generative models are employed to create synthetic datasets tailored for specific tasks by leveraging their vast knowledge. These synthetic datasets can address the challenges of data scarcity, reducing the need for expensive and time-consuming manual annotation. These data are then used to train smaller, task-focused models that can effectively substitute their larger counterparts. Furthermore, the ability to generate task-specific synthetic data allows for the customization of smaller models to meet the unique requirements of diverse applications, enhancing their overall utility and efficiency.

Large language models, pre-trained on extensive corpora and comprising billions of parameters, have demonstrated remarkable flexibility and success in solving diverse problems. LLMs distillation [26] utilizes these generative models as teachers to generate annotations from unlabeled data [18][9]. These synthetic labels are then used to train smaller student models, enabling them to replicate the teacher's predictions. While student models offer a more practical and resource-efficient alternative, their performance is inherently tied to that of the teacher model. Consequently, careful engineering of the synthetic data generation process is crucial to ensure high-quality annotations and maximize the benefits of distillation.

1.3 GLiNER

GLiNER is a novel model that addresses the Open NER task by using small Bidirectional Language Models (BiLM), such as BERT or deBERTa [8]. The core idea is to address the task as the matching of *entity type* embeddings to *textual span* representations in latent space, rather than as a generation task like the majority of the models [21] [28]. GLiNER architecture (Figure 1.2) is composed of three main component:

- Pre-trained textual encoders like BERT and deBERTa.
- Span representation module that computes span embeddings from token embeddings.
- An entity representation module that computes entity embeddings that the model seeks to extract.

With this architecture, the span and the entity embeddings are represented in the same latent space, their compatibility (degree of matching) represents the probability for a certain span to match a certain entity type. The input of the model is represented by a single sequence that combines entity types and input text. The two subsequences are delimited by the [SEP] token while each entity type is associated with the special token [ENT]. During training, GLiNER objective is to maximize the scores between the correct span-type pairs (positive pairs) and reduce for the incorrect pairs (negative pairs). Negative pairs are created by randomly sampling entity types that are not present

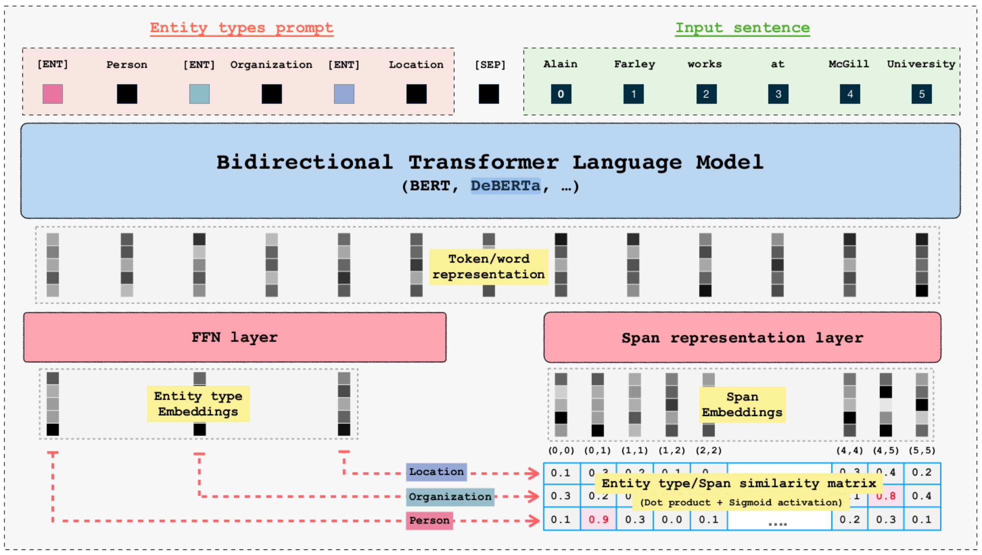

Figure 1.2: GLiNER employs a BiLM and takes as input the entity types and the target text. Each entity is separated by a learned token [ENT]. The BiLM outputs representations for each token that will be used to compute the similarity between spans of text and entities.

in the current sample. This is done for a better alignment between training and real-world scenarios, where some entity types might be absent.

An essential consideration when training GLiNER is how to handle empty samples, as these lack any positive pairs. For the entity input sequence, using the same input (like the full set of entities) for empty samples can lead to a performance drop, especially if many are present in the training set. GLiNER's architecture may learn a simple association between a static input sequence and the absence of entities if empty passages dominate the data. This association could become problematic during inference when the same input sequence is provided. To mitigate this issue, one approach is to sample a random subset of entity types as the input sequence for empty texts. However, if the variety of entity subsets is limited or if empty samples make up a large portion of the data, this approach might not be sufficient. In these cases, balancing the number of positive and negative samples is effective in addressing the problem. GLiNER outperforms state-of-the-art large language models like ChatGPT in zero-shot scenarios but also offers a more resource-efficient alternative, crucial for environments with limited computing power. GLiNER is versatile, performing well in multiple languages, including those it wasn't trained on.

1.4 BUSTER

BUsiness Transaction Entity Recognition [29] is an Entity Recognition (ER) benchmark that focuses on the main actors involved in a business transaction. The dataset is composed of about 3779 manually annotated documents from the Electronic Data Gathering, Analysis, and Retrieval system (EDGAR) reporting business transaction documents and acquisition reports. Specifically, the gathered document belongs to a specific family of the ones present on EDGAR called *Exhibit 99.1*. It consists of a disclosure document that summarizes all the details of the operation announced in the form, and it is designed to provide investors with a complete and detailed view of the operation. BUSTER identifies three families of tags (the full tag-set is reported in table 1.1):

- **Parties** which groups tags used to identify the entities directly involved in the transaction.
- **Advisors** which groups tags identifying any external facilitator and advisor of the transaction.
- **Generic_Info** which identifies tags reporting any information about the transaction.

BUSTER annotations were designed with a strong emphasis on contextual clarity. Entities are tagged only when their class is explicitly clear from the linguistic context, ensuring that their role is unambiguous. Each sentence is evaluated independently from others in the document, meaning that an entity identified in one sentence is not automatically considered in subsequent occurrences unless the context supports it.

Tag Family	Tag Name	Description
Parties	BUYING_COMPANY	The company which is acquiring the target.
	SELLING_COMPANY	The company which is selling the target.
	ACQUIRED_COMPANY	The company target of the transaction.
Advisors	LEGAL_CONSULTING_COMPANY	A law firm providing advice on the transaction, such as government regulation, litigation, anti-trust, structured finance, tax, etc.
	GENERIC_CONSULTING_COMPANY	A general firm providing any other type of advice, such as financial, accountability, due diligence, etc.
Generic_Info	ANNUAL_REVENUES	The past or present annual revenues of any company or asset involved in the transaction.

Table 1.1: Description of the tag-set defined in BUSTER.

The dataset is randomly split into 5 folds, and the experiments conducted use the following:

- **FOLD 1**: this fold contains annotated examples and serves as a foundational set. Samples from this fold are used to guide the generation process, providing the LLMs with valuable examples to support annotation generation.
- **FOLD 2**: this fold is the target set for the generation process. Documents in this subset are presented to the LLM which will generate synthetic annotations. This fold will also represent the training dataset for the student models with two variants of labels: synthetic and gold.
- **FOLD 3**: this fold functions as the final test set for the distilled model. To ensure a fair comparison between the student and teacher models, the LLMs are also evaluated on this set. Annotations generated for this fold are excluded from the distillation process.

1.4.1 Statistics

Documents in BUSTER (Figure 1.3) have a length of around 700 words and most of them fall into the 500-1000 range. Also, documents with more than 2000 words are extremely rare.

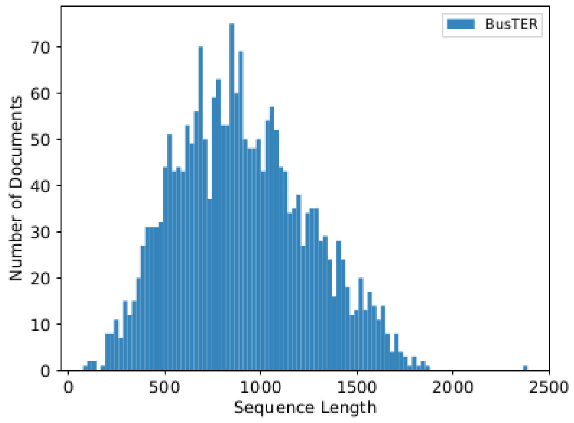

Figure 1.3: Document's length distribution in terms of words.

In Figure 1.4, it is reported the distribution of the three tag families based on their position in the text. The tags belonging to the Parties family (in orange) are centered in the initial parts of the documents, while the remaining are distributed more uniformly. No tags occur beyond the 1500th word.

