Entity Extraction in Premodern German Administrative Documents: Accuracy and Cost in the Age of Generative AI

Prada Ziegler, Ismail
University of Bern, University of Basel

Preprint Version of August 2026.

Abstract

Named Entity Recognition in historical texts faces unique challenges, including non-standardized language, transcription noise, and limited annotated material. We evaluate two approaches on two tasks on the Historical Land Records of Basel, a dataset of 829 German-language documents from the 14th to 18th century. Task 1 involves classic NER, identifying persons, locations, and organizations, while Task 2 addresses nested entity annotations, additionally requiring the identification of grammatical structures such as heads, appositions, and attributes. We compare generative LLM-based tagging (GPT-4o) with domain-specific fine-tuned sequence taggers using Flair contextual character embeddings, examining the impact of variable training data. GPT outperforms Flair in few-shot scenarios, particularly for low-frequency tags, while Flair achieves superior performance with larger training sets and longer spans. We find that, no matter which system, a substantial increase in accuracy can be gained by increasing the amount of training data. We also analyze annotation cost and speed, showing that manually curated datasets can be generated efficiently and support well-performing models. Our findings demonstrate that LLMs provide strong initial results, but lighter traditional systems remain competitive with sufficient training data.

Introduction

Named Entity Recognition (NER) is a field of Natural Language Processing concerned with extracting references to entities - typically persons, places, and organizations, but sometimes extended to include works of art, brands, and other categories - from text. In historical editions, the annotation and linking of these entities was commonplace long before the advent of the internet. When entities are properly annotated and identified in a database, the resulting information can be used to build large-scale registries through which historians, sociologists, and other researchers, as well as the interested public can quickly find materials of interest. Understandably, an increasing number of historical projects are seeking to enrich their digitized datasets through automated methods.

Historical texts, however, present a number of challenges for building well-performing NER models. In addition to the difficulties posed by high spelling variation and noisy transcriptions, the persistent lack of available training data has long hindered research in historical NLP (Ehrmann et al., 2023). The extent of this data scarcity is evident in the survey on premodern NER by Ehrmann et al. (2023): among the datasets listed, only one German-language resource refers to documents produced before the 18th century. Even within that dataset, *Deutsches ROman Corpus*, only two out of ninety novel fragments date from before 1700, and nine from before 1800 (Krug et al., 2017). Overall, just one dataset in the survey includes documents from medieval and early modern periods: a multilingual collection of annotated medieval charters (Aguilar, 2022).

Unsurprisingly, researchers have sought solutions that require little or no training data. One such method was to rely on rule-based systems in combination with gazetteers - for example, lists of common person names (Grover et al., 2008). For historical texts that are linguistically close to their modern counterparts, a valid strategy is to apply a model trained on modern language and fine-tune it using a small amount of historical data (Blouin et al., 2022).

More recently, with the rise of generative large language models (LLMs) such as GPT, zero- and few-shot approaches have become increasingly prevalent (González-Gallardo et al., 2023; Hiltmann et al., 2025).[1] Although most studies indicate that traditional methods, such as recurrent neural networks and BERT architectures, continue to outperform generative LLMs on the majority of information extraction tasks (González-Gallardo et al., 2023; Wang et al., 2025; Keraghel et al., 2024), the ease of use and minimal preparation required to obtain acceptable results can tempt scholars in the (digital) humanities to rely exclusively on generative LLMs. Consequently, results produced by these models are often accepted as sufficiently accurate, and in some cases, no systematic evaluation of the automated output is conducted.

This paper examines what we risk losing when depending solely on zero- and few-shot systems. We do so by presenting a case study that explores performance differences depending on the amount of available training material. Specifically, we evaluate two Named Entity Recognition strategies, one based on GPT-4o (OpenAI, 2024) and the other on the Flair framework (Akbik et al., 2019), using a historical dataset of German-language administrative documents. We then reflect on these methods with respect to both accuracy and financial cost.

We evaluate the accuracy of the models in two distinct settings: First, we consider a standard NER task in which only proper names must be identified. Second, we address a nested entity recognition task employing an expanded tagset designed for more complex historical inquiries. Because GPT-4o possesses pre-trained knowledge about named entity recognition, our goal here is to assess how effectively the model can follow specific annotation guidelines in such complex contexts. **The case study further illustrates how the amount of available training data influences the performance of both approaches.**

Dataset

The foundational dataset for this case study is the *Historical Land Records of Basel Ground Truth* corpus (HLR-GT) (Prada Ziegler et al., 2025). The corpus comprises 829 digitized and

[1] Zero shot approaches rely only on pre-trained knowledge of a model, while few-shot approaches include a few examples for the model in the prompt.

annotated archival index cards containing excerpts from administrative documents dating from the fourteenth to the eighteenth century. It provides an ideal scenario for Natural Language Processing on historical material, as it features all the characteristic challenges: noise introduced by handwritten text recognition (HTR), non-standardized language, scarcity of annotated material, and the absence of a suitable baseline model.

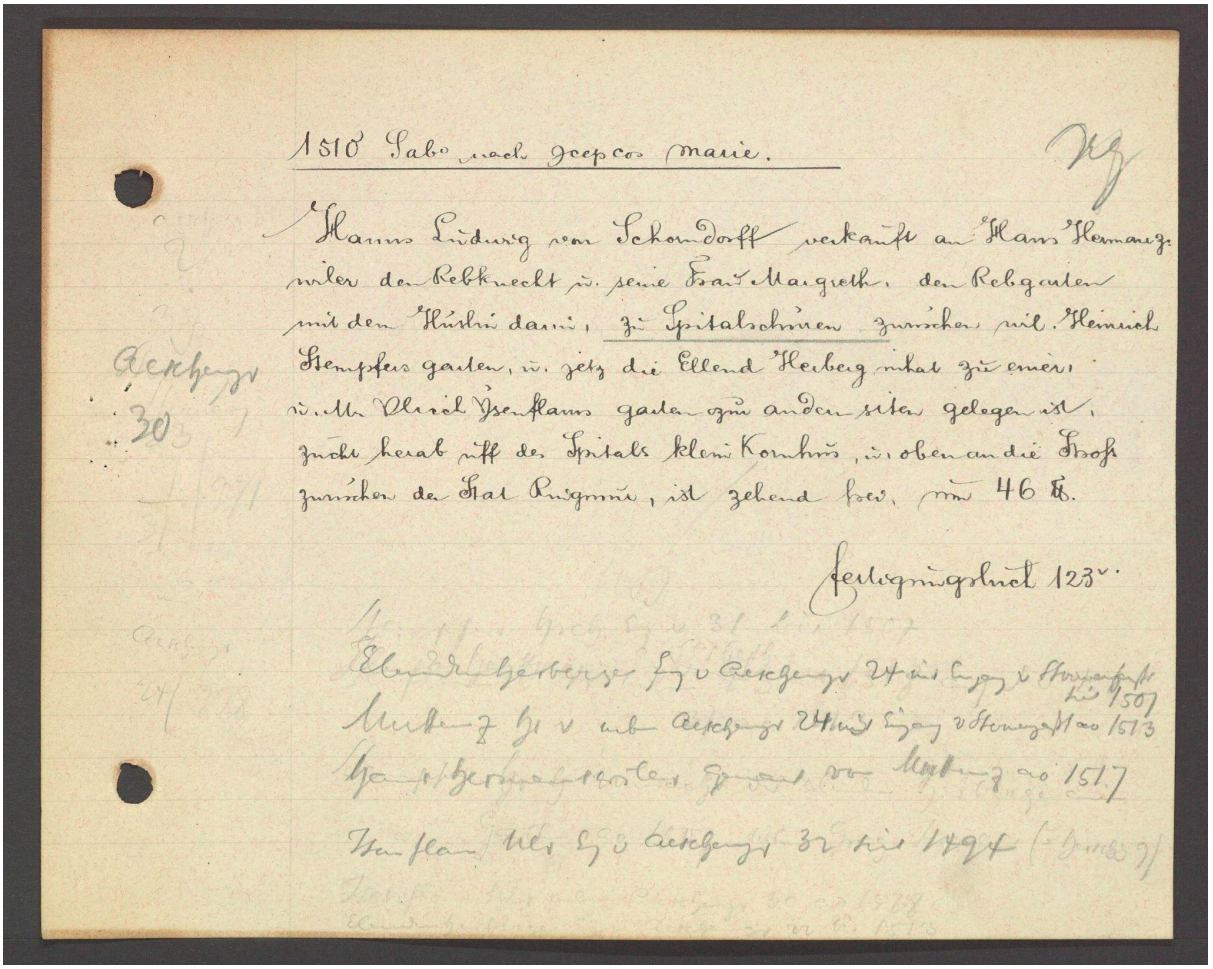

Figure 1: One archival index card on which the HLR-GT is based. Only the main text is part of the corpus; header, marginalia, and footnotes were removed. Image Source: State Archive of Basel-Stadt HGB 11/34, pp. 4.

