Generative Large Language Models for Unimodal Key Information Extraction

Bachelor Thesis

February 7, 2023

Ivo Adam Schaper

schaper@cl.uni-heidelberg.de

Institut für Computerlinguistik

Ruprecht-Karls-Universität Heidelberg

Supervisor	Prof. Dr. Katja Markert
Reviewers	Prof. Dr. Katja Markert
	Prof. Dr. Michael Herweg

Abstract

We investigate pre-trained generative large language models (LLM) used “out-of-the-box” for key information extraction (KIE) from visually rich documents (VRD). Specifically, we benchmark Flan-T5, GPT-NeoX, and InstructGPT on the Kleister Charity and SROIE data sets where our best method reaches 70.5 and 78.0 in F1 respectively. Notably, our approach makes no use of layout or image information and considers only the textual representation of documents. To the best of our knowledge, applying pre-trained generative LLM in an auto-regressive way for KIE has not been explored and KIE itself has not been formally defined in the literature; our work closes these gaps. We propose a pipeline approach which can handle documents exceeding a given model's maximum input size which is in general constrained due to required computation for Transformer-based LLM scaling to the power of two with sequence length. Finally, in addition to exploring the relationship of parameter size and performance of the considered models, we explore the effects of temperature during sampling and of the number of examples during inference.

Zusammenfassung

Diese Arbeit untersucht die Anwendung von vortrainierten Sprachmodellen ("large language models") für die Extraktion von Schlüssel-Wert-Paaren ("Key Information Extraction, KIE") aus bildreichen Dokumenten ("visually rich documents"). Konkret werden die Modelle Flan-T5, GPT-NeoX und InstructGPT auf den "SROIE" und "Kleister Charity" Datensätzen evaluiert. Die beste Methode erreicht einen F1-Score von 70.5 beziehungsweise 78.0. Insbesondere ist anzumerken, dass dieser Ansatz keinerlei Informationen über das Layout oder das Bild des Dokuments, sondern nur textuelle Information, nutzt. Die Anwendung von vortrainierten generativen Modellen auf autoregressive Weise für KIE wurde bislang nicht erforscht und die Aufgabe selbst ist formal undefiniert; die Arbeit schließt diese Lücken. Außerdem stellt die Arbeit einen Pipeline-Ansatz vor, welcher in der Lage ist auch Dokumente, die die maximale Eingabelänge eines bestimmten Modells überschreiten, zu verarbeiten. Die Eingabelänge ist quadratisch mit der benötigten Rechenleistung verwandt und deshalb im Allgemeinen beschränkt. Dies ist ein allgemeines Problem von Transformer-basierten Sprachmodellen. Ferner untersucht die Arbeit die Beziehung zwischen der Parametergröße der betrachteten Modelle und der Performanz. Schließlich erforscht die Arbeit die Auswirkungen des Temperatur-Parameters bezüglich der "temperature sampling" Methode, sowie die Auswirkung der Anzahl der Beispiele während der Inferenz auf die Performanz.

Contents

List of Figures VI

List of Tables VIII

1 Introduction 1
- 1.1 Key Information Extraction (KIE) 1
- 1.2 Research Hypotheses 2
- 1.3 Methodology 4

2 Related Work 5
- 2.1 Information Extraction 5
- 2.2 Document Intelligence 6

3 Data Sets 7
- 3.1 Kleister Charity 7
- 3.2 SROIE 10

4 Methods 12
- 4.1 Baselines 12
- 4.2 Pipeline approach with LLM 16

5 Experiments 22
- 5.1 Design 22
- 5.2 Kleister Charity 22
- 5.3 SROIE 32

6 Discussion 35
- 6.1 Results on Kleister Charity 35
- 6.2 Results on SROIE 41
- 6.3 Limitations 43

7 Conclusion 44
7.1 Summary . . . 44
7.2 Future Work . . . 45
7.3 Acknowledgments . . . 46

A Appendix 47
A.1 Kleister Charity . . . 47
A.2 SROIE . . . 57

Bibliography 59

List of Figures

1 Illustration of our pipeline approach. 4

2 Example page out of a 64-page financial report of the Kleister Charity data set with marked values of some keys of interest. 8

3 Token distribution of documents in the Kleister Charity development and test sets. 9

4 Example receipt of the SROIE data set. 11

5 Token distribution of documents in the SROIE test set (μ=346, σ=93). . . . 11

6 Histograms of number of non-null values per key over the sub-documents on Kleister Charity development set for best setting of Flan-T5, GPT-NeoX, and InstructGPT. Error bars represent minimum and maximum over three runs. 28