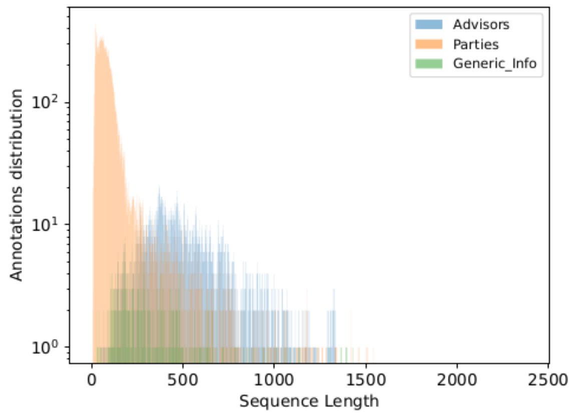

Figure 1.4: Distribution of tags families inside the documents.

Chapter 2

Synthetic data generation

Large language models are powerful tools that require careful handling to perform optimally. Simply asking the model to solve a task often results in incomplete or inadequate outcomes. The prompt plays a crucial role, a well-crafted one is a good starting point but it may not be sufficient on its own. It is also essential to leverage all available information to guide the model and make its task easier. To effectively obtain high-quality labels a novel **generation pipeline** is proposed (2.1). It comprises different pieces that could be modified according to the current task. The components of the pipeline are the following:

- **Chunker**: divide the document into passages, crucial for having chunks that contain all the contextual information needed to identify entities.
- **Retriever**: receives the passages of the document and selects which of these potentially contains entities, its goal is to filter out empty passages.
- **Example generator**: generate dynamic demonstration according to the current chunk. These examples are included in the prompt and serve as a guide.
- **LLM**: for each passage received generate a set of annotations.

- **Annotated Examples**: set of already annotated documents, used for retrieval and generating examples.

The different pieces work in cooperation with each other but are essentially independent, they can be modified and removed without compromising the functionality of the others. During the experiments, different combinations of the pipelines were tried by removing and trying different versions of the same component. The only essential piece is the language model which must be always present since it is the one producing the annotations. The novel dataset is created by the documents and the synthetic annotations.

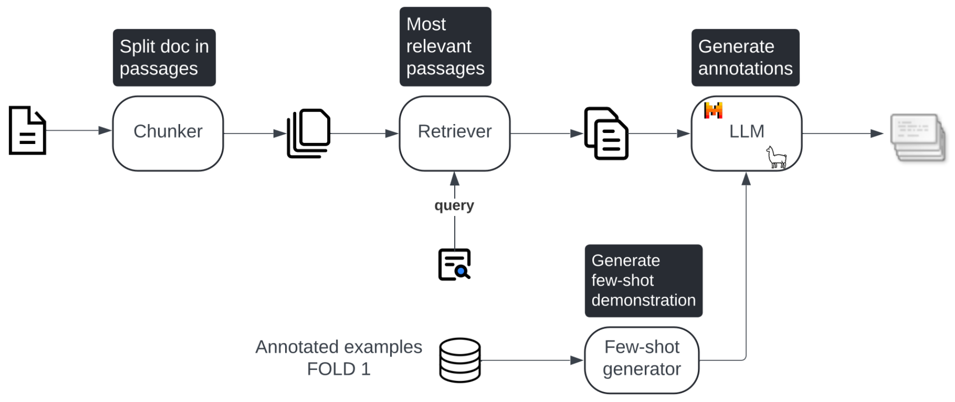

Figure 2.1: Pipeline for annotation process. Each document is initially divided into several chunks that will be filtered by the retriever. The unfiltered chunks are then passed to the LLM alongside the generated demonstration. At the end of the pipeline, a new set of annotations is created.

2.1 Document chunking

Choosing the correct chunking method is crucial since it is the first step of the pipeline and will affect all the others. Furthermore, the context provided to the LLM is decided at this step, the model should have all the information needed to perform the recognition without being distracted by unnecessary context. Different chunking strategies have been tried from hard chunking once reached a specific token number to more refined methods like paragraph splitting, which tries to keep all paragraphs (and then sentences, and then words) together as long as possible. The chunker model chosen is a **sentencizer** which divides texts into sentences. The main advantages of this approach are the following:

- The entities in BUSTER are local and bounded to their context this means that an entity is annotated only in the parts of the texts where it actively holds that role. The same entity a few sentences after is no longer considered as such.
- Once the entities are recognized it is necessary to find their position in the text. LLMs often struggle with generating accurate positional information. For this reason, smaller sentences make the entity search easier and a simple matching in the text could be sufficient.

On the other hand, this approach tends to generate a significant number of chunks (about 23 per document) which could represent a sensible slowdown compared to other methods.

2.2 Prompting

Prompt engineering plays a pivotal role when working with language models. A well-structured prompt can significantly influence how the model interprets tasks and the results' quality.

Initially, a single prompt was employed to handle all entities. However, the model struggled with this approach due to the prompt's excessive length and high information density. Incorporating examples presented another challenge, as each class required at least one example, further lengthening the prompt and making it harder to interpret. This added complexity increased the chances of confusing the model with examples from different classes. For this reason, three different shorter and specific prompts were created, one for each tag family 1.1. The three prompts share a similar structure but are tailored to fit the specific characteristics of each entity family. This approach allows for more precise descriptions and the inclusion of additional examples without overloading the prompt with excessive or mixed-class information.

The family-based prompt division proves advantageous by narrowing the model's focus to a subset of entities at a time. The entities within each family often exhibit similarities, and by presenting them together, the model is compelled to differentiate between closely related entities. This strategy helps the model focus on subtle distinctions, typically where entity confusion arises.

Another important refining to the prompt is the addition of the dummy class *OTHER* to all the prompts. This class acts as a backup for the model when it is not sure about its predictions, it also acts as another way to define what it is not an entity and it turns out to be beneficial to the performance.

The prompt used during the following experiment is composed of four different parts:

- **Task introduction**: the first part of the prompt explains the task and describes the nature of the text given as input.
- **Classes definitions**: the second part introduces the classes to be identified, each class description contains:
  - **Definition**: a brief description of the class describing what is the characteristic that identifies an entity belonging to that class.
  - **Guidelines**: instruction on how the entity should look to be part of the class, special cases, and also when an entity should not be considered.
- **Output format**: contains the format of the generated labels and which information should be inserted. The two pieces of information required are the **entity type** and the **entity name**. The chosen format is JSON string.

- **Examples**: in this section, examples are shown to help the model understand the task and the desired output.

Despite the advantages of this tag-wise approach, there are some drawbacks that it is important to mention:

- **More LLM calls**: three different prompts imply as well three calls to the language for each passage making the process slower by an $\times 3$ factor.
- **Conflicting predictions**: another issue arises when the language model assigns the same entity two different entity types across different prompts. In a single-prompt scenario, this was unlikely, but with multiple prompts, it becomes a problem as correct predictions may be overwritten by incorrect ones.

Annual revenues prompt

You are a helpful assistant. Your task is to recognize entities in the given text. Specifically, you will see chunks of documents containing information statements about company acquisitions, ownership changes, share purchases, and public announcements.

The entities you must recognize are:

- ANNUAL_REVENUES

DEFINITION: refers to the past or present total income earned by a company or entity within a specific fiscal year.

GUIDELINES: Do not annotate generic terms such as `revenue`, but the numerical amounts with the currency symbol, for example, `"$120 million"`. Make sure to distinguish projected revenues from present and past revenues. Only annotate actual, past or present revenues. Do not annotate future revenues indicated with verbs like `"will generate"`, `"expect"`, and `"project"`.

- OTHER

DEFINITION: `"other"` refers to entities that do not fit into any of the other categories.

GUIDELINES: Use this category for all the other amounts like net proceeds, cash on hand, debt, purchase prices, operating losses, sales prices, projected revenues from acquisitions, asset values, and other financial amounts.

Perform entity recognition with utmost strictness; only provide an answer if you are absolutely certain of its accuracy, otherwise, refrain from giving an answer.
You must produce a json file containing the recognized entities, for each entity you must specify the following:

- The entity name
- The entity type

Here some examples to help you:

```
EXAMPLE 1
{text}
{entities}
...
```

What are the entities in this text:

{input text}

Figure 2.2: Prompt for the *Generic Info* tag family, all the entities in the family are listed along with the additional *OTHER* class. Each entity presents its own definition and guidelines.

2.3 Passage retrieval

The retrieval component acts as a filter over document chunks, selecting the most relevant passages and discarding sections where entities are unlikely to be found. This approach accelerates computation and is expected to improve overall generation quality. Specifically, it aims to solve and mitigate the following problems:

- **Too many chunks**: documents in BUSTER are about 700 words long, and the sentencizer used as chunker produces about 23 chunks per document.
- **Three prompts for passage**: with the current prompting to process an entire document 69 calls are needed on average per document.
- **Most chunks are empty**: the entities are densely present in the documents' initial and middle parts (Figure 1.4), and 9/10 of the created passages do not present any entity.
- **LLM calls are slow and costly**: in an experimental scenario where several trials and tests are performed the costs and time needed become unsuitable.