From a scholarly perspective, the corpus offers a rich foundation of information well-suited for quantitative analysis. Each index card contains an excerpt or copy of an original document written in Early New High German. Most entries record a single transaction, trade, or list of dues payments, though less frequent document types, such as court cases or testaments, are also represented. While capitalization and punctuation are generally present, both are inconsistent; although this information can assist the models in learning and predicting annotations, it may also occasionally introduce noise. Documents from the same source (e.g., court books) often exhibit strong structural similarity. For instance, a property purchase frequently follows the formula: “<Actor A> sells <Actor B> <Property> for <Money>.” Systems that learn to recognize such formulaic structures tend to perform better. However, they are also at risk of overfitting and may fail to generalize to documents that deviate from these patterns. These observations were previously noted in Prada Ziegler (2024).

```
<span element="reference" class="per">Frawen
    <span element="head" class="nam">Margareth Weibert</span> ,
    <span element="appo" class="fam">weiland
        <span element="reference" class="per">H .
            <span element="head" class="nam">Hanns Heinrich Wentzen</span>
            <span element="appo" class="occ">
                <span element="head" class="occ">Stattgerichts Besitzers</span> in
                <span element="reference" class="gpe-org">
                    <span element="head" class="nam">Mehrerem Basell</span>
                </span>
            </span>
        </span> , hinderlassener
        <span element="head" class="fam">Wittib</span>
    </span>
</span>
```

Figure 2: A BeNASch-annotated person reference in XML-encoding.

The HLR is annotated according to a variant of the BeNASch standard[2], which employs a deeply nested entity annotation scheme. An example of this structure is shown in figure 2. In the excerpt, three separate entities are mentioned (*reference*): *Margareth Weibert*, *Hanns Heinrich Wentzen*, and *Greater Basel* (the part of the city west of the Rhine). Through nesting, additional layers of information are encoded. Heads, appositions, and attributes (the latter not present in this example) provide further details about each entity, including names (*nam*), occupations (*occ*), and family relationships (*fam*). In subsequent processing, this nested information can be leveraged to extract relationships. For instance, *Margareth* is related to *Hanns* as *Wittib* (widow), and *Hanns* is employed by *Greater Basel* as *Stattgerichts Besitzer* (member of the court).
For Task 1, representing a conventional named entity recognition setup, all annotations were removed except head elements with the *nam* class. For Task 2, the full tagset was retained.
The dataset comprises a total of 54,925 tokens across 829 documents. Document length varies considerably, with an average of 66.26 tokens and a standard deviation of 34.33 tokens. The shortest document contains 7 tokens, while the longest extends to 237 tokens.

Approaches

In this study, we compare two approaches to named entity recognition on historical texts: LLM-based tagging and specifically trained supervised taggers. For the latter, we developed and applied Flair models in three distinct configurations. In parallel, we implemented a commercially available LLM using a strategy that has proven successful in prior work.
Flair leverages contextual character embeddings (Akbik et al., 2018), which have demonstrated robust performance for premodern German (Schweter et al., 2019; Hodel et al., 2023).

Flair: Base

This approach employs the pre-trained Flair model *ner-german*[3] (Akbik et al., 2018). This model was trained on the CoNLL-03 revised dataset (Tjong Kim Sang and De Meulder,

[2] https://dhbern.github.io/BeNASch/ (last accessed 08.10.2025).
[3] https://huggingface.co/flair/ner-german (last accessed: 09.10.2025)

2003), consisting primarily of modern, annotated newspaper articles. Using this model out of the box is the quickest approach, but its lack of knowledge about the target domain is expected to limit performance. It is important to note that the annotation guidelines for CoNLL-03 and the reduced HLR-GT dataset are not fully aligned. Hence some annotations that would be considered correct under CoNLL-03 standards may not be recognized as correct in our evaluation.

Flair: Fine-tuned

In this approach, the pre-trained *ner-german* model is fine-tuned on the HLR-GT dataset. Fine-tuning is a suitable strategy when only limited raw data is available and training a language model from scratch is impractical. The objective is to enrich the base model with domain-specific knowledge as well as familiarity with the desired annotation guidelines. Fine-tuning was performed using the available training data for each scenario, with a learning rate of 0.00005 over 50 epochs. The number of epochs was determined through evaluation on the validation set during initial testing.

Flair: Customized

In this approach, we fine-tuned contextual character embeddings, the *de-forward* and *de-backward* models provided by Flair, using all available text from the historical land records[4] (including documents which are not part of the annotated dataset, totaling approximately 9 million tokens). Using these fine-tuned embeddings, we then trained custom SequenceTagger models in Flair with default parameters.[5]

Other “Traditional” Embeddings

In preliminary experiments, we investigated the performance of BERT (Devlin et al., 2019), fastText (Bojanowski et al., 2017), and word2vec embeddings (Mikolov et al., 2013) in combination with the Flair sequence tagger, both with and without stacking alongside contextual character embeddings. When used without stacking, performance was significantly worse, while stacking did not yield any significant improvement. Therefore, these embeddings were not included in the experiments reported in this case study.

Generative AI (GPT-4o)

Several strategies have been proposed for prompting generative LLMs to perform entity extraction. In this study, we follow the recent approach by Hiltmann et al. (2025) and conduct our experiments with the framework NER4all.[6] All experiments were executed using *gpt-4o-2024-08-06.* In the following sections, we detail the changes we made to the default NER4all configuration.

---

[4] Not published at this date.
[5] Sequence Tagger models in Flair are simple Bi-LSTM+CRF architectures.
[6] At the time of writing, the code of NER4all is not public. Hiltmann and his team provided the code to us directly. We will publicly share our version of the code with all adaptations such as similarity-based example retrieval and nested annotation after NER4all has been published.

Detailed Context Prompt

Following the recommendations of Hiltmann et al. (2025), we augmented the prompt with detailed contextual information. We selected a German full prompt, which demonstrated slightly better performance on the validation set. Further task-specific fine-tuning and adaptation of the prompt are described in the relevant sections. All prompt fine-tuning was conducted using a validation set comprising 83 documents. Complete prompts are provided in the appendix.

Similarity-based Examples

As part of the prompt, the system is provided with a set of examples, which is particularly important for learning non-standard annotation guidelines. In Hiltmann et al. (2025), examples were manually selected to ensure coverage of all tags in the tagset.
When we simulate scenarios with limited training data in the following experiments, if the available training data is smaller than the number of shots, the examples are chosen randomly. Curated examples can perform better in such cases, as random selection may omit some entity classes or include only generic examples.
Given that the number of available examples often exceeds the prompt's context length, we implemented a strategy proposed by Wang et al. (2025), who found that selecting contextually appropriate examples rather than random ones can substantially improve model output. The strategy aims to find examples that are similar to the given input. We use the *DocumentPoolEmbeddings* provided by the Flair framework to combine the language models trained for the *flair-custom* model to generate embeddings for all samples in our training data. During prompt construction, the input sample is embedded using the same method, and cosine distances to all training samples are calculated. The examples with the smallest distances, i.e., the most similar samples, are then selected to be presented to the model.

Input	seinem Haus zur Büttenen genannt bey sant Urbans Brung nen gelegen
1	seinem Eckhaus hinder sant Elsbethen ob des Spittal Kornhaus zum Widerlin genannt gelegen
2	seinem hauss zum Fürst genant an der Gerbergassen , gelegen
3	seinem Haus gegen sant Elsbethen überhin gelegen

Table 1: An example of how the similarity-based retrieval works to get annotation examples of different house references. Tags are omitted in favor of readability; the prompt will include each example with and without tags.

Increased Shots

Hiltmann et al. (2025) report that a maximum of 32 examples ("shots") is typically optimal due to the context window limitations of the models. In our experiments, we evaluated different shot counts on the validation set and selected the configuration that yielded the best performance: 128 shots for Task 1 and 64 shots for Task 2. Although the performance differences between 32, 64, and 128 shots were generally small, they were statistically significant. In Task 2, for instance, reducing the number of shots from 64 to 32 resulted in an approximate 2 percentage point decrease in performance.

Accuracy

Task 1: Named Entity Extraction

Task Description

In this task, the models are required to identify conventional named entity spans, typically corresponding to proper nouns. The target categories are persons (PER), organizations (ORG), and locations (LOC). Geopolitical entities, which are tagged separately in the HLR-GT dataset, were mapped to these categories depending on context (e.g., “I live in Basel” ⇒ LOC, “I am a citizen of Basel” ⇒ ORG).

The HLR-GT contains 2,583 PER tags, 2,079 LOC tags, and 518 ORG tags.

We assess the effect of available training data, models were trained or prompted using subsets of 4, 8, 16, 32, 64, 128, 256, 512, and 662/745 samples.[7] Thresholds of 4 and 8 were omitted for Flair models, as performance at these levels was entirely unusable. Each experimental configuration was executed three times to improve reliability of the results.

Generative AI Setup

Validation set experiments indicated that a 128-shot configuration was optimal for this task. Based on a review of NER4all’s performance on the validation set, we introduced additional instructions to address recurring errors in the model output. Specifically, the guidelines were extended as follows: 1) Annotate only proper names. 2) Assign geopolitical entities to the appropriate category based on context. 3) For organizations and locations, annotate the grammatical head. 4) Do not annotate names of saints when they occur as part of other entity names.

Results

The results are plotted in figure 3 and listed as a table in appendix D. As expected, GPT-4o dominates the few-shot scenarios, outperforming the best-performing Flair models by 5-10 percentage points. Consistent with prior findings (Hiltmann et al., 2025), the non-fine-tuned Flair model performs substantially worse than all other approaches, likely due to its inability to handle the historical language and lack of knowledge regarding the annotation guidelines. Among the Flair models, the custom model trained from scratch achieved the highest performance, surpassing even the fine-tuned Flair model. The fine-tuned model’s relatively lower performance appears to result from the absence of a corpus-specific language model. Regarding GPT-4o, there appears to be a critical threshold in the number of examples provided: performance increases sharply from 16 samples (68.76% F1) to 32 samples (74.51% F1), with further significant gains observed as the number of samples doubles up to 256. Beyond this point, performance begins to plateau, with a slight upward trend observed at 662 samples. For Flair-Custom, results are highly variable when training data is limited.