7 Average number of collisions per key versus number of sub-documents on Kleister Charity development set for best setting of GPT-NeoX and InstructGPT. Shaded area represents 95% confidence interval of three runs. Note: in case of two sub-documents, every collision is by definition also a full collision. 29

8 Accuracy versus number of sub-documents on Kleister Charity development set for best setting of GPT-NeoX and InstructGPT. Shaded area of linear regression line represents 95% confidence interval estimated using bootstrap sampling. Error bars represent minimum and maximum over three runs. . . 30

9 Accuracy versus parameter size of Flan-T5, GPT-NeoX, and InstructGPT on Kleister Charity development set. 31

10 Average accuracy versus number of shots for best temperature setting of Flan-T5 (t=0.0), GPT-NeoX (t=1.0), and InstructGPT (t=0.1) on SROIE test set. 33

11 Accuracy versus parameter size for best temperature setting of Flan-T5 (t=0.0), GPT-NeoX (t=1.0), and InstructGPT (t=0.1) on SROIE test set. 34

12 Average number of collisions per key versus number of sub-documents on Kleister Charity development set for best setting of Flan-T5. Shaded area represents 95% confidence interval of three runs. Note: in case of two sub-documents, every collision is by definition also a full collision. 49

13 Accuracy versus number of sub-documents on Kleister Charity development set for best setting of Flan-T5. Shaded area of linear regression line represents 95% confidence interval estimated using bootstrap sampling. Error bars represent minimum and maximum over three runs. 49

14 Collision percentages by key on Kleister Charity development set for best setting of Flan-T5. Error bars represent minimum and maximum over three runs. 56

15 Collision percentages by key on Kleister Charity development set for best setting of GPT-NeoX. Error bars represent minimum and maximum over three runs. 56

16 Collision percentages by key on Kleister Charity development set for best setting of InstructGPT. Error bars represent minimum and maximum over three runs. 56

List of Tables

1 Definitions of terms used in research hypotheses. 3

2 Prevalence of keys of interest in the documents of the Kleister Charity data set according to the gold standard. Rounded to 3 decimal places. 10

3 Synonyms and expected named entity types of keys for Kleister Charity specific baseline. 13

4 Overview of considered LLM, their parameter size and the avenue of accessing them. 18

5 Hyperparameters of considered LLM during inference. 20

6 Overview of input constraints in terms of number of tokens of considered LLM. 20

7 Explanations of terms used in additional evaluation. 23

8 Ratio of finding a string match for the keys of interest with error_percentage=0.18 of general and Kleister Charity specific baselines on Kleister Charity. Note that values for “Address” keys for specific baseline are not retrieved by string match, but with approach detailed in section 4.1. No ranges are given, as the baselines are deterministic if run on the same day (cf. section 4.2). Average is micro-averaged. Results rounded to three decimal places. 24

9 Accuracy of general and Kleister Charity specific baselines on Kleister Charity. Baselines use window w=40 and error_percentage=0.18. No ranges are given, as the baselines are deterministic if run on the same day (cf. section 4.2). Average is micro-averaged over predicted key-value pairs. Results rounded to three decimal places. 24

10 “null” metrics of Flan-T5 on Kleister Charity development set. Ranges are sample standard deviations over three runs. Values are micro-averaged. Results rounded to three decimal places. 25

11 Accuracy of Flan-T5 on Kleister Charity. Best accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places. 25

12 “null” metrics of GPT-NeoX on Kleister Charity development set. Ranges are sample standard deviations over three runs. Values are micro-averaged. Results rounded to three decimal places. 26

13 Accuracy of GPT-NeoX on Kleister Charity. Best accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places. . . 26

14 “null” metrics of InstructGPT on Kleister Charity development set. Ranges are sample standard deviations over three runs. Values are micro-averaged. Results rounded to three decimal places. 27

15 Accuracy of InstructGPT on Kleister Charity. Best accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places. . . 27

16 Accuracy and F1 of Flan-T5 with best temperature setting (t=0.0) on SROIE test set. Best accuracy for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places. 32

17 Accuracy and F1 of GPT-NeoX with best temperature setting (t=1.0) on SROIE test set. Best accuracy for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places. 33

18 Accuracy and F1 of InstructGPT with best temperature setting (t=0.1) on SROIE test set. Best accuracy for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places. 33

19 Relative changes in “null”-F1 by key of InstructGPT (t=0.1) from zero- to one-shot setting on Kleister Charity development set. Relative changes with $p < 0.01$ as determined by unpaired t-test marked **bold** and rounded to one decimal place. 40