This section presents various retrieval methods along with their corresponding results. All retrieval approaches were evaluated on FOLD_2, with FOLD_1 as the training set when applicable. The primary focus of the retriever is on maximizing recall rather than precision. Since the filtered passages will not be passed to the language model, achieving high recall is crucial. Although precision values may be low, this is secondary, as the model is still able to disregard irrelevant chunks that do not contain entities.

Retriever's role is to select which passages from a document could contain an entity going to filter out all the nonpromising chunks. It does not explicitly perform any type of classification but in a certain way, it guides the classification since it decides which passages are suitable for which prompt.

2.3.1 Tag-wise query

A query-based retriever is a system designed to retrieve relevant information or documents from a large dataset in response to a specific query. In this case, the content of the query is to search through the chunks of a document and identify the most relevant passages by comparing the semantic similarity between the query and the available information. The goal of a query-based retriever is to efficiently surface chunks that are most likely to match the corresponding query, prioritizing recall to ensure that no potentially relevant information is overlooked. Three different queries are defined, one for each tag family:

- **Parties**: *Which are the companies sales and acquisition?*
- **Advisors**: *Which consulting firm are advising the transaction?*
- **Generic Info**: *Which are the annual revenues?*

These queries are then compared with the passages using a cross-encoder and from each query, the top k passages are selected. A passage could be chosen by different queries and each query has a different value for k

From table 2.1 it is clear how the query approach is particularly effective on the revenues while struggling more on the parties. Also, the consulting firm reports a good recall. However, it is clear that to have a high recall value precision must be sacrificed. The main problems of this retriever are the classic ones for the query based:

- **Query tuning**: tuning the query is crucial for achieving the best performance.
- **K tuning**: k values should be tuned to achieve the best performance. Furthermore, setting also a similarity threshold can be important to avoid retrieving a fixed number of chunks even though the similarity is low.

Tag-family	Precision	Recall	F1	k
Parties	39.39	19.66	26.23	1
	37.57	37.48	37.52	2
	35.56	53.19	42.62	3
	31.85	63.45	42.41	4
	28.65	71.27	40.87	5
Advisors	14.36	46.19	21.91	1
	10.88	69.92	18.82	2
	8.44	81.36	15.29	3
	6.73	86.44	12.49	4
	5.58	89.41	10.50	5
Generic Info	12.65	73.28	21.57	1
	7.84	90.84	14.44	2
	5.54	96.18	10.47	3
	4.22	97.71	8.10	4
	3.41	98.47	6.59	5

Table 2.1: Retrieval results for the tag-wise query method varying the top k. Each query is associated with a tag family and it is used to retrieve from the document's chunk the candidates for that tag set.

2.3.2 Similar passage retrieval

Similar Passage Retrieval (SPR) is a novel method that uses the annotated dataset to filter out unnecessary chunks. The idea is to evaluate a chunk by comparing it to similar ones. If similar texts contain entities, it's likely the evaluated passage will as well; conversely, if none of the similar passages have entities, the chunk can be filtered out.

This data-driven approach aims to remove some of the drawbacks of tag-wise query retrieval, starting from the query tuning. Since the passage itself is the query, the tuning process is completely absent making this method more robust and easier to adapt to other contexts or datasets. Additionally, the search of *k* becomes less impelling, since the SPR is capable of selecting fewer chunks than the chosen *k* without setting any type of threshold.

The retrieval process 2.3 is executed document per document, three candidate lists are present, one for each tag family. Each passage of the document is compared with the most similar examples from the corpus. After, the passage is added to a candidate list if there is at least one entity of the corresponding type in one of the similar examples. A passage could be inserted in different lists if entities from different tag families are present. To sum up, for each passage in a document:

- The most similar k passages are retrieved from the annotated examples. The retrieval is performed using the cosine similarity between the embedded passage and the embedded corpus of the examples
- Check if any retrieved examples present an entity.
- If an entity is found, add the passage to the corresponding list using as a score the similarity of the passage containing that entity, otherwise discard the passage.
- From each candidate list return the top k passages.

The list insertion policy could also be different from the one proposed here. A possible variant could be a voting scheme where each example expresses a positive vote if presents an entity otherwise a negative one, everything weighted by the similarity score.

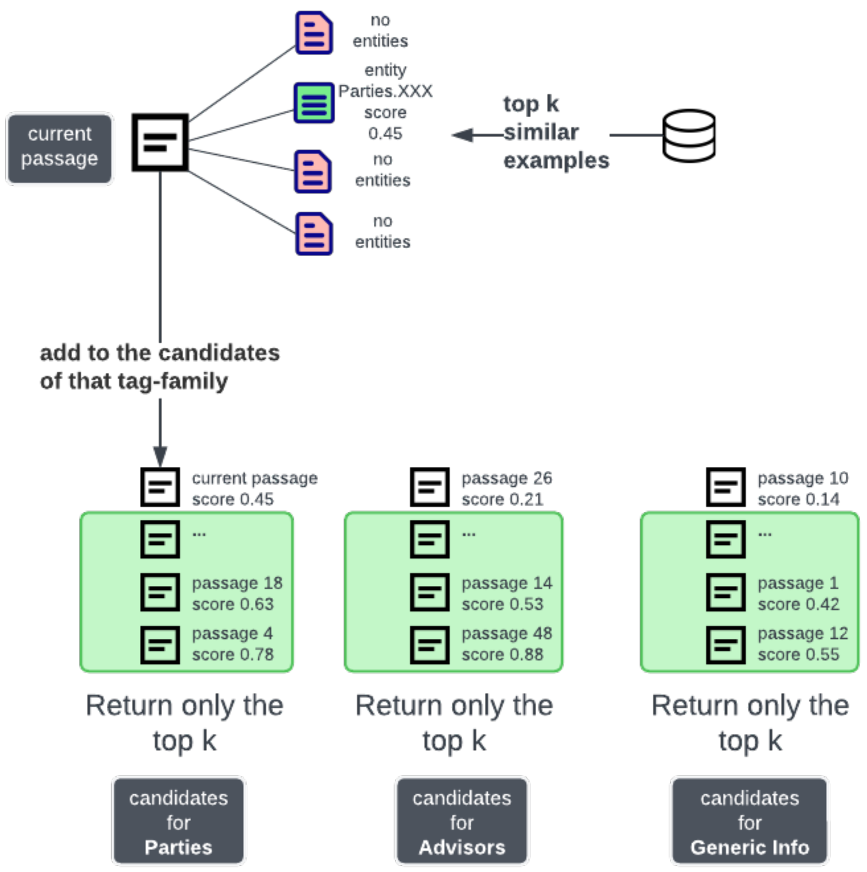

Figure 2.3: In this scenario the current passage is added to the candidate's list for *Parties*, due to the presence of an entity of that type in one of the similar examples. However, at the end of the process, it would not be selected since there are other chunks in the list with better scores.

Tag-family	Precision	Recall	F1	k
Parties	86.04	39.32	53.97	1
	65.40	54.17	59.26	2
	52.61	62.33	57.06	3
	44.79	67.32	53.80	4
	39.12	70.55	50.33	5
Advisors	72.99	42.37	53.62	1
	61.48	66.95	64.10	2
	52.03	75.85	61.72	3
	46.45	83.05	59.57	4
	41.61	85.17	55.91	5
Generic Info	44.44	39.69	41.94	1
	38.79	63.36	48.12	2
	31.21	70.99	43.36	3
	26.72	74.05	39.27	4
	23.68	78.63	36.40	5

Table 2.2: SPR results with different values of the top k passages selected for each list. The number of examples retrieved to evaluate the passages is set to four.

2.3.3 Similar passage retrieval masked

A potential variation of the SPR is the masked version. While the overall process remains unchanged, the key distinction lies in the masking of entities within the corpus. This masking aims to enhance retrieval by making the process independent of company names. The presence of these names could skew the similarity retrieval, leading it to favor examples based on matching company names rather than true semantic relevance.

Tag-family	**Precision**	**Recall**	**F1**	**k**
	76.20	36.62	49.47	1
	57.36	51.22	54.12	2
Parties	46.22	59.57	52.05	3
	39.79	65.94	49.63	4
	35.24	70.74	47.05	5
	67.50	34.32	45.51	1
	58.05	58.05	58.05	2
Advisors	46.37	70.34	55.89	3
	40.49	76.69	53.00	4
	36.12	80.51	49.87	5
	42.61	37.40	39.84	1
	36.61	62.60	46.20	2
Generic Info	30.60	74.05	43.30	3
	26.28	78.63	39.39	4
	23.63	82.44	36.73	5

Table 2.3: SPR masked results with different values of the top k passages selected for each list. The number of examples retrieved to evaluate the passages is set to four.

2.4 Evaluation

2.4.1 Method

From the result shown in 2.5 and 2.4 the SPR in both its variants outperforms the query method in all the metrics. SPR achieves a better level of precision with a higher recall and a higher percentage of work saved. Even modifying the k value for the query method to have an exact match in recall would have anyway a lower precision making it a less preferable method.