[7] The maximum available training sample count varies slightly, because the sample count for Flair always includes the 10% of samples used for validation, while the sample count for GPT is exclusively using the training data at this point.

This may be attributable to the relatively small validation set, comprising approximately 10% of the total documents (e.g., only 3 documents in the validation set for the 32-sample scenario). A small validation set may cause training to terminate before an optimal model state is reached; however, initial testing indicated that increasing the validation set proportion generally worsened average performance. At around 128 samples, Flair-Custom matches GPT-4o performance, while at 256 samples, it outperforms all other models. Performance appears to plateau after 512 samples.

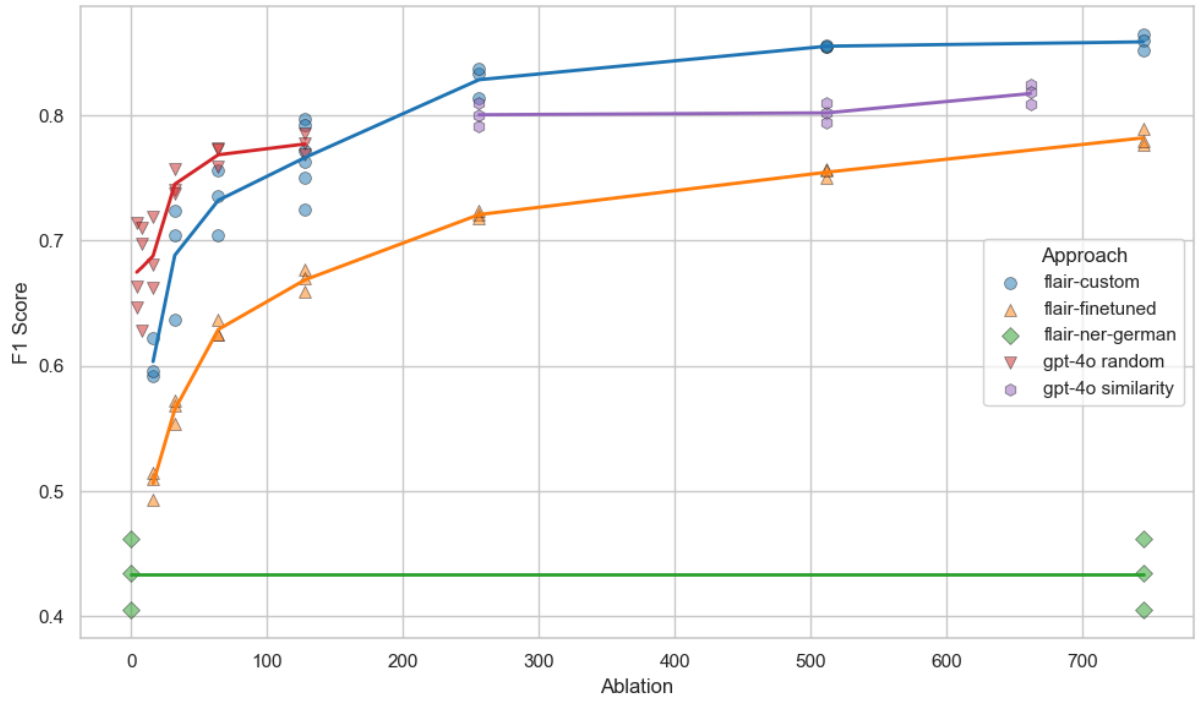

Figure 3: Performance comparison of chosen models on different training data settings.

Error Analysis

To investigate the observed performance plateaus, we analyzed predictions classified as incorrect during automated evaluation. We found that 66% of Flair's errors and 54% of GPT-4o's errors could be attributed to one of the following categories:

1) *HTR-induced noise:* Errors stemming from heavy transcription errors, where even human annotators could only roughly infer the correct annotation (Flair: 13%, GPT-4o: 8%).

2) *Ground truth inaccuracies:* Cases in which the model's prediction was actually correct, but the ground truth was erroneous (Flair: 19%, GPT-4o: 13%).

3) *Reasonable deviations:* Predictions that differed from the ground truth but were reasonable enough that they would not significantly affect downstream tasks, such as entity linking (Flair: 34%, GPT-4o: 33%).

Approximately 10% of errors in both models were simple ORG–LOC confusions. While potentially more impactful for downstream tasks such as event recognition, these errors are of limited relevance for tasks like creating a corpus index. Additionally, GPT-4o occasionally annotated non-named entity references (e.g., "Brugg" as a bridge), which accounted for 10% of its errors. Again, for most downstream tasks, this would not be particularly problematic behaviour. Overall, only 28% of GPT-4o's errors and 24% of Flair's errors could not be

explained by one of the categories above. Given the already strong performance of both models, this analysis suggests that further improvements would require substantially larger amounts of training data.

Task 2: Nested Extraction

Task Description

The HLR-GT dataset features a complex nested annotation scheme, in which entity references are further annotated to capture additional information about the entities they contain. Task 2 evaluates model performance on the full annotation set. Each tag consists of an element tag (e.g. *reference*) and, in some cases, an associated class tag (e.g. *person*). An overview of element tags is provided in table 2.

Tag	Description	Example
REF	Entity reference	**Johann the tailor**
HEAD	Head element in reference	**Johann** the tailor
APPO	Apposition in reference	Johann **the tailor**
ATTR	Attribute in reference	the house **in Basel**
DATE	Reference to a date	**1573** / **sunday after martini**
CURR	Monetary value or similar	**5 lb** / **2 satchels of corn**
EV	Event Trigger	Johann **sells** a house

Table 2: Overview of the element tags the systems must detect in task 2.

For references, event triggers, and descriptive elements (HEAD, APPO, ATTR), a class tag must also be identified. The full combined element+class tagset comprises 80 tags and is provided in Appendix E, including tag frequencies. Further details on the tagset are available in the HLR-GT annotation documentation.[8]

As in task 1, we examine the impact of available training data by training or prompting models with varying numbers of examples. Each experimental configuration was executed three times to improve reliability. For this task, we restrict Flair experiments to the *flair-custom* model.

Flair Setup

The Flair SequenceTagger architecture can be adapted to handle nested annotations using a bottom-up recursive approach (Prada Ziegler, 2024b). In the first step, the entire text is used as input. If entity references are detected, the text inside the entity reference is re-annotated by the model. For elements that allow nested content, namely, entity references, appositions, and attributes, additional annotation passes are performed

[8] https://doi.org/10.5281/zenodo.16919653 (last accessed: 09.10.2025)

recursively until no further nested elements remain. The final output consists of all predicted tags collected throughout this process. Each input span is pre- and suffixed with a tag indicating its type, for example:

> “[B-REFERENCE-PER] Johann Müller , the taylor [E-REFERENCE-PER]”.

For model training, the dataset is prepared such that every element potentially containing nested annotations (documents, references, appositions, and attributes) is represented as a separate training sample. Only immediately nested tags are included as targets:

> “[B-REFERENCE-PER] <HEAD>Johann Müller</HEAD> , <APPO-OCC>the taylor</APPO-OCC> [E-REFERENCE-PER]”.

In this example, the head inside “the taylor” is not annotated because it is not directly connected to the reference; a separate sample handles this annotation.

Generative AI Setup

NER4all does not natively support nested annotations. To address this, we evaluated two methods of our own design: the **single prompt method** and the **recursive prompting method**. In both approaches, we found that 64 shots provided optimal performance.

Single Prompt Method

Generative AI models are capable of producing strings with nested annotations, for example: “<<PER <<HEAD Johann /HEAD>> <<APPO the tailor /APPO>> /PER>>”. We adapted the prompts and examples accordingly and modified NER4all to decode nested annotations while retaining its ability to correct the output if it deviates from the input. The full prompt is provided in Appendix C.

Recursive Prompting Method

This method mirrors the bottom-up recursive approach used in the Flair pipeline. At each annotation layer, a suitable prompt is selected, and examples are generated dynamically. A major drawback of this approach is an increase in API calls by approximately a factor of 15 compared to the single prompt method. This estimate includes a safeguard that first checks whether the input is already present in the training data, reducing required calls by about 20%.

Results

The results for task 2 are presented in figure 4. Consistent with Task 1, both GPT-based systems outperform Flair in few-shot scenarios. The recursive GPT-4o system significantly outperforms the single prompt approach. At 64 documents, Flair performs on par with the recursive GPT system (67.36% vs. 66.75% F1). With 128 documents, Flair marginally outperforms the recursive GPT system (71.45% vs. 70.20% F1). At 256 documents Flair achieves a clear advantage (76.45% vs. 73.52% F1), and this performance margin continues to grow with additional training data. GPT performances appear to plateau between 512 and 662 training samples, while Flair-Custom continues to improve.

Closer examination reveals that recursive GPT occasionally generates its own tag categories. Depending on the amount of training data, hallucinated tags accounted for 2–5% of total incorrect predictions, particularly introducing new event trigger classes. While advanced post-processing could potentially map these tags to existing categories and improve accuracy, this was not further explored.