20 Gold values with corrections of example used for one-shot setting on Kleister Charity. 48

21 F1 scores on Kleister Charity with provided evaluation script of general and Kleister Charity specific baselines. Baselines use window w=40 and error_percentage=0.18. No ranges are given, as the baselines are deterministic if run on the same day (cf. section 4.2). Average is macro-averaged over keys. Results rounded to one decimal place. 50

22 F1 scores of Flan-T5 on Kleister Charity with provided evaluation script. Best scores on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is macro-averaged over keys. Results rounded to one decimal place. 50

23 F1 scores of GPT-NeoX on Kleister Charity with provided evaluation script. Best scores on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is macro-averaged over keys. Results rounded to one decimal place. 51

24 F1 scores of InstructGPT on Kleister Charity with provided evaluation script. Best scores on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is macro-averaged over keys. Results rounded to one decimal place. 51

25 Standard and enient ("oracle" considering *all* values predicted in sub-documents and perfect unification) accuracy of Flan-T5 on Kleister Charity. Best standard accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places. 52

26 Standard and lenient ("oracle" considering *all* values predicted in sub-documents and perfect unification) accuracy of GPT-NeoX on Kleister Charity. Best standard accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places. 53

27 Standard and lenient ("oracle" considering *all* values predicted in sub-documents and perfect unification) accuracy of InstructGPT on Kleister Charity. Best standard accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places. 54

28 Correlations of Flan-T5 on Kleister Charity development set. Values are micro-averaged. Results rounded to three decimal places. 54

29 Correlations of GPT-NeoX on Kleister Charity development set. Values are micro-averaged. Results rounded to three decimal places. 55

30 Correlations of InstructGPT on Kleister Charity development set. Values are micro-averaged. Results rounded to three decimal places. 55

31 Gold values of the the first example used for one- and two-shot settings on SROIE. 58

32 Gold values of the the second example used for the two-shot settings on SROIE. 58

1 Introduction

Generative large language models, in particular GPT-3 [Brown et al., 2020, Ouyang et al., 2022], have shown impressive capabilities in various down-stream natural language processing tasks including machine translation, question answering, text summarization, classification, and more. The purpose of this work is to investigate the under-explored potential of generative large language models for key information extraction (KIE), and to explore the effects of temperature when sampling, of providing examples during inference and of the parameter size of models on their performance on this task. Accompanying code of our work is available at `https://gitlab.cl.uni-heidelberg.de/schaper/unimodal-kie`.

The remainder of our work is structured as follows: In the following section, we explain the task of key information extraction and offer a formal definition. Next, we present our research hypotheses that guide this work and show our general approach of solving the task with generative LLM. In the following chapters, we start off by discussing related work in chapter 2. Then, we present the data sets we use in chapter 3. Next, we show in detail our specific methods in chapter 4 and the results of our experiments in chapter 5. Following this, we discuss our findings and their implications in chapter 6. Finally, we provide a conclusion of our work, including a look towards future research in chapter 7.

1.1 Key Information Extraction (KIE)

KIE deals with the problem of extracting relevant key-value pairs from visually rich documents, along with a specified set of desired keys (cf. task 3 in Huang et al. [2019]). For instance, given an invoice we may be interested in retrieving the values for the keys “Invoice Number”, “Seller”, and “Total”. Also, “[in contrast] to [question answering] problems, there is no question in natural language but rather a phrase or keyword [is provided]” [Borchmann et al., 2021], i.e., just the names of the keys are used. KIE is also different from named entity recognition (NER), where the task is to find and assign named entities to abstract categories like “Person” and “Location”. KIE has potentially wide applications in the business domain, where crucial information is often siloed in PDF documents and extracted manually.

We note that there are many slightly different definitions of KIE, e.g., per Zhu et al. [2022], “[t]he goal of Key Information Extraction (KIE) is to extract useful structured information from form-style documents such as invoices or medical reports”. In other related work, KIE is also referred to as “Property Extraction” [Borchmann et al., 2021] and “relationship extraction with key-value pairs at segment-level” [Li et al., 2021]. These slight discrepancies necessitate a more formal definition, which, to the best of our knowledge, does not exist at the time of writing. We provide such a formal definition below.

Formal Definition of KIE

Let d be a document, let $K_d = (k_1, ..., k_n)$ be the keys of interest and let $V_d^* = (v_1, ..., v_n)$ be the correct values that correspond with K_d. Also, $\forall v \in V_d^* : v \in U$, where U is the set that contains all strings, including the empty string ε which represents a value that is undefined. Combining K_d and V_d^*, the gold standard can equivalently be expressed as $P_d^* = (p_1, ..., p_n) = ((k_1, v_1), ..., (k_n, v_n))$, i.e., as the correct key-value pairs of d. KIE is then a task which we define as follows: Given a document d and keys of interest K_d, find their corresponding values V_d^*, or equivalently, find the correct key-value pairs P_d^*.