	Micro				Macro			
Method	Precision	Recall	F1	Ws	Precision	Recall	F1	Ws
spr	**42.09**	**70.07**	**52.59**	**85.27**	**38.31**	**76.33**	**51.01**	**94.21**
spr masked	37.72	68.43	48.64	84.17	34.64	75.03	47.39	93.69
query	20.66	66.37	31.51	74.73	17.65	72.69	28.40	88.83

Table 2.4: Overall retrieval results for each method presented.

	Parties				Advisors				Generic info			
Method	Precision	Recall	F1	Ws	Precision	Recall	F1	Ws	Precision	Recall	F1	Ws
spr	**44.79**	**67.32**	**53.80**	**87.37**	**46.45**	**83.05**	**59.57**	**97.67**	**23.68**	78.63	36.40	**97.60**
spr masked	39.79	65.94	49.63	86.07	40.49	76.69	53.00	97.53	23.63	**82.44**	**36.73**	97.47
query	31.85	63.45	42.41	83.26	8.44	81.36	15.29	87.43	12.65	73.28	21.57	95.81

Table 2.5: Retrieval results for each tag family.

2.4.2 Masking

The standard SPR performs better for all families except for *Generic Info*, which has the same results. Masking the entity removes the influence of the company names on the similarity score, and that could be the reason for the performance loss on the *Advisors* and on *Parties*. The most significant drop in performance is observed with the *Advisor* tags. Companies engaged in advising activities represent a small subset of the dataset, and whenever they appear in a document, they are likely to fulfill a consulting role. Masking their names results in a loss of information that, in most cases, would have been crucial for retrieving relevant passages. This problem is not present in the *Generic Info* family which only contains the class *ANNUAL_REVENUES*. Since the numerical value, is unlucky equal between different contexts, masking this entity does not bring any valuable information loss. Indeed seems that for this class the masking is beneficial, from table 2.5 it is clear that with the same precision level, SPR masked achieves a better recall.

2.5 Few-shot generator

Large language models have shown remarkable benefits from learning by example. Limited but high-quality demonstrations showing how the model should accomplish the task can dramatically improve the performance of the model. In the experiments, the examples would be provided in three different ways:

- **Few-shot**: traditional few-shot learning using handcrafted examples.

- **Dynamic few-shot learning**: select examples from a pool of options based on the current passage.
- **Dynamic few-shot learning masked**: select examples with masked entities from a pool, based on the current passage.

For all these methods the same number of examples has been used for a fair comparison.

2.5.1 Few-shot

The few-shot method involved crafting specific examples for each tag family and its associated entities. These examples were derived from challenging cases that the model struggles to annotate. While the entity names are fictional, the text and structure closely resemble actual instances. Additionally, examples using the class *OTHER* were included to encourage the model to select it when no relevant entity is found, helping it understand that not all company names should be treated as entities.

2.5.2 Dynamic few-shot learning

Dynamic Few-Shot Learning (DFSL) is a few-shot technique that dynamically adapts demonstrations based on the current input [12][4]. This approach uses a pre-labeled pool of examples, selecting those most similar to the current passage, and making the prompt more flexible by tailoring specific examples to specific cases. The DFSL process involves the following steps:

- The chunked documents present in the FOLD_1 are embedded using a sentence encoder, which, in this case, is the same as the one used by the SPR.
- The current passage is used to retrieve the most similar passages from the annotated set.

- The prompt is enriched with these examples and their corresponding entities, formatted according to the model's expected output.

It is important to note that when both SPR and DFSL are applied, the examples used for retrieval are the same as those inserted as dynamic examples.

2.5.3 Dynamic few-shot learning masked

This variant of DFSL masks the entity names in the demonstration provided to the LLM. Similar to the approach used in retrieval, the presence of company names might lead the model to simply copy the provided examples rather than using them as guidance. Masking the entity names aims to prevent this by ensuring the model relies on the context rather than the names themselves.

2.6 Experimental setup

The experiments involved testing various models and configurations of the pipeline components to determine which combinations performed best and whether the proposed methods improved the generation process. All experiments used the same tag-specific prompt, with an equal number of examples across configurations where applicable. Masked configurations use masked variants for both SPR and DFSL. From the possible combinations, only the most relevant setups were retained for comparison:

- **Model**:
  - Llama 3.1 8B
  - Mixtral 8x7B
  - Llama 3.1 70B
- **Examples**:
  - Zero-shot
  - Few-shot
  - DFSL
- **Retriever**:
  - SPR
  - Query based
  - None
- **Masking**:
  - Yes
  - No

2.7 Quantitative results

The reported metrics were all calculated on FOLD_2 of the BUSTER dataset. In addition to the standard micro and macro scores, the total number of calls to the language model is provided to better assess the retriever's performance and the time saved. Detailed single-class metrics are available in the appendix A.

					Micro			Macro		
Model	Examples type	Retrieval	Masked	Llm calls	Precision	Recall	F1	Precision	Recall	F1
Mixtral 8x7B	few-shot	spr		3143	**51.95**	52.64	**52.29**	**58.32**	60.93	**59.31**
Mixtral 8x7B	few-shot	spr	✓	3425	48.94	49.96	49.45	57.80	57.77	57.54
Mixtral 8x7B	dfsl	spr		3143	42.01	54.93	47.61	48.97	62.75	54.81
Mixtral 8x7B	dfsl	spr	✓	3425	39.62	52.40	45.13	48.68	61.07	53.95
Mixtral 8x7B	zero-shot			54288	25.20	**62.10**	35.85	31.48	**63.92**	41.11
Mixtral 8x7B	zero-shot	spr		3143	45.85	49.94	47.81	50.82	54.63	52.35
Mixtral 8x7B	zero-shot	spr	✓	3425	42.98	47.97	45.34	49.78	52.75	50.82
Mixtral 8x7B	zero-shot	query		6064	37.69	46.12	41.48	42.18	50.72	45.73

Table 2.6: Summary results for Mixtral 8x7B model.

					Micro			Macro		
Model	Examples type	Retrieval	Masked	Llm calls	Precision	Recall	F1	Precision	Recall	F1
Llama3.1	few-shot	spr		3143	53.04	50.86	51.93	56.68	59.41	56.66
Llama3.1	few-shot	spr	✓	3425	51.04	49.66	50.34	55.76	57.94	55.68
Llama3.1	dfsl	spr		3143	41.55	**56.39**	47.85	47.32	**64.51**	53.84
Llama3.1	dfsl	spr	✓	3425	39.77	52.31	45.19	45.90	60.71	51.28
Llama3.1	zero-shot			54288	33.08	54.30	41.11	29.80	51.24	37.17
Llama3.1	zero-shot	spr		3143	47.62	46.66	47.14	45.65	46.55	45.18
Llama3.1	zero-shot	spr	✓	3425	45.39	44.69	45.04	45.09	44.20	43.83
Llama3.1	zero-shot	query		6064	43.29	42.28	42.78	39.46	46.38	41.76
Llama3.1 70B	few-shot	spr		3143	55.98	52.73	54.31	62.73	62.03	61.99
Llama3.1 70B	dfsl	spr		3143	**57.22**	55.33	**56.26**	**63.56**	63.84	**63.51**
Llama3.1 70B	zero-shot	spr		3143	54.21	53.29	53.75	60.69	62.88	61.64

Table 2.7: Summary results for Llama 3.1.

2.7.1 Passage retrieval

Models utilizing retrieval show better performance with fewer language model calls. Despite the detailed recommendations in the prompt, the models tend to be more lenient in their predictions, favoring recall but significantly reducing precision. SPR yields the best results and is extensively used in other experiments.

2.7.2 Models

Despite having a big difference in the number of parameters (8 vs 45 billion) Llama and Mixtral show comparable results in all configurations. Notably in the zero-shot setups without retrieval, Llama seems to have better followed the prompt because is more conscious in his predictions showing more balanced values for precision and recall. On the other hand, Mixtral is more balanced between different classes reporting better macro scores. As expected, Llama 70B shows the best performance and outperforms the other models in all the configurations.

2.7.3 Examples type

Few-shot and DFSL methods demonstrate better macro performance, as the inclusion of examples aids the model in handling more challenging classes. Specifically, *ANNUAL_REVENUES* and *SELLING_COMPANY* benefit the most from this approach. However, this improvement is less evident in the micro scores due to the limited number of examples for these classes.

Dynamic few-shot learning outperforms traditional few-shot only when used with Llama 70B. Smaller models seem unable to fully leverage this method, favoring simpler few-shot approaches instead. This may be because similar examples could confuse smaller models, causing them to mix up the examples with the actual query, whereas larger models can better distinguish between the two parts of the prompt.

2.7.4 Masking

The masking had the goal of making the models focus more on the sense of the passage rather than the companies involved in the action. This approach does not help either the retrieval as shown in 2.4 or the generation's results. For both models and all the different configurations, the masked setups perform worse compared to the standard one.