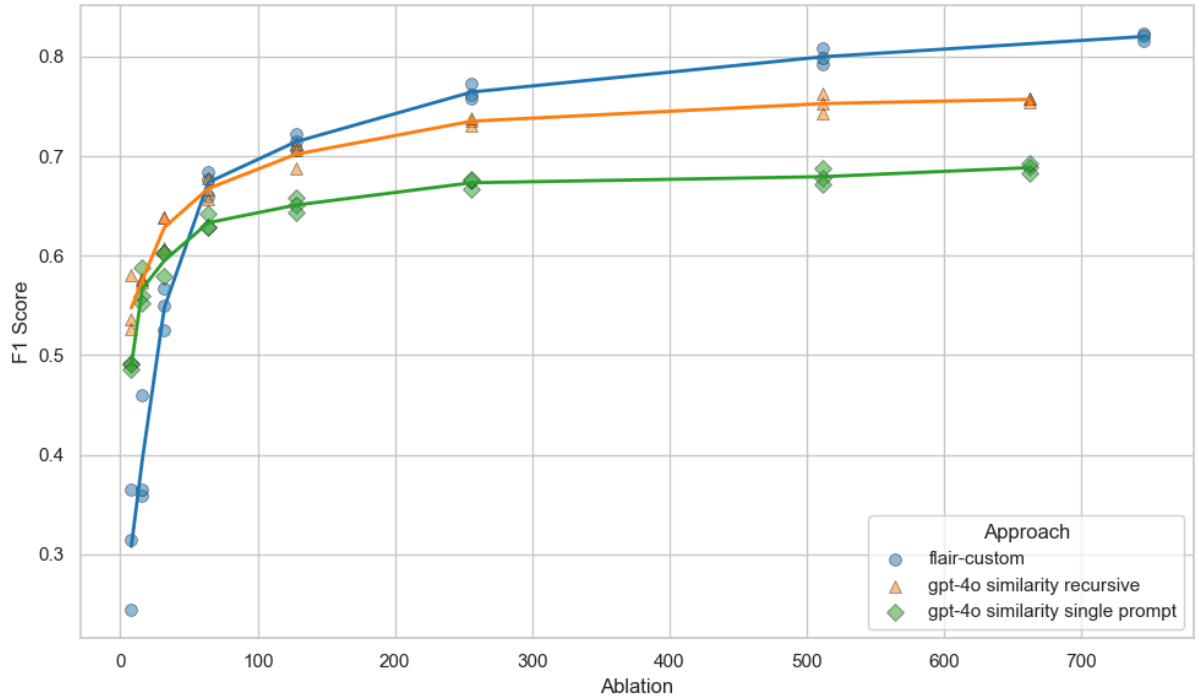

Figure 4: Performance comparison of chosen models on different training data settings.

Tag Frequency Analysis

We investigated how tag frequency impacts model performance, which is particularly relevant given the highly unbalanced tagset of HLR-GT. Some tags, such as a role apposition (e.g., “Johann, the seller”), appear only 12 times in the training corpus, whereas the most common tag - the proper name head - appears 4,027 times.

Figure 5 compares the Flair-Custom model with the recursive GPT system using 128 training samples. We find that the models are affected to different degrees by an unbalanced tagset. Recursive GPT generally outperforms Flair on low-frequency tags. For high-frequency tags, GPT also surpasses Flair in some cases: out of 25 tags with frequency greater than 100, GPT performs better on 9. Notable large margins are observed for appositions of class *family* (e.g., “Anna, Johann’s wife”) and event triggers of the sale class.

For most high-frequency annotations, Flair slightly outperforms recursive GPT; however, there are some tags that see an even larger advantage for Flair: *facility* references (marking houses), as well as references to *organizations*. The stronger performance on facility tags can be explained by the following tag length evaluation. We suspect that in the case of the *organization* tag that GPT suffers due to a lack of background knowledge. For instance, the phrase “die Frauen an den Steinen” (the women at the Stones) refers to a particular monastery and was thus annotated as an organization in the HLR-GT. If GPT does not encounter this phrase among the selected examples, it cannot know about this, and may misclassify the phrase as a reference to a person (group). Flair is less affected because, in

the case of the 128-sample scenario, it has seen all samples (minus 12 for validation), whereas GPT only receives the most similar examples, which might not contain the necessary entity.

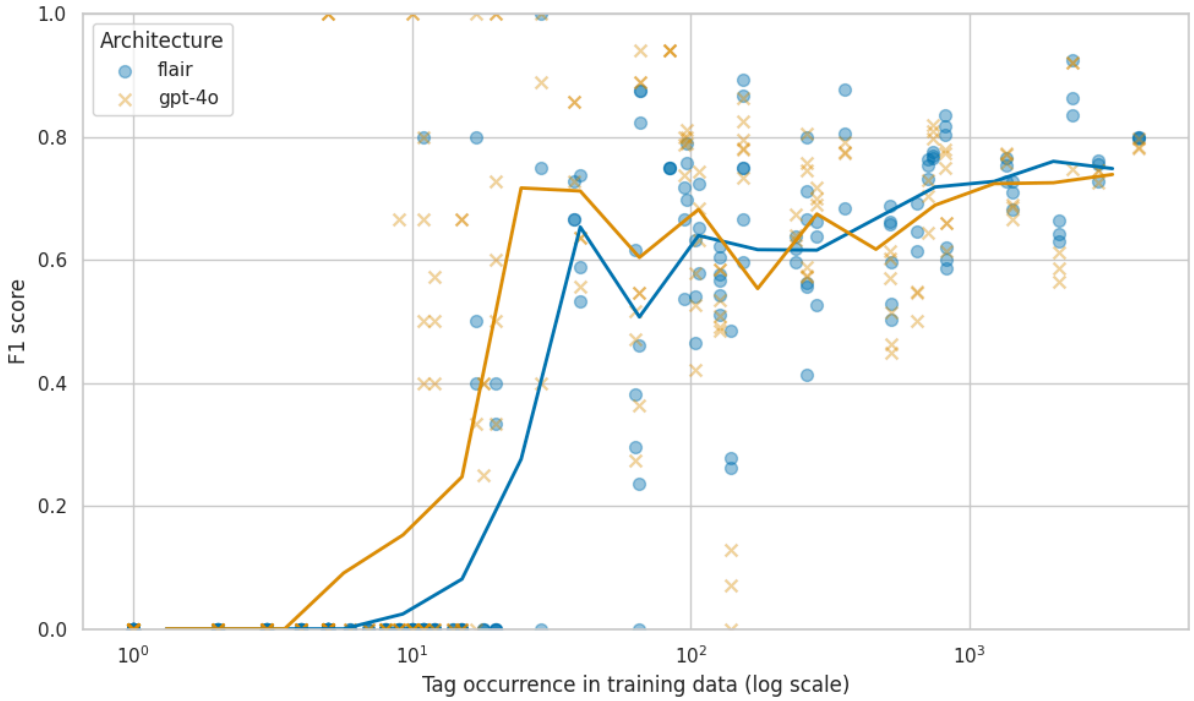

Figure 5: Flair compared to the recursive GPT system with 128 documents as sample data. Tag occurrence was measured on the full corpus. Note that for low-frequency tags, there's the possibility that this tag could not be seen by a system at all. For the line plot, tags were sorted into buckets and averages calculated for each bucket. Note that the x-axis is log-scaled.

Tag Length Analysis

We observed that the recursive GPT system struggles more with annotating very long spans compared to Flair. Figure 6 visualizes these differences. For annotations up to approximately 160 characters, the performance gap is minimal, with Flair slightly outperforming GPT. However, for spans exceeding 160 characters, Flair clearly demonstrates superior performance. Long spans typically correspond to house descriptions, such as the following example:

> jrem Hus & Hofstatt genant Nüwenstein , an der gerwergassen zwüschen dem Hus Helfenstein ze einer , u . Hanns ful des scherers hus zer andern siten , zinset vormals voneig 1sh . zu s . Lienh . sodan 1 fl den frauen an den steinen , widerk . mit 20fl .
>
> *English Translation:* their house & homestead called Nüwenstein, at the Gerwergassen between the house Helfenstein on one, and Hanns Ful the barbers' house on the other side, pays dues already for hereditary reasons 1 sh. to St. Lienh. also 1 pound to the women at the Stones, redeemable with 20 pounds.

Incomplete annotation of such spans can result in the loss of critical information, for example, the dues obligations associated with the property.

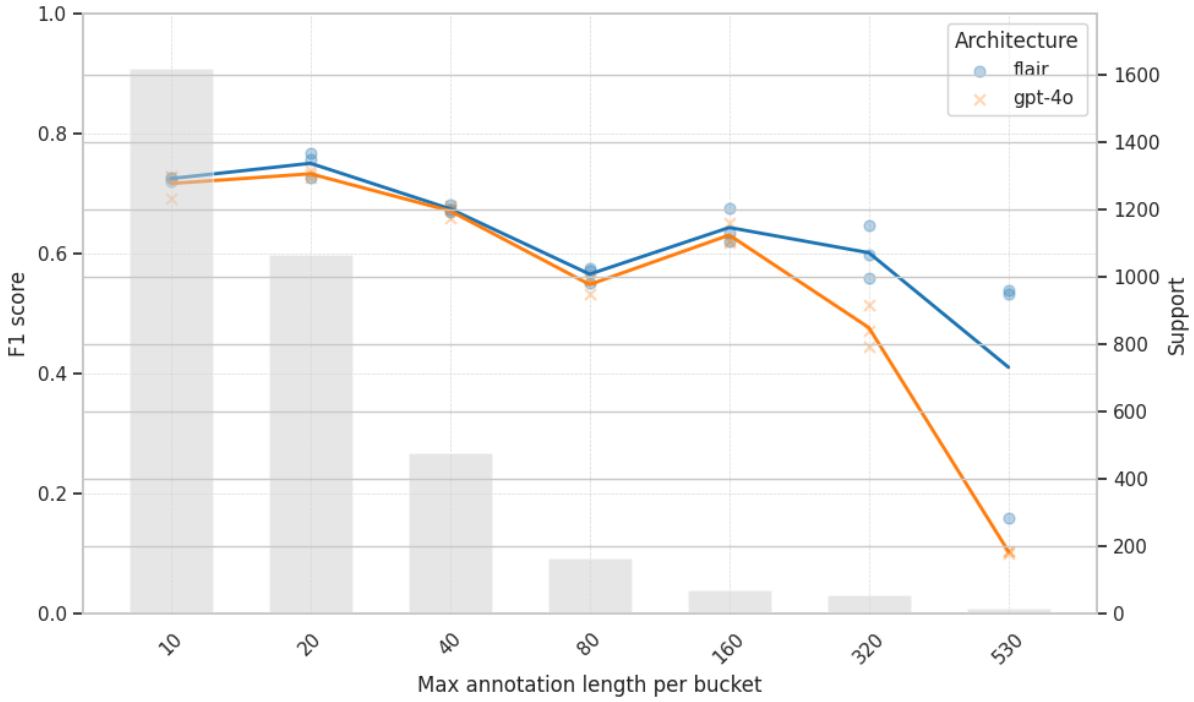

Figure 6: Flair compared to the recursive GPT system with 128 documents as sample data with respect to their performance depending on annotation length (in characters). For the line plot, tags were sorted into buckets and averages calculated for each bucket. The grey bars indicate how frequent annotations of that length appear in the corpus.