Chapter 3

Distillation

In this section, the LLMs used in the last chapter will act as teachers for smaller student models (<1B). The distillation process aims to obtain a smaller model that performs similarly to the bigger one with a fraction of the cost. Specifically, the synthetic data generated by the LLM will be used as training data while the students will try to imitate this prediction. Since the labeled data are noisy the training process would be crucial for reducing the noise and achieving at least comparable performance. Training results are strictly related to the quality of the generated data. From all the possible models presented, two teachers have been designed:

- **Llama 70B DFSL with SPR**: the best performing model. The data generated by this model would be referred to as **SILVER** labels
- **Mixtral 8x7B DFSL with SPR**: a less performing model used as a comparison to understand how much label quality will impact the distillation process. This model has about nine points less of micro F1 and its generated data would be referred to as **BRONZE** labels

Different student models are used to assess how various architectures adapt to synthetic data, identifying which models are more sensitive or more robust.

The two student models tested are:

- **BERT**: a common baseline for many NLP tasks. A classification head is added on top to perform token-level classifications.
- **GLiNER**: a zero-shot entity extractor built on a bidirectional transformer encoder.

BERT serves as a standard baseline for NLP tasks, and its pre-trained version provides a strong foundation for feature extraction, allowing the classification head to be trained from scratch for token classification. In contrast, GLiNER is a more recent model designed specifically for entity extraction. Unlike BERT, it doesn't require additional components and can be fine-tuned as-is. Its pre-trained version is already tailored for NER tasks, making it more specialized than BERT for entity recognition.

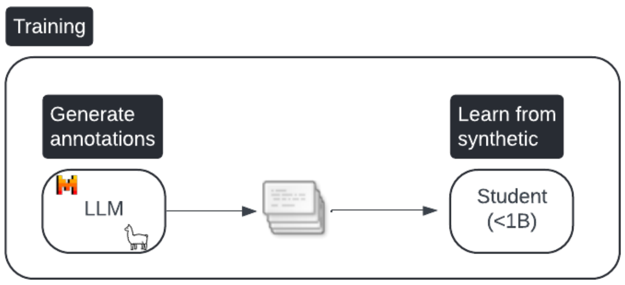

Figure 3.1: The student model uses synthetic data generated by the LLM as the training set.

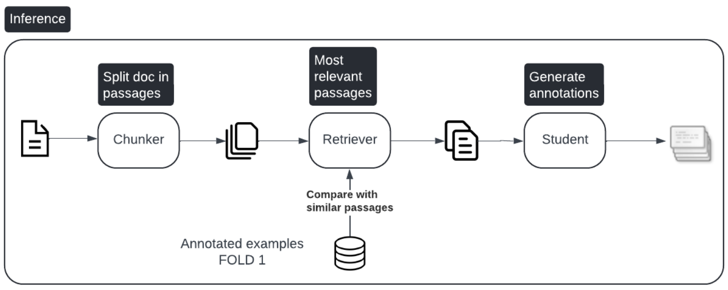

Figure 3.2: Student at inference time. A retrieval step can be added to filter out chunks as done in the synthetic generation.

3.1 Experimental setup

The models were trained for twenty epochs, GLiNER used a focal loss while BERT a weighted cross-entropy for dealing with the class unbalance. GLiNER is fine-tuned using the hyperparameters from the authors' repository. GLiNER faced a significant challenge due to the high number of passages without entities. GLiNER's input consists of two parts: the first contains the possible entities, and the second includes the input text. Since the entities to be recognized remain the same, the left part of the input stays unchanged. With nine out of ten passages lacking entities, the model quickly formed a bias, associating the label set with the absence of entities. To mitigate this problem, a random subset of the possible labels was sampled for passages without entities. While this helped, the model still struggled with the overwhelming number of empty passages. The training set was adjusted to balance the number of positive and negative passages, forcing the model to learn proper entity recognition without relying on shortcuts. Additionally, BERT was also trained with a configuration where the number of positive and negative passages was balanced. This adjustment was made to evaluate whether BERT could similarly benefit from this approach and improve its ability to distinguish between passages with and without entities. Here are the different setups:

- **Model**:
  - BERT
  - GLiNER
- **Training Data**:
  - Gold
  - Silver
  - Bronze
- **Negative Ratio**:
  - Original
  - Balanced
- **Retrieval**:
  - Yes
  - No

3.2 Quantitative results

The reported metrics are computed on the FOLD_3 of BUSTER. The overall micro and macro results for BERT and GLINER are shown in 3.1 and 3.2. Specific per-class metrics are presented in appendix B.

				Micro			Macro		
Model	Training data	Retrieval	Negative ratio	Precision	Recall	F1	Precision	Recall	F1
BERT	gold		balanced	41.60	**78.19**	54.30	45.95	**79.43**	57.38
BERT	gold		original	61.83	71.32	**66.24**	60.75	73.08	**66.16**
BERT	gold	spr	balanced	54.44	67.21	60.15	55.46	70.94	61.70
BERT	gold	spr	original	**64.67**	64.30	64.48	**63.40**	67.33	65.17
BERT	bronze		balanced	25.02	75.50	37.59	33.79	77.17	45.70
BERT	bronze		original	45.93	66.43	54.31	46.52	67.45	54.61
BERT	bronze	spr	balanced	42.12	63.74	50.72	46.95	68.19	55.09
BERT	bronze	spr	original	52.63	61.17	56.58	51.75	62.89	56.34
BERT	silver		balanced	35.88	77.73	49.09	39.29	79.28	51.93
BERT	silver		original	57.83	64.50	60.98	59.79	67.58	63.35
BERT	silver	spr	balanced	48.61	66.75	56.25	50.02	70.77	58.22
BERT	silver	spr	original	59.67	60.42	60.04	60.86	64.80	62.68

Table 3.1: BERT summary results on different configurations.

				Micro			Macro		
Model	Training data	Retrieval	Negative ratio	Precision	Recall	F1	Precision	Recall	F1
GLiNER	gold		balanced	73.63	**68.10**	**70.76**	74.79	**68.20**	**70.59**
GLiNER	gold		original	**88.84**	23.10	36.66	**88.81**	15.23	24.24
GLiNER	gold	spr	balanced	75.07	64.20	69.21	75.74	62.69	67.13
GLiNER	gold	spr	original	87.13	29.56	44.14	86.50	17.59	25.85
GLiNER	bronze		balanced	68.88	44.22	53.86	69.95	51.49	55.77
GLiNER	bronze		original	77.90	15.90	26.41	82.65	27.92	35.32
GLiNER	bronze	spr	balanced	72.33	42.78	53.76	69.21	47.17	53.24
GLiNER	bronze	spr	original	77.78	15.20	25.42	82.14	28.68	35.81
GLiNER	silver		balanced	61.09	57.10	59.03	67.81	58.72	60.23
GLiNER	silver		original	82.35	14.68	24.92	80.52	29.24	35.70
GLiNER	silver	spr	balanced	64.94	54.71	59.39	69.94	61.44	64.99
GLiNER	silver	spr	original	80.03	13.20	22.65	67.00	24.95	31.75

Table 3.2: GLiNER summary results on different configurations.

3.2.1 Models

GLiNER is the model capable of reaching the best performance on the gold labels. On the other hand, BERT turns out to be the best model for synthetic labels showing better performance as the quality of labels decreases. These results could suggest that even though GLiNER would be the best model for this dataset, it needs high-quality data to effectively reach the best results while BERT is less sensible making it preferable as a distilled model. Additionally, BERT shows a better recall against a better precision of GLiNER

3.2.2 Labels quality

Data quality is crucial for training effective models, low-quality labels lead to weaker models. As expected, models trained on gold labels perform the best, followed by those trained on silver, with bronze labels yielding the lowest-performing models. However, while the differences are noticeable, they are not as large as one might anticipate based on the label quality.

Between silver and bronze labels, there is about a nine-point difference in F1 score. Yet, the gap between the distilled models trained on these labels is only around four points. Similarly, while the difference between silver and gold labels is around forty-four points, the corresponding model difference is just six points.

This shows that the quality of distilled models does not scale proportionally with the quality of the labels. A possible explanation for this could be the models' ability to achieve decent performance quickly, even with a relatively lo small dataset. After this initial steep learning curve, the performance plateaus. At this point, further improvements likely focused on rare or complex cases that require increasingly higher-quality data. These difficult examples demand more refined data to be learned effectively.

3.2.3 Data balancing

As mentioned earlier, GLiNER's training benefits from a balanced ratio of positive and negative passages, experiments using the original ratio led to poor performance. In contrast, BERT demonstrated the opposite trend, performing better with the original ratio, highlighting that this issue was specific to GLiNER.

Interestingly, BERT achieved the best recall across all experimental setups with the balanced configuration. Fewer negative examples make the model less strict in identifying entities, resulting in a trade-off, higher recall but lower precision. This finding is particularly valuable in scenarios where high recall is prioritized, such as when model predictions are reviewed by humans. In such cases, having higher recall is often preferable to higher precision.

3.2.4 Passage retrieval

Retrieval is introduced only during the inference step, removing passages with entities from the training data would negatively impact model performance. Additionally, this would not introduce anything new, as balancing the number of negatives already filters out empty passages.

Surprisingly, for BERT, retrieval improves performance when the model is trained with a balanced number of positive and negative passages. Since this version of BERT is recall-oriented, retrieval helps by reducing the number of passages considered, thereby boosting precision.

For all the setups of GLiNER, the results remain nearly the same, with at most a slight drop in performance around a couple of points for the retrieval version. In general, unlike the corresponding teacher models, retrieval doesn't enhance the overall quality of the student models. However, it allows for faster inference with limited impact on metrics.

3.3 Teacher-student comparison

Once the distillation process is complete, a set of student models can be compared to their teacher. Table 3.3 presents the results of the two teacher models on FOLD_3, which is the same fold used for evaluating the students. Notably, in several student setups, the performance surpasses the teachers, with the best configurations achieving a micro F1 score around six points higher. However, the macro F1 scores remain similar between students and teachers.

This indicates that while both students and teachers maintain the same balance across classes, the student models produce better annotations for a larger number of samples. Similar macro F1 scores are expected since the quality of synthetic predictions in each class influences the training, if a class lacks strong annotations, the student's performance is limited accordingly. Notably, the micro F1 score is better for the students, indicating that they have become superior annotators compared to their teachers. The distillation process has preserved the quality ratio across classes while improving the overall number of accurate predictions.

					Micro			Macro		
Model	Examples type	Retrieval	Masked	Llm calls	Precision	Recall	F1	Precision	Recall	F1
Llama3.1 70B	dfsl	spr		3113	**52.53**	55.76	**54.09**	**61.26**	**64.05**	**62.51**
Mixtral 8x7B	dfsl	spr		3113	42.49	**56.20**	48.39	50.94	63.25	56.17

Table 3.3: Teacher results for the test set (FOLD_3).

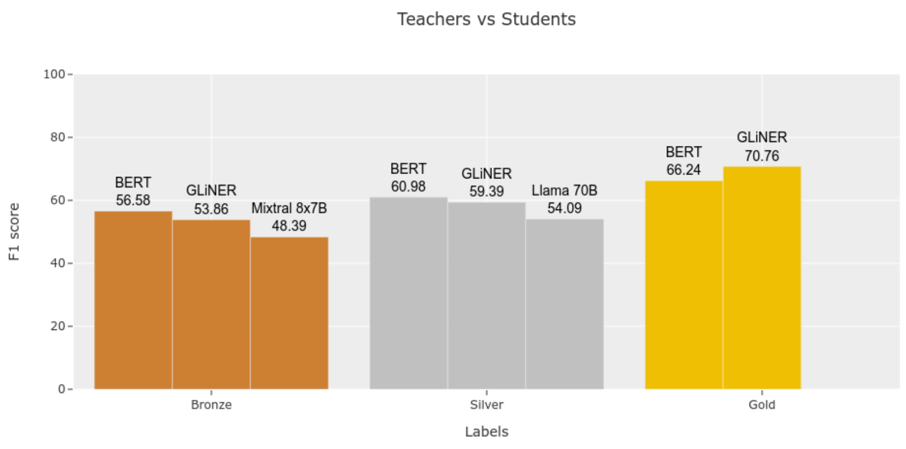

Figure 3.3: Comparison between students and teachers for the test set. The comparison is made by selecting the best configuration for each training data. For each category, the last column represents the corresponding teacher.

3.4 Conclusion

The presented work shows a successful distillation of larger generic models into smaller task-specific ones. Large language models have been proven to be adapted to noisy label generators, that with the correct engineering, can be refined to produce valuable synthetic annotations. To enhance the performance of these models a specific generation pipeline has been built, introducing two major components the retriever and the few-shot generator. The retriever not only allows a speed-up to the generation process by providing only the best candidates' chunks but also improves precision by filtering out empty passages that the model could misinterpret. The few-shot generator provides the model with valuable demonstrations that not only improve general performance but also help the model to have a better balance between different classes. How these examples are created is also important, smaller models prefer handmade examples engineered to their needs while Llama 70B benefits from dynamic generations of the examples.

The distillation process involved fine-tuning two different models GLiNER and BERT, with the last one that works slightly better on the synthetic data.

Both of them outperformed their teachers in all the setups showing how the training process effectively obtained valuable knowledge from the noisy labels. Even though the pipeline is constructed on a specific study case it can be easily adapted by changing the single components to better suit other scenarios. Summing up the most important findings:

- **Retrieval to reduce inference**: initially used to improve the LLM performance, it proved valid for the BERT styles model. It drastically reduces the amount of inference needed and improves the LLM performance while slightly reducing the ones of the student's models.
- **Retrieval by examples**: retrieval by similar examples (SPR) over a query-based method avoids any type of query tuning leading to a full data-driven approach, achieving better results.
- **Specialized prompts**: shorter, entity-focused prompts enable the model to concentrate on specific tags. Including a dummy class also provides an option for the model if it is uncertain about its predictions.
- **The distillation reduces noise**: even though the distilled models are not at the level of the gold ones, they turn out to be better of their teachers, showing how the training process has absorbed part of the noise.

Coming back to the research questions that motivated this work, the results demonstrate that LLMs can be effectively adapted as label generators for specific technical domains, such as business transactions. Techniques like passage retrieval and few-shot generation have significantly improved the speed and quality of the annotation process, enabling more efficient data labeling. Distilling smaller models using synthetic data has proven both feasible and effective, with the student models achieving superior performance compared to their teacher. The findings highlight that data quality remains a critical factor in building high-performing models. While higher-quality annotations generally lead to better models, the relationship is not strictly linear; substantial improvements in annotation quality do not always translate to proportional gains in the performance of distilled models. This underscores the complex interplay between data quality and model efficacy.

3.5 Future works

Here are some possible expansion and further experiments that could be conducted:

- **Multi-model distillation**: since the retrieval allows a complete separation between tag families it could be possible to have three different students, one for each family. In this way, each model could focus on just one family of entities. The retrieval would act as a router to the right model.
- **Iterate distillation**: the distilled model can be used to generate better annotations that could be used to repeat the distillation problem aiming for better performance.
- **Task specific chunker**: fine-tuning the chunker on the dataset could improve the performance making it aware of the specific format of the file.
- **Entity matching**: currently the entity matching in the text is done by a simple text search. A better method can involve the generation of the position in the text by the LLM.
- **Mixed gold-silver training set**: by integrating a selection of gold (human-annotated) data with the synthetically generated annotations, particularly for the most challenging cases. This strategy keeps human annotation efforts minimal by focusing only on high-impact cases while benefiting from the scale of synthetic annotations.

Bibliography

[1] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, É. Goffinet, D. Hesslow, J. Launay, Q. Malartic, D. Mazzotta, B. Noune, B. Pannier, and G. Penedo. The falcon series of open language models, 2023. arXiv: 2311.16867 [cs.CL].

[2] Appendix C: named entity task definition (v2.1). In *Sixth Message Understanding Conference (MUC-6): Proceedings of a Conference Held in Columbia, Maryland, November 6-8, 1995*, 1995. URL: https://aclanthology.org/M95-1024.

[3] D. Ashok and Z. C. Lipton. Promptner: prompting for named entity recognition, 2023. arXiv: 2305.15444 [cs.CL]. URL: https://arxiv.org/abs/2305.15444.

[4] D. Dannenhauer, Z. Dannenhauer, D. Christou, and K. Hatalis. A case-based reasoning approach to dynamic few-shot prompting for code generation. In *ICML 2024 Workshop on LLMs and Cognition*, 2024. URL: https://openreview.net/forum?id=Kt9bM32oDY.

[5] W. De Mulder, S. Bethard, and M.-F. Moens. A survey on the application of recurrent neural networks to statistical language modeling. *Comput. Speech Lang.*, 30(1):61–98, March 2015. ISSN: 0885-2308. DOI: 10.1016/j.csl.2014.09.005. URL: https://doi.org/10.1016/j.csl.2014.09.005.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: pre-training of deep bidirectional transformers for language understanding, 2019.

arXiv: 1810 . 04805 [cs.CL]. URL: https : / / arxiv . org / abs / 1810.04805.

[7] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, T. Liu, B. Chang, X. Sun, L. Li, and Z. Sui. A survey on in-context learning, 2024. arXiv: 2301.00234 [cs.CL]. URL: https://arxiv.org/abs/2301.00234.

[8] P. He, J. Gao, and W. Chen. Debertav3: improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *ArXiv*, abs/2111.09543, 2021. URL: https://api.semanticscholar.org/CorpusID:244346093.

[9] Y. Heng, C. Deng, Y. Li, Y. Yu, Y. Li, R. Zhang, and C. Zhang. Proggen: generating named entity recognition datasets step-by-step with self-reflexive large language models, 2024. arXiv: 2403.11103 [cs.CL]. URL: https://arxiv.org/abs/2403.11103.