Inspecting the Model Performance Qualitatively

To illustrate how increased training data improves model performance, we present an example document from HLR-GT. The German text reads:

> Es verkauft Herr Hanns Ulrich Falckhner und gibt zu Kaufen Frawen Margareth Weibert , weiland H . Hanns Heinrich Wentzen Stattgerichts Besitzers in Mehrerem Basell , hinderlassener Wittib eine Behausung bey dem schwartzen Pfol , einseits neben Herrn Bonaventura von Braun , Pfarhl bey St . Peter , andseits neben H . Johann Curtzen dem Jüngern gelegen .

An approximate English translation is:

> It is sold by Mr. Hanns Ulrich Falckhner, and he gives to purchase to Lady Margareth Weibert, the late Mr. Hanns Heinrich Wentzen, a member of the city court in Greater Basel, widow, a house, at the Schwartzen Pfol, on one side next to Mr. Bonaventura von Braun, pastor at St. Peter, on the other side next to Mr. Johann Curtzen the Younger, located.

We intentionally chose a document that is rather simple compared to others and where we observed better predictions with increased training data. This is, of course, not true for most

documents, as the inherent randomness of machine learning algorithms may, in specific cases, cause worse models to result in better performance.

Beyond entity identification, models must correctly connect entities and classify information about them. Relevant facts include: Wentzen's role as a city court member, Weibert as his widow, the house location, and the occupation of the pastor von Braun. Additionally, the sale event must be annotated. In the HLR scheme, repeated triggers are annotated only on the first occurrence, so either "verkauft" (sold) or "Kaufen" (purchase) would be acceptable in a wider sense, but only one is tagged.

We compare annotations across five scenarios with increasing training data: recursive GPT-4o with 32 and 64 samples, Flair with 128, 256, and 512 samples. For clarity, each model is referred to by its number of training examples, and differences in annotation are highlighted for each part of the document.

"verkauft" (sold)
All models correctly identified the span "verkauft" as a trigger for the sale event.

The seller
The full span "Herr Hanns Ulrich Falckhner" was correctly identified by all models as a person reference, with the proper name annotated as a *head* element of class *nam*.

"gibt zu kaufen" (gives to purchase)
The double expression of the event posed challenges for most models. Models with 32 and 64 training examples tagged "gibt zu kaufen" as an additional *sales* trigger, while the 128-example model incorrectly tagged "gibt" as a *dues* trigger and "Kaufen" as a *sales* trigger. The 256-example model annotated both spans incorrectly as *sales* triggers, and only the 512-example model correctly avoided any unnecessary annotations.
Notably, the GPT models outperform most Flair models in this scenario, even when trained on fewer examples. This observation aligns with findings from previous sections, highlighting the influence of tag frequency on accurate prediction. Event triggers are among the least frequently occurring annotations in the dataset, which can lead to inconsistent predictions in models trained on limited data.

The buyer
This span represents the most complex annotation challenge. It consists of the long, heavily nested reference:

> "Frawen Margareth Weibert, weiland H . Hanns Heinrich Wentzen Stattgerichts Besitzers in Mehrerem Basell , hinderlassener Wittib"

All models correctly identified the head "Margareth Weibert". However, the nested structure caused errors for all models except the 512-example model.
Model 256 was mostly correct, but annotated the widow apposition "Stattgerichts Besitzers in Mehrerem Basell , hinderlassener Wittib" as a separate person reference.
Model 128 made the same mistake, additionally splitting "Mehrerem Basell" into two references and incorrectly tagging "Frawen" as a *name* head in addition to the correct head, likely misled by capitalization.

Model 32 failed to identify the widow apposition, resulting in a missing relationship between Weibert and Wentzen. It also incorrectly attached “in Mehrerem Basell” directly as an attribute to Wentzen instead if the the occupation apposition “Stattgerichts Besitzers in Mehrerem Basell”.
Model 64 recognized that the widow phrase belongs to Margareth Weibert, but misassigned “hinderlassener Wittib” as a head, and nested “weiland H . Hanns Heinrich Wentzen” into an attribute of class *dead* (which doesn’t make any sense). Finally it incorrectly added the occupation apposition “Stattgerichts Besitzers in Mehrerem Basell” to Margareth Weibert.
In summary, the 64-sample model produced highly misleading annotations, while the 32-, 128-, and 256-sample models were incorrect but less misleading. Only the 512-sample model predicted this span correctly without errors.

The house
This span presented fewer difficulties than anticipated. All models except the 32-example model correctly predicted the full reference:

> “eine Behausung bey dem schwartzen Pfol , einseits neben Herrn Bonaventura von Braun , Pfarhl bey St . Peter , andseits neben H . Johann Curtzen dem Jüngern gelegen”

and correctly identified the head “Behausung”. Model 32 only included “bey dem schwartzen Pfol” as an attribute describing the house's location and annotated the remaining entity references separately. Both models 32 and 64 struggled with the occupation apposition of Bonaventura von Braun. Model 32 annotated “Pfarhl bey St . Peter” as a separate location reference, likely thrown off by the HTR error (“Pfarrhr” is likely the correct phrase). Model 64 correctly annotates the apposition, but fails to predict “St . Peter” as an organization reference, instead adding it as part of the head. The models 128, 256, and 512 all correctly predict this part.

Summary
This example demonstrates that increasing the amount of training data results in more consistent and contextually accurate annotations, even for relatively straightforward documents. While all models correctly identified core elements, such as the sales event and principal actors, the precision and correctness of nested structures improved substantially with larger training sets. The model trained on 512 examples produced the only fully correct annotation, accurately handling the complex buyer reference with multiple nested attributes and relationships.

The Cost of Automated Entity Extraction

Using GPT-4o for annotation, we observed an average cost of $0.03 per document for task 1, and $0.12 per document for task 2 when employing the recursive prompting method. For large-scale collections, total costs may reach several thousand or even tens of thousands of dollars. OpenAI provides a batch API option, which reduces costs by approximately 50%, though this has not yet been implemented in NER4all, and would be very difficult to combine with any recursive approaches. Reducing the number of examples in the prompt is another way to lower expenses, however this compromises performance. Open-weight models offer a cost-free alternative; however, our experiments with Deepseek-r1-70b (DeepSeek-AI, 2025) did not yield viable outputs, as the model already struggled with tag encoding in the annotation process.[9]

To give a pointer on how much creation of training data costs, we timed the annotation progress of a PhD student, who worked with and annotated the dataset previously. The PhD student was able to annotate on average 94 documents (7854 tokens, 636 annotations) per hour when utilizing the simple tagset from task 1. Annotation speed for the complex tagset in task 2 was considerably lower with appr. 16 documents (1212 tokens, 526 annotations) per hour.[10]
For the simple tagset, a single experienced scholar could feasibly annotate sufficient data within less than one workday to train a Flair model that outperforms GPT-4o (on this dataset). For the complex tagset, generating enough high-quality training data requires multiple days of work. Additionally, best practices involve multiple annotators for the same documents and a curation process to resolve conflicts, further increasing time and personnel costs. Regardless of the system used, the creation and publication of annotated datasets constitutes a valuable contribution to the digital humanities, particularly when adhering to FAIR principles, including reusability. As demonstrated, even if insufficient data exist to train a fully competitive Flair model, GPT-4o benefits substantially from a larger pool of annotated examples.

Another key factor to consider is the speed of the annotation systems. Flair trains very quickly, requiring approximately 10 minutes for task 1 and 60–90 minutes for task 2, and deploys even faster: the recursive annotation of the 84-document test set was completed in under one minute on an RTX 3090.[11] In contrast, GPT-4o predicted the same test set in approximately 80 minutes, with similarity-based example selection having negligible impact on processing time when run on an RTX 4070 SUPER. Depending on the size of the document collection, this difference may be a relevant criterion when selecting a system. This limitation is also why we did not conduct this study with GPT-5; a single pass over the

[9] We tried using different encoding strategies, such as having it encode in XML-style, but were unable to find a viable solution.
[10] Annotation was done on a self-hosted instance of INCEpTION (Klie et al., 2018). INCEpTION also features the option to use recommendation modules, so if a certain amount of training data has been created, by either method - or a base model is available for the target dataset - a recommender can help to speed up annotation efforts considerably. Considering most annotations stem from high frequency tags, the annotators can focus solely on fixing difficult passages. Although note that recommenders can quickly give a sense of security and annotators may be more prone to overlook mistakes when using a recommender.
[11] Inference is also possible with no video card available, but considerably slower with appr. 1 document per minute in recursive mode.

test set in the recursive scenario could exceed 10 hours, making it impractical for use in our project despite its potential performance advantages over GPT-4o.