[10] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015. arXiv: 1503.02531 [stat.ML]. URL: https://arxiv.org/abs/1503.02531.

[11] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023. arXiv: 2310 . 06825 [cs.CL].

[12] Z. Jie and W. Lu. Leveraging training data in few-shot prompting for numerical reasoning, 2023. arXiv: 2305.18170 [cs.CL]. URL: https://arxiv.org/abs/2305.18170.

[13] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. Spanbert: improving pre-training by representing and predicting spans, 2020. arXiv: 1907 . 10529 [cs.CL]. URL: https : / / arxiv . org / abs/1907.10529.

[14] J. Kazama, T. Makino, Y. Ohta, and J. Tsujii. Tuning support vector machines for biomedical named entity recognition. In pages 1–8, January 2002. DOI: 10.3115/1118149.1118150.

[15] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners, 2023. arXiv: 2205.11916 [cs.CL]. URL: https://arxiv.org/abs/2205.11916.

[16] B. Li, G. Fang, Y. Yang, Q. Wang, W. Ye, W. Zhao, and S. Zhang. Evaluating chatgpt's information extraction capabilities: an assessment of performance, explainability, calibration, and faithfulness, 2023. arXiv: 2304.11633 [cs.CL]. URL: https://arxiv.org/abs/2304.11633.

[17] J. Mayfield, P. McNamee, and C. Piatko. Named entity recognition using hundreds of thousands of features. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 184–187, 2003. URL: https://aclanthology.org/W03-0429.

[18] Y. Meng, M. Michalski, J. Huang, Y. Zhang, T. Abdelzaher, and J. Han. Tuning language models as training data generators for augmentation-enhanced few-shot learning, 2023. arXiv: 2211.03044 [cs.CL]. URL: https://arxiv.org/abs/2211.03044.

[19] A. Mikheev, M. Moens, and C. Grover. Named entity recognition without gazetteers. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL '99, pages 1–8, Bergen, Norway. Association for Computational Linguistics, 1999. DOI: 10.3115/977035.977037. URL: https://doi.org/10.3115/977035.977037.

[20] G. Popovski, S. Kochev, B. Korousic-Seljak, and T. Eftimov. Foodie: a rule-based named-entity recognition method for food information extraction. In *International Conference on Pattern Recognition Applications and Methods*, 2019. URL: `https://api.semanticscholar.org/CorpusID:88488573`.

[21] O. Sainz, I. García-Ferrero, R. Agerri, O. L. de Lacalle, G. Rigau, and E. Agirre. GoLLIE: annotation guidelines improve zero-shot information-extraction. In *The Twelfth International Conference on Learning Representations*, 2024. URL: `https://openreview.net/forum?id=Y3wpuxd7u9`.

[22] S. Schweter and A. Akbik. Flert: document-level features for named entity recognition, 2021. arXiv: `2011.06993 [cs.CL]`. URL: `https://arxiv.org/abs/2011.06993`.

[23] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: open and efficient foundation language models, 2023. arXiv: `2302.13971 [cs.CL]`.

[24] S. Wang, X. Sun, X. Li, R. Ouyang, F. Wu, T. Zhang, J. Li, and G. Wang. Gpt-ner: named entity recognition via large language models, 2023. arXiv: `2304.10428 [cs.CL]`. URL: `https://arxiv.org/abs/2304.10428`.

[25] X. Wei, X. Cui, N. Cheng, X. Wang, X. Zhang, S. Huang, P. Xie, J. Xu, Y. Chen, M. Zhang, Y. Jiang, and W. Han. Chatie: zero-shot information extraction via chatting with chatgpt, 2024. arXiv: `2302.10205 [cs.CL]`. URL: `https://arxiv.org/abs/2302.10205`.

[26] X. Xu, M. Li, C. Tao, T. Shen, R. Cheng, J. Li, C. Xu, D. Tao, and T. Zhou. A survey on knowledge distillation of large language models, 2024. arXiv: `2402.13116 [cs.CL]`. URL: `https://arxiv.org/abs/2402.13116`.

[27] U. Zaratiana, N. Tomeh, P. Holat, and T. Charnois. Gliner: generalist model for named entity recognition using bidirectional transformer, 2023. arXiv: `2311.08526 [cs.CL]`.

[28] W. Zhou, S. Zhang, Y. Gu, M. Chen, and H. Poon. Universalner: targeted distillation from large language models for open named entity recognition, 2024. arXiv: `2308.03279 [cs.CL]`. URL: `https://arxiv.org/abs/2308.03279`.

[29] A. Zugarini, A. Zamai, M. Ernandes, and L. Rigutini. Buster: a "business transaction entity recognition" dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 605–611. Association for Computational Linguistics, 2023. DOI: `10.18653/v1/2023.emnlp-industry.57`. URL: `http://dx.doi.org/10.18653/v1/2023.emnlp-industry.57`.

Appendix A

Synthetic generations results

					Buying company			Selling company			Acquired company		
Model	Examples type	Retrieval	Masked	Llm calls	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Mixtral 8x7B	dfsl	spr		2286	37.11	54.61	44.19	38.64	48.32	42.94	45.84	50.85	48.21
Mixtral 8x7B	dfsl	spr	✓	2521	33.77	51.56	40.81	37.52	48.77	42.41	43.54	47.79	45.57
Mixtral 8x7B	few-shot	spr		2286	**49.68**	51.00	**50.33**	40.04	**48.99**	**44.06**	**52.77**	49.08	**50.86**
Mixtral 8x7B	few-shot	spr	✓	2521	45.80	48.17	46.95	**40.41**	48.10	43.92	48.69	46.57	47.61
Mixtral 8x7B	zero-shot			18096	22.71	**63.33**	33.43	21.31	25.50	23.22	26.84	**67.14**	38.35
Mixtral 8x7B	zero-shot	spr		2286	44.74	49.89	47.18	30.40	22.37	25.77	44.64	52.27	48.16
Mixtral 8x7B	zero-shot	spr	✓	2521	40.43	47.50	43.68	28.98	20.36	23.92	42.44	50.71	46.21
Mixtral 8x7B	zero-shot	tag-wise		3030	35.85	44.78	39.82	25.71	20.36	22.72	38.21	49.15	42.99

Table A.1: Mixtral 8x7B results for different configurations on the *Parties* classes.

					Generic consulting company			Legal consulting company			Annual revenues		
Model	Examples type	Retrieval	Masked	Llm calls	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Mixtral 8x7B	dfsl	spr		857	49.73	67.79	57.37	**79.19**	89.39	**83.99**	43.30	65.54	52.15
Mixtral 8x7B	dfsl	spr	✓	904	49.71	64.04	55.97	78.01	83.33	80.59	49.53	**70.95**	58.33
Mixtral 8x7B	few-shot	spr		857	69.11	67.04	**68.06**	69.64	88.64	78.00	68.70	60.81	**64.52**
Mixtral 8x7B	few-shot	spr	✓	904	66.41	63.67	65.01	74.32	83.33	78.57	**71.19**	56.76	63.16
Mixtral 8x7B	zero-shot			36192	28.64	**71.54**	40.90	67.58	**93.18**	78.34	21.83	62.84	32.40
Mixtral 8x7B	zero-shot	spr		857	69.23	64.04	66.54	74.84	87.88	80.84	41.08	51.35	45.65
Mixtral 8x7B	zero-shot	spr	✓	904	**70.76**	62.55	66.40	76.39	83.33	79.71	39.69	52.03	45.03
Mixtral 8x7B	zero-shot	tag-wise		3034	45.69	65.54	53.85	72.66	76.52	74.54	34.98	47.97	40.46

Table A.2: Mixtral 8x7B results for different configurations on the *Advisors* and *Generic Info* classes.

					Buying company			Selling company			Acquired company		
Model	Examples type	Retrieval	Masked	Llm calls	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Llama3.1	dfsl	spr		2286	43.39	**56.33**	49.02	23.47	52.35	32.41	43.66	51.39	47.21
Llama3.1	dfsl	spr	✓	2521	41.06	50.78	45.40	23.91	51.45	32.65	41.38	48.54	44.67
Llama3.1	few-shot	spr		2286	**56.58**	48.72	52.36	31.81	49.66	38.78	55.08	48.61	51.64
Llama3.1	few-shot	spr	✓	2521	54.55	47.28	50.65	31.53	49.66	38.58	51.51	47.59	49.47
Llama3.1	zero-shot			18096	39.29	53.00	45.13	23.78	40.27	29.90	35.91	**61.78**	45.42
Llama3.1	zero-shot	spr		2286	51.56	44.89	48.00	28.47	34.45	31.17	46.94	51.60	49.16
Llama3.1	zero-shot	spr	✓	2521	48.49	42.78	45.45	29.76	36.02	32.59	44.16	49.29	46.58
Llama3.1	zero-shot	tag-wise		3030	49.13	39.17	43.59	28.72	31.54	30.06	45.91	45.69	45.80
Llama3.1 70B	dfsl	spr		2286	55.80	54.22	**55.00**	42.99	53.47	47.66	**58.46**	50.92	**54.43**
Llama3.1 70B	few-shot	spr		2286	54.81	50.00	52.30	41.41	**53.91**	46.84	57.13	48.95	52.72
Llama3.1 70B	zero-shot	spr		2286	53.68	49.89	51.71	**46.11**	51.68	**48.73**	51.45	50.71	51.08

Table A.3: Llama 3.1 results for different configurations on the *Parties* classes.