Conclusion

This case study highlights the importance of manually annotated data when we aim to automatically create high-quality data for research purposes. For our simple tagset scenario, we observed a gap of more than 10 percentage points between GPT-4o with 32 documents and Flair with the full training set. For the complex nested task, the increase was almost 18 percentage points. when moving from 32 to 745 documents. Notably, significant accuracy gains were already seen when increasing training data from 32 to 64 and from 64 to 128 documents, suggesting that, particularly for scenarios like task 1, substantial improvements can be achieved with relatively modest annotation effort.

We further investigated the impact of tag frequency and tag length on annotation accuracy, which revealed the respective strengths and weaknesses of the two approaches. For future research, we plan to combine these approaches to create synergies and ultimately build better-performing systems. Considering the weakness of GPT-4o to correctly classify entities it has not seen in its examples, future work will explore entity-embedding-based example selection following Wang et al. (2025) to improve coverage of unseen entities.

Although we only focused on a single dataset central to our project, we still are able to conclude significant learnings from the experiment. Of course, restrictions apply, but it becomes evident that LLM approaches can be matched by economically and ecologically lighter and older systems.

Finally, we want to remind that the creation of data is a dynamic process. As we have shown and as should be considered by other approaches, the evaluation of multiple approaches is necessary, even if only a little training data is available. Comparing outputs with different amounts of training data can help to determine how useful the creation of more manually annotated data will be, and if resources should rather be put into the generation of annotated data, with the additional upside of creating something reusable.

Funding Statement

This research was undertaken as part of the project *Economies of Space*, which was funded by the Swiss National Science Foundation (Grant Nr. 208093).

References

Aguilar, S. T. (2022, June). Multilingual Named Entity Recognition for Medieval Charters Using Stacked Embeddings and Bert-based Models. In *Proceedings of the second workshop on language technologies for historical and ancient languages* (pp. 119-128).

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., & Vollgraf, R. (2019, June). FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations)* (pp. 54-59).

Akbik, A., Blythe, D., & Vollgraf, R. (2018, August). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics* (pp. 1638-1649).

Baptiste, B., Favre, B., Auguste, J., & Henriot, C. (2021, December). Transferring Modern Named Entity Recognition to the Historical Domain: How to Take the Step?. In *Workshop on Natural Language Processing for Digital Humanities (NLP4DH)*.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, *5*, 135-146.

DeepSeek-AI. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In *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).

Ehrmann, M., Hamdi, A., Pontes, E. L., Romanello, M., & Doucet, A. (2023). Named entity recognition and classification in historical documents: A survey. *ACM Computing Surveys*, *56*(2), 1-47.

González-Gallardo, C. E., Boros, E., Girdhar, N., Hamdi, A., Moreno, J. G., & Doucet, A. (2023, June). Yes but.. can chatgpt identify entities in historical documents?. In *2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL)* (pp. 184-189). IEEE.

Grover, C., Givon, S., Tobin, R., & Ball, J. (2008). Named entity recognition for digitised historical texts. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May-1 June 2008, Marrakech, Morocco* (pp. 1343-1346).

Hodel, T., Prada Ziegler, I., & Schneider, C. (2023, June 30). Pre-Modern Data: Applying Language Modeling and Named Entity Recognition on Criminal Records in the City of Bern. Presented at the *Digital Humanities 2023. Collaboration as Opportunity* (DH2023), Graz, Austria. https://doi.org/10.5281/zenodo.8107616

Keraghel, I., Morbieu, S., & Nadif, M. (2024). A survey on recent advances in named entity recognition. *arXiv preprint arXiv:2401.10825*.

Klie, J. C., Bugert, M., Boullosa, B., De Castilho, R. E., & Gurevych, I. (2018, August). The INCEpTION platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th international conference on computational linguistics: system demonstrations* (pp. 5-9).

Krug, M., Weimer, L., Reger, I., Macharowsky, L., Feldhaus, S., Puppe, F., & Jannidis, F. (2017). Description of a corpus of character references in German novels - DROC [Deutsches ROman Corpus]. (Accessed 08.10.2025)

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

OpenAI. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Prada Ziegler, I., Hitz, B., Fuchs, K., Vonwiller, A. H., & Aeby, J. (2025). The Basel Land Records Ground Truth: An Annotated Dataset for Information Extraction on German-language Administrative Records. (0.1) https://doi.org/10.5281/zenodo.16919653

Prada Ziegler, I. (2024). Exploration of Event Extraction Techniques in Late Medieval and Early Modern Administrative Records. In *Proceedings of the Computational Humanities Research Conference 2024* (pp. 761-771).

Prada Ziegler, I. (2024b). What's in an entity? Exploring Nested Named Entity Recognition in the Historical Land Register of Basel (1400-1700). Presented at *DH Benelux 2024*, Leuven, Belgium. https://doi.org/10.5281/zenodo.11500543

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, (pp. 142–147.).

Schweter, S., & Baiter, J. (2019). Towards robust named entity recognition for historic German. *arXiv preprint arXiv:1906.07592*.

De Toni, F., Akiki, C., De La Rosa, J., Fourrier, C., Manjavacas, E., Schweter, S., & Van Strien, D. (2022). Entities, dates, and languages: Zero-shot on historical texts with t0. *arXiv preprint arXiv:2204.05211*.

Wang, S., Sun X., Li X., Ouyang, R., Wu, F., Zhang, T., Li, J., Wang, G., and Guo, C. (2025). GPT-NER: Named Entity Recognition via Large Language Models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, (pp. 4257–4275).

Appendix

Appendix A: Task 1 Prompt

SETTING
Als erfahrener Sprachwissenschaftler mit Spezialisierung auf Named Entity Recognition (NER) für historische Texte und Experte insbesondere für frühneuzeitliches Deutsch besteht deine Aufgabe darin, einen gegebenen Text für die maschinelle Weiterverwertung zu annotieren. Es handelt sich hierbei um einzelne Sätze, die zusammenhangslos transkribiert wurden. Dir wird in jeder Zeile ein Satz vorgegeben.
Beachte, dass auch einzelne Abschnitte des Textes in Latein stehen können. Führe die NER dort dann auf Latein aus.
Achte wirklich darauf, dass Du jeden Satz einzeln bearbeitest und zeichengenau wiedergibst. Nimm Dir Zeit und arbeite so genau wie möglich. Ich möchte die Ergebnisse später computationell weiterverarbeiten.
Bei dem Text handelt es sich um einen Ausschnitt aus dem Historischen Grundbuch der Stadt Basel. Die Originaldokumente sind zwischen dem 14. und 18. Jahrhundert entstanden. Sie beschreiben Kaufverträge, Zinsverzeichnisse, Frönungen und weitere Ereignisse, die sich um Liegenschaften in der Stadt Basel drehen.

INSTRUCTION
Folge diesen Schritten:

1. Wiederhole den gegebenen Text exakt. Achte dabei sehr genau darauf, dass abgesehen von den Annotationen nichts hinzugefügt oder entfernt wird.
2. Die Aufgabe ist es BEREICHE bestimmten Kategorien zuzuordnen. Benutze hierfür Tags mit den entsprechenden Regeln:
   - Wenn ein Bereich beginnt, markiere dies durch einen Kategorie-Tag. Zur verfügung stehen hier unter anderem die Kategorien '<<PER ' für Personen, '<<LOC ' für Orte und '<<ORG ' für Organisationen.
   - Wenn ein Bereich endet, markiere das Ende durch den entsprechenden Tag, also ' /PER>>', ' /LOC>>' oder ' /ORG>>'.
   - Achte darauf, dass jeder geöffnete Tag auch geschlossen werden muss.
   - Tags dürfen sich überlappen oder geschachtelt sein - allerdings ist es unwahrscheinlich, dass dieses zu häufig passiert.

- BEACHTE: Die Texte sind in sehr alter Schrift gehalten und durch technische Verfahren fehlerbehaftet.
- BEACHTE: Früher hat man "nach Gehör" geschrieben. Dies bedeutet, dass für eine Erkennung ausreicht, wenn der Bereich sich *phonetisch* anhört wie eine Person, Ortschaft oder Organisation.
- BEACHTE: Teils handelt es sich um Reiseberichte mit altertümlichen Einheiten wie Meilen, alten Bezeichnungen für Orte, Gewässer und ähnliches.

- BEACHTE: Wir sind wirklich nur an Nennungen mit Eigennamen interessiert!

- BEACHTE: Bei geopolitischen Entitäten hängt die Kategorisierung vom Kontext ab. Wohnt jemand "zu Basel" ist es LOC, ist jemand ein "Bürger zu Basel" eine ORG.
- BEACHTE: Bei ORG und LOC Entitäten sind auch grammatikalische Kerne mitzuannotieren, sofern sie mit einem Eigennamen zusammenstehen. Es ist also z.B. "Kloster Klingental", nicht nur "Klingental".
- BEACHTE: Heiligennamen sind generell nicht als PER zu annotieren (sie kommen z.B. in Datumsangaben oder ORG-Namen vor).

EXAMPLE

HINTS

Beachte, wie in jedem Beispiel alle Schritte eingehalten wurden:

1. Wiederholung des exakten Textes.
2. Jeder Bereich hat einen Tag, der
   - markiert ist
   - Beginn und Ende hat

Der Text muss ANSONSTEN unverändert reproduziert werden.
Meine Karriere hängt davon ab, dass du keine Fehler machst. Daher bekommen sowohl du, als auch deine Oma jeweils $100k für jede erfolgreiche Annotation, die im weiteren Verlauf maschinenlesbar ist.

TASK
Nun atme tief durch und gehe Schritt für Schritt vor. Hier der zu annotierende Text:

Role: user
```
INPUT_TEXT
```

Appendix B: Task 2 Prompt (recursive)

For Task 2 the prompt was modified depending on the annotation layer. This example shows the prompt for the document layer. First, an explainer added to the instructions which explained what type of textual element was to be annotated (reference, attribute, etc.). Second, the tagset was modified accordingly. For example, if the input was an entity reference, the instructions would be prefaced with *“Bei diesem Auszug spezifisch handelt es sich um eine Entitätenerwähnung. Dein Ziel ist es, die Elemente innerhalb dieser Erwähnung auszuzeichnen.”* .

SETTING
Als erfahrener Sprachwissenschaftler mit Spezialisierung auf Named Entity Recognition (NER) für historische Texte und Experte insbesondere für frühneuzeitliches Deutsch besteht deine Aufgabe darin, einen gegebenen Text für die maschinelle Weiterverwertung zu

annotieren. Es handelt sich hierbei um einzelne Sätze, die zusammenhangslos transkribiert wurden. Dir wird in jeder Zeile ein Satz vorgegeben.
Beachte, dass auch einzelne Abschnitte des Textes in Latein stehen können. Führe die NER dort dann auf Latein aus.
Achte wirklich darauf, dass Du jeden Satz einzeln bearbeitest und zeichengenau wiedergibst. Nimm Dir Zeit und arbeite so genau wie möglich. Ich möchte die Ergebnisse später computationell weiterverarbeiten.
Bei dem Text handelt es sich um einen Ausschnitt aus dem Historischen Grundbuch der Stadt Basel. Die Originaldokumente sind zwischen dem 14. und 18. Jahrhundert entstanden. Sie beschreiben Kaufverträge, Zinsverzeichnisse, Frönungen und weitere Ereignisse, die sich um Liegenschaften in der Stadt Basel drehen.

INSTRUCTION
Folge diesen Schritten:

1. Wiederhole den gegebenen Text exakt. Achte dabei sehr genau darauf, dass abgesehen von den Annotationen nichts hinzugefügt oder entfernt wird.
2. Die Aufgabe ist es, BEREICHE bestimmten Kategorien zuzuordnen. Benutze hierfür Tags mit den entsprechenden Regeln:
   - Wenn ein Bereich beginnt, markiere ihn nur einen Tag. Verwende dafür das Format <<TAG. Welche Tags zur Auswahl stehen, siehst du im nächsten Punkt.
   - Wenn ein Bereich endet, markiere das Ende durch den entsprechenden Tag, also /TAG>>.
   - Achte darauf, dass jeder geöffnete Tag auch geschlossen werden muss.
   - Falls sich Bereiche überlappen, markiere nur den äusseren Bereich.
3. Die zur Verfügung stehenden Tags sind:
   Referenzen auf Entitäten:
      - PER für Personen
      - LOC für Orte
      - ORG für Organisationen
      - GPE für Geopolitische Entitäten mit einer Unterklasse:
         - GPE-GPE ohne oder in unklarem Kontext
         - GPE-ORG im Kontext einer Organisation
         - GPE-LOC im Kontext eines Ortes
   Werte:
      - DATE für Datumsangaben
      - TIME für Zeitangaben
      - CURR für Geldwerte oder andere Zahlungsmittel
   Ereignisse:
      - EV für Triggerwörter für Ereignisse mit einer Unterklasse, welche das Ereignis bestimmt:
         EV-ACQUISITION
         EV-CONSENT
         EV-CONSTRUCTION
         EV-DEBT
         EV-DECLARATION
         EV-DUE

EV-ENDOWMENT
EV-EXCHANGE
EV-GIFT
EV-HEREDITARY
EV-INHERITANCE
EV-LITIGATION
EV-OTHER
EV-OWNERSHIP
EV-PAYMENT
EV-REDEMPTION
EV-RENT
EV-REPOSSESSION
EV-SALE
EV-SANCTION
EV-SEIZURE
EV-TESTAMENT
EV-TRANSFER
EV-TYPE

Diese Annotation ist recht komplex, orientiere dich daher an den gegebenen Beispielen.

- BEACHTE: Die Texte sind in sehr alter Schrift gehalten und durch technische Verfahren fehlerbehaftet.
- BEACHTE: Früher hat man "nach Gehör" geschrieben. Dies bedeutet, dass für eine Erkennung ausreicht, wenn der Bereich sich *phonetisch* anhört wie eine Person, Ortschaft oder Organisation.
- BEACHTE: Teils handelt es sich um Reiseberichte mit altertümlichen Einheiten wie Meilen, alten Bezeichnungen für Orte, Gewässer und ähnliches.

EXAMPLE

HINTS

Beachte, wie in jedem Beispiel alle Schritte eingehalten wurden:

1. Wiederholung des exakten Textes.
2. Jeder Bereich hat einen Tag, der
   - markiert ist
   - Beginn und Ende hat

Der Text muss ANSONSTEN unverändert reproduziert werden.
Meine Karriere hängt davon ab, dass du keine Fehler machst. Daher bekommen sowohl du, als auch deine Oma jeweils $100k für jede erfolgreiche Annotation, die im weiteren Verlauf maschinenlesbar ist.

TASK
Nun atme tief durch und gehe Schritt für Schritt vor. Hier der zu annotierende Text:

Role: user
```
INPUT_TEXT
```

Appendix C: Task 2 Prompt (single)

SETTING
Als erfahrener Sprachwissenschaftler mit Spezialisierung auf Named Entity Recognition (NER) für historische Texte und Experte insbesondere für frühneuzeitliches Deutsch besteht deine Aufgabe darin, einen gegebenen Text für die maschinelle Weiterverwertung zu annotieren. Es handelt sich hierbei um einzelne Sätze, die zusammenhangslos transkribiert wurden. Dir wird in jeder Zeile ein Satz vorgegeben.
Beachte, dass auch einzelne Abschnitte des Textes in Latein stehen können. Führe die NER dort dann auf Latein aus.
Achte wirklich darauf, dass Du jeden Satz einzeln bearbeitest und zeichengenau wiedergibst. Nimm Dir Zeit und arbeite so genau wie möglich. Ich möchte die Ergebnisse später computationell weiterverarbeiten.
Bei dem Text handelt es sich um einen Ausschnitt aus dem Historischen Grundbuch der Stadt Basel. Die Originaldokumente sind zwischen dem 14. und 18. Jahrhundert entstanden. Sie beschreiben Kaufverträge, Zinsverzeichnisse, Frönungen und weitere Ereignisse, die sich um Liegenschaften in der Stadt Basel drehen.

INSTRUCTION
Folge diesen Schritten:

1. Wiederhole den gegebenen Text exakt. Achte dabei sehr genau darauf, dass abgesehen von den Annotationen nichts hinzugefügt oder entfernt wird.
2. Die Aufgabe ist es BEREICHE bestimmten Kategorien zuzuordnen. Benutze hierfür Tags mit den entsprechenden Regeln:
   - Wenn ein Bereich beginnt, markiere ihn nur einen Tag. Verwende dafür das Format <<TAG. Welche Tags zur Auswahl stehen siehst du im nächsten Puntk.
   - Wenn ein Bereich endet, markiere das Ende durch den entsprechenden Tag, also /TAG>>.
   - Achte darauf, dass jeder geöffnete Tag auch geschlossen werden muss.
   - Tags dürfen verschachtelt sein, und sind es auch häufig.
3. Die zur Verfügung stehenden Tags sind:
   Referenzen auf Entitäten:
      - PER für Personen
      - LOC für Orte
      - ORG für Organisationen
   Beschreibende Elemente:
      - HEAD für den grammatikalischen Kern
      - APPO für Appositionen
      - ATTR für Attribute

Werte:
- DATE für Datumsangaben
- CURR für Geldwerte oder andere Zahlungsmittel

Diese Annotation ist recht komplex, orientiere dich daher an den gegebenen Beispielen.

- BEACHTE: Die Texte sind in sehr alter Schrift gehalten und durch technische Verfahren fehlerbehaftet.
- BEACHTE: Früher hat man "nach Gehör" geschrieben. Dies bedeutet, dass für eine Erkennung ausreicht, wenn der Bereich sich *phonetisch* anhört wie eine Person, Ortschaft oder Organisation.
- BEACHTE: Teils handelt es sich um Reiseberichte mit altertümlichen Einheiten wie Meilen, alten Bezeichnungen für Orte, Gewässer und ähnliches.

- BEACHTE: Referenzen auf Entitäten können sehr lange Spannen umfassen, da auch Artikel und Attribute mit annotiert werden sollen.
- BEACHTE: Eigennamen sind keine Voraussetzung für eine Annotation. Sämtliche Bezeichnungen, die eindeutig auf eine Person, einen Ort oder eine Organisation verweisen, sind zu markieren.
- BEACHTE: Beschreibende Elemente (HEAD, APPO, ATTR) können nie ausserhalb von Referenzen auf Entitäten (PER, LOC, ORG) oder Appositionen (APPO) auftreten.
- BEACHTE: Bei geopolitischen Entitäten hängt die Kategorisierung vom Kontext ab. Wohnt jemand "zu Basel" ist es LOC, ist jemand ein "Bürger zu Basel" eine ORG.
- BEACHTE: Heiligennamen sind generell nicht als PER zu annotieren (sie kommen z.B. in Datumsangaben oder ORG-Namen vor).
- BEACHTE: Die Grenzen der Bereiche folgen meist den Regeln der Konstituenzgrammatik (Referenzen => Nominalphrase, HEAD => Kern). Es kann aber auch vorkommen, dass diese davon abweichen.