					Generic consulting company			Legal consulting company			Annual revenues		
Model	Examples type	Retrieval	Masked	Llm calls	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Llama3.1	dfsl	spr		857	62.24	**66.67**	64.38	72.84	89.39	80.27	38.32	70.95	49.76
Llama3.1	dfsl	spr	✓	904	62.30	58.80	60.50	69.03	81.06	74.56	37.72	**73.65**	49.89
Llama3.1	few-shot	spr		857	**76.44**	49.81	60.32	61.78	89.39	73.07	58.43	70.27	63.80
Llama3.1	few-shot	spr	✓	904	74.57	48.31	58.64	64.67	81.82	72.24	57.75	72.97	64.48
Llama3.1	zero-shot			36192	38.89	52.43	44.66	39.30	**93.18**	55.28	1.63	6.76	2.63
Llama3.1	zero-shot	spr		857	72.59	53.56	61.64	62.77	89.39	73.75	11.59	5.41	7.37
Llama3.1	zero-shot	spr	✓	904	75.84	50.56	60.67	63.53	81.82	71.52	8.75	4.73	6.14
Llama3.1	zero-shot	tag-wise		3034	47.10	48.69	47.88	47.47	78.03	59.03	18.44	35.14	24.19
Llama3.1 70B	dfsl	spr		857	69.17	65.54	**67.31**	**80.14**	88.64	**84.17**	74.82	70.27	**72.47**
Llama3.1 70B	few-shot	spr		857	67.70	65.17	66.41	76.47	88.64	82.11	**78.86**	65.54	71.59
Llama3.1 70B	zero-shot	spr		857	65.93	**66.67**	66.29	75.64	89.39	81.94	71.33	68.92	70.10

Table A.4: Llama 3.1 results for different configurations on the *Advisors* and *Generic Info* classes.

Appendix B

Distillation results

				Buying company			Selling company			Acquired company		
Model	Training data	Retrieval	Negative ratio	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BERT	gold		balanced	42.43	**81.79**	55.87	48.35	**70.83**	57.47	37.56	**74.23**	49.88
BERT	gold		original	67.27	75.68	**71.23**	56.73	63.82	60.06	57.44	65.68	**61.28**
BERT	gold	spr	balanced	57.52	70.51	63.36	56.55	65.35	**60.63**	49.94	59.87	54.45
BERT	gold	spr	original	**70.40**	68.35	69.36	**59.53**	60.96	60.24	**59.77**	56.77	58.23
BERT	bronze		balanced	20.11	78.53	32.02	29.92	62.72	40.51	29.19	73.27	41.75
BERT	bronze		original	43.64	70.51	53.91	49.46	60.09	54.26	48.66	61.63	54.39
BERT	bronze	spr	balanced	38.83	65.33	48.71	45.83	57.89	51.16	42.97	58.98	49.72
BERT	bronze	spr	original	52.30	64.75	57.86	53.25	57.46	55.27	54.04	55.67	54.84
BERT	silver		balanced	36.68	81.38	50.57	30.52	70.61	42.62	34.62	73.56	47.08
BERT	silver		original	58.69	67.95	62.98	52.83	65.57	58.51	55.75	57.14	56.44
BERT	silver	spr	balanced	50.19	70.16	58.52	40.82	65.35	50.25	48.14	59.13	53.07
BERT	silver	spr	original	61.98	62.13	62.06	52.65	63.16	57.43	57.37	53.61	55.42

Table B.1: BERT results for different configurations on *Parties* classes.

				Generic consulting company			Legal consulting company			Annual revenues		
Model	Training data	Retrieval	Negative ratio	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BERT	gold		balanced	43.71	**81.23**	56.84	67.98	79.61	73.33	35.65	88.89	50.89
BERT	gold		original	61.40	77.39	**68.47**	72.35	**80.92**	**76.40**	49.32	75.00	59.50
BERT	gold	spr	balanced	58.75	75.86	66.22	71.34	76.97	74.05	38.68	77.08	51.51
BERT	gold	spr	original	63.79	73.56	68.33	**73.29**	77.63	75.40	53.63	66.67	59.44
BERT	bronze		balanced	33.55	79.31	47.15	61.11	79.61	69.14	28.86	89.58	43.65
BERT	bronze		original	50.00	70.88	58.64	50.26	63.82	56.23	37.09	77.78	50.22
BERT	bronze	spr	balanced	49.74	73.56	59.35	67.05	76.32	71.38	37.25	77.08	50.23
BERT	bronze	spr	original	58.67	67.43	62.75	52.84	61.18	56.71	39.38	70.83	50.62
BERT	silver		balanced	39.62	78.93	52.75	60.29	**80.92**	69.10	34.03	**90.28**	49.43
BERT	silver		original	**67.42**	68.20	67.81	64.74	73.68	68.92	59.32	72.92	65.42
BERT	silver	spr	balanced	52.62	73.18	61.22	67.05	77.63	71.95	41.30	79.17	54.29
BERT	silver	spr	original	63.77	67.43	65.55	66.87	73.03	69.81	**62.50**	69.44	**65.79**

Table B.2: BERT results for different configurations on *Advisors* and *Generic Info* classes.

				Buying company			Selling company			Acquired company		
Model	Training data	Retrieval	Negative ratio	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
GLiNER	gold		balanced	74.38	**74.90**	**74.64**	75.18	**67.11**	**70.92**	70.04	**58.37**	**63.68**
GLiNER	gold		original	**91.85**	30.80	46.13	**90.51**	27.19	41.82	**82.97**	19.31	31.33
GLiNER	gold	spr	balanced	78.34	68.74	73.23	74.69	66.01	70.08	68.82	58.00	62.95
GLiNER	gold	spr	original	89.48	43.00	58.08	88.17	32.68	47.68	81.38	22.47	35.21
GLiNER	bronze		balanced	65.09	46.48	54.24	63.40	52.41	57.38	75.36	30.54	43.47
GLiNER	bronze		original	71.81	9.47	16.74	75.00	23.68	36.00	76.00	11.16	19.46
GLiNER	bronze	spr	balanced	69.21	46.37	55.53	66.67	48.25	55.98	75.27	31.06	43.97
GLiNER	bronze	spr	original	70.51	8.89	15.79	74.03	25.00	37.38	76.06	7.93	14.36
GLiNER	silver		balanced	55.37	64.44	59.56	60.14	55.26	57.60	66.78	44.86	53.67
GLiNER	silver		original	82.24	12.38	21.52	74.83	24.78	37.23	71.43	1.10	2.17
GLiNER	silver	spr	balanced	60.54	57.06	58.75	60.14	54.61	57.24	67.77	44.93	54.04
GLiNER	silver	spr	original	76.17	12.26	21.12	72.12	16.45	26.79	78.79	1.91	3.73

Table B.3: GLiNER results for different configurations on *Parties* classes.

				Generic consulting company			Legal consulting company			Annual revenues		
Model	Training data	Retrieval	Negative ratio	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
GLiNER	gold		balanced	76.30	**78.93**	**77.59**	84.94	**92.76**	**88.68**	67.90	37.16	48.03
GLiNER	gold		original	87.50	8.05	14.74	**100.00**	0.66	1.31	80.00	5.41	10.13
GLiNER	gold	spr	balanced	80.00	73.56	76.65	85.26	87.50	86.36	67.35	22.30	33.50
GLiNER	gold	spr	original	**93.33**	5.36	10.14	**100.00**	0.66	1.31	66.67	1.35	2.65
GLiNER	bronze		balanced	73.74	78.54	76.07	80.59	90.13	85.09	61.54	10.81	18.39
GLiNER	bronze		original	90.09	38.31	53.76	83.01	83.55	83.28	**100.00**	1.35	2.67
GLiNER	bronze	spr	balanced	83.49	67.82	74.84	83.12	87.50	85.26	37.50	2.03	3.85
GLiNER	bronze	spr	original	88.89	45.98	60.61	83.33	82.24	82.78	**100.00**	2.03	3.97
GLiNER	silver		balanced	70.88	77.39	73.99	81.18	90.79	85.71	72.50	19.59	30.85
GLiNER	silver		original	90.65	48.28	63.00	83.97	86.18	85.06	80.00	2.70	5.23
GLiNER	silver	spr	balanced	77.02	73.18	75.05	83.12	87.50	85.26	71.03	**51.35**	**59.61**
GLiNER	silver	spr	original	89.84	44.06	59.13	85.07	75.00	79.72	0.00	0.00	0.00

Table B.4: GLiNER results for different configurations on *Advisors* and *Generic Info* classes.