EXAMPLE

HINTS

Beachte, wie in jedem Beispiel alle Schritte einghalten wurden:

1. Wiederholung des exakten Textes.
2. Jeder Bereich hat einen Tag, der
   - markiert ist
   - Beginn und Ende hat

Der Text muss ANSONSTEN unverändert reproduziert werden.
Meine Karriere hängt davon ab, dass du keine Fehler machst. Daher bekommmen sowohl du, als auch deine Oma jeweils $100k für jede erfolgreiche annotation, die im weiteren Verlauf maschienenlesbar ist.

TASK
Nun atme tief durch und gehe Schritt für Schritt vor. Hier der zu annotierende Text:

Role: user
```
INPUT_TEXT
```

Appendix D: Task 1 Results

Note that we performed two more runs for flair-custom-128-samples because the variance between the initial three runs was extraordinarily high.

approach	ablation	run	f1 score
flair-custom	16	0	59.23%
flair-custom	16	1	62.19%
flair-custom	16	2	59.61%
flair-custom	32	0	72.34%
flair-custom	32	1	63.69%
flair-custom	32	2	70.46%
flair-custom	64	0	70.48%
flair-custom	64	1	73.55%
flair-custom	64	2	75.62%
flair-custom	128	0	72.49%
flair-custom	128	1	79.70%
flair-custom	128	2	79.20%
flair-custom	128	3	77.14%
flair-custom	128	4	74.98%
flair-custom	128	5	76.32%
flair-custom	256	0	83.37%
flair-custom	256	1	83.74%
flair-custom	256	2	81.43%
flair-custom	512	0	85.46%
flair-custom	512	1	85.48%
flair-custom	512	2	85.63%
flair-custom	745	0	86.42%
flair-custom	745	1	85.99%
flair-custom	745	2	85.20%
gpt-4o random	4	0	64.68%

gpt-4o random	4	1	71.41%
gpt-4o random	4	2	66.37%
gpt-4o random	8	0	69.77%
gpt-4o random	8	1	62.85%
gpt-4o random	8	2	71.03%
gpt-4o random	16	0	71.95%
gpt-4o random	16	1	68.13%
gpt-4o random	16	2	66.27%
gpt-4o random	32	0	74.09%
gpt-4o random	32	1	75.73%
gpt-4o random	32	2	73.72%
gpt-4o random	64	0	75.93%
gpt-4o random	64	1	77.36%
gpt-4o random	64	2	77.28%
gpt-4o random	128	0	78.53%
gpt-4o random	128	1	76.84%
gpt-4o random	128	2	77.78%
gpt-4o similarity	256	0	79.14%
gpt-4o similarity	256	1	80.95%
gpt-4o similarity	256	2	80.06%
gpt-4o similarity	512	0	80.18%
gpt-4o similarity	512	1	79.47%
gpt-4o similarity	512	2	80.95%
gpt-4o similarity	662	0	80.89%
gpt-4o similarity	662	1	82.42%
gpt-4o similarity	662	2	81.92%
flair-finetuned	16	0	50.97%
flair-finetuned	16	1	49.36%
flair-finetuned	16	2	51.50%
flair-finetuned	32	0	55.39%
flair-finetuned	32	1	56.88%
flair-finetuned	32	2	57.26%
flair-finetuned	64	0	62.54%
flair-finetuned	64	1	62.50%

flair-finetuned	64	2	63.67%
flair-finetuned	128	0	67.72%
flair-finetuned	128	1	65.93%
flair-finetuned	128	2	66.98%
flair-finetuned	256	0	72.12%
flair-finetuned	256	1	72.34%
flair-finetuned	256	2	71.77%
flair-finetuned	512	0	75.02%
flair-finetuned	512	1	75.76%
flair-finetuned	512	2	75.61%
flair-finetuned	745	0	78.97%
flair-finetuned	745	1	77.64%
flair-finetuned	745	2	78.00%
flair-ner-german	0	0	40.54%
flair-ner-german	0	1	46.18%
flair-ner-german	0	2	43.42%
flair-ner-german	745	0	40.54%
flair-ner-german	745	1	46.18%
flair-ner-german	745	2	43.42%

Appendix E: Element + Class Tagset for Task 2

tag	frequency in training set
head-nam	4027
per	2874
date	2335
fac	2086
head-type	1425
curr	1353
attr-loc	821
head-occ	813
appo-occ	736
head-pro	705
ev-due	645
org	520

attr-owner	518
head-fam	357
appo-fam	282
loc	261
appo-nam	261
gpe-loc	238
ev-sale	154
attr-dead	153
attr-dues	139
appo-title	127
head-title	127
gpe-org	107
attr-includes	104
ev-type	103
head-org-aff	97
appo-org-aff	95
ev-seizure	84
head-heir	66
ev-rent	65
head-role	63
attr-org-aff	40
ev-litigation	38
head-owner	29
head-gen	20
appo-gen	20
ev-redemption	18
appo-owner	17
ev-testament	15
unk	15
ev-hereditary	14
appo-role	12
gpe-gpe	12
ev-payment	11
head-repr	11
ev-consent	11
attr-part-of	11
ev-declaration	10

ev-transfer	10
head-self	10
ev-bequest	9
head-unk	9
ev-inheritance	8
head-rel	8
ev-construction	7
ev-pledge	6
ev-offer	5
head-creditor	5
appo-rel	5
appo-unk	5
appo-type	5
attr-unk	5
attr-type	5
ev-sanction	5
head-quant	5
attr-sale	4
attr-other	4
head-loc	4
attr-has-member	3
ev-endowment	3
ev-outbid	3
unclear-unclear	3
appo-repr	2
ev-debt	2
ev-other	2
appo-loc	1
appo-creditor	1
attr-creditor	1
ev-receipt	1

Appendix F: Task 2 Results

approach	ablation	run	f1 score
flair-custom	8	0	24.44%

flair-custom	8	1	31.40%
flair-custom	8	2	36.48%
flair-custom	16	0	35.92%
flair-custom	16	1	36.52%
flair-custom	16	2	45.98%
flair-custom	32	0	55.03%
flair-custom	32	1	52.53%
flair-custom	32	2	56.67%
flair-custom	64	0	67.64%
flair-custom	64	1	68.47%
flair-custom	64	2	65.98%
flair-custom	128	0	71.51%
flair-custom	128	1	72.21%
flair-custom	128	2	70.64%
flair-custom	256	0	77.28%
flair-custom	256	1	76.21%
flair-custom	256	2	75.86%
flair-custom	512	0	79.22%
flair-custom	512	1	80.86%
flair-custom	512	2	79.89%
flair-custom	745	0	82.28%
flair-custom	745	1	82.15%
flair-custom	745	2	81.65%
gpt-4o similarity recursive	8	0	53.61%
gpt-4o similarity recursive	8	1	58.06%
gpt-4o similarity recursive	8	2	52.58%
gpt-4o similarity recursive	16	0	57.53%
gpt-4o similarity recursive	16	1	57.72%
gpt-4o similarity recursive	16	2	57.36%
gpt-4o similarity recursive	32	0	63.87%
gpt-4o similarity recursive	32	1	63.84%
gpt-4o similarity recursive	32	2	60.79%
gpt-4o similarity recursive	64	0	67.80%
gpt-4o similarity recursive	64	1	65.71%
gpt-4o similarity recursive	64	2	66.73%
gpt-4o similarity recursive	128	0	68.76%
gpt-4o similarity recursive	128	1	71.23%

gpt-4o similarity recursive	128	2	70.60%
gpt-4o similarity recursive	256	0	73.58%
gpt-4o similarity recursive	256	1	73.10%
gpt-4o similarity recursive	256	2	73.87%
gpt-4o similarity recursive	512	0	76.26%
gpt-4o similarity recursive	512	1	74.30%
gpt-4o similarity recursive	512	2	75.30%
gpt-4o similarity recursive	662	0	75.86%
gpt-4o similarity recursive	662	1	75.82%
gpt-4o similarity recursive	662	2	75.45%
gpt-4o similarity single prompt	8	0	49.22%
gpt-4o similarity single prompt	8	1	49.10%
gpt-4o similarity single prompt	8	2	48.53%
gpt-4o similarity single prompt	16	0	55.95%
gpt-4o similarity single prompt	16	1	55.27%
gpt-4o similarity single prompt	16	2	58.77%
gpt-4o similarity single prompt	32	0	57.99%
gpt-4o similarity single prompt	32	1	60.29%
gpt-4o similarity single prompt	32	2	60.22%
gpt-4o similarity single prompt	64	0	62.82%
gpt-4o similarity single prompt	64	1	64.24%
gpt-4o similarity single prompt	64	2	62.88%
gpt-4o similarity single prompt	128	0	65.81%
gpt-4o similarity single prompt	128	1	65.12%
gpt-4o similarity single prompt	128	2	64.34%
gpt-4o similarity single prompt	256	0	66.73%
gpt-4o similarity single prompt	256	1	67.54%
gpt-4o similarity single prompt	256	2	67.71%
gpt-4o similarity single prompt	512	0	67.85%
gpt-4o similarity single prompt	512	1	67.14%
gpt-4o similarity single prompt	512	2	68.82%
gpt-4o similarity single prompt	662	0	68.91%
gpt-4o similarity single prompt	662	1	68.30%
gpt-4o similarity single prompt	662	2	69.33%