Another, less strict, assumption is that d is a semi-structured document (e.g., an invoice, a bill of ladings, or a receipt) as opposed to free-form text, e.g., a novel, a non-disclosure agreement, or a newspaper article. However, we do not formally define this here.

1.2 Research Hypotheses

The central focus of this work is to test the following hypothesis:

Pre-trained generative large language models can reach satisfactory performance on KIE from text-only inputs and without training.

We define these terms more precisely in Table 1. Additionally, to guide our investigation, we put forth three sub-hypotheses:

a. As the correct solution is well-defined, sampling at higher temperatures[1] negatively impacts performance due to increased diversity.

b. Performance improves when providing few-shot examples during inference.[2]

c. For Transformer [Vaswani et al., 2017] language models, Kaplan et al. [2020] observe that "performance has a power-law relationship with [parameter size] when not bottlenecked by [compute or data set size]". As the models considered in this work are all based on the Transformer architecture, we hypothesize that we can observe that same relationship.

Term	**Definition**
Generative (large) language model (LLM)	A probability distribution over sequences of words[3]. "Generative" in this case specifically describes the fact that the probability of a token given all preceding tokens is modeled. There is no formal definition of what "large" means in this context; the term was popularized in Brants et al. [2007] and is commonly used to refer to language models that have been pre-trained on vast amounts of data from the internet in an unsupervised or semi-supervised way.
Satisfactory performance	80% of the values for given and findable keys are correctly found. We do not consider whether the other values are wrong or missing entirely.
Text-only inputs	Models are only given the textual representation of a given document; no additional image or layout information is given.
Without training	Models' weights are not updated with respect to the data which we consider for evaluation.
Few-shot examples	Refers to a small number of examples or demonstrations, typically less than 100, provided to the model at **inference time** as conditioning. We use this term in the framework of "in-context learning" as used in Brown et al. [2020].

Table 1: Definitions of terms used in research hypotheses.

[1] First introduced in Ackley et al. [1985], applied to text generation in Ficler and Goldberg [2017], Fan et al. [2018], Caccia et al. [2018], among others.

[2] Brown et al. [2020] observe this behavior on a wide range of tasks with GPT-3.

[3] cf. chapter 4 in Jurafsky and Martin [2008]

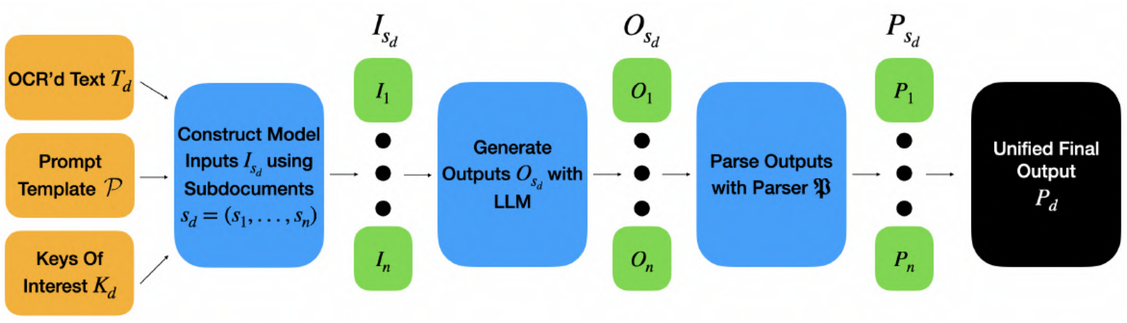

OCR'd Text T_d
Prompt Template $\mathcal{P}$
Keys Of Interest K_d
Construct Model Inputs I_{s_d} using Subdocuments $s_d = (s_1, \dots, s_n)$
I_{s_d}
I_1
I_n
Generate Outputs O_{s_d} with LLM
O_{s_d}
O_1
O_n
Parse Outputs with Parser $\mathfrak{P}$
P_{s_d}
P_1
P_n
Unified Final Output P_d

Figure 1: Illustration of our pipeline approach.

1.3 Methodology

This section aims to explain our general approach of using generative LLM for KIE. Our approach can be thought of as a pipeline with distinct steps which we illustrate in Figure 1. We describe the steps in more detail below. We elaborate on the specifics of this approach in chapter 4.

We first construct model input $I_d = T_d \circ \mathcal{P}$, where T_d is the textual representation of the document d as determined by Optical Character Recognition (OCR), $\mathcal{P}$ is a prompt template which contains the keys of interest K_d and $\circ$ is concatenation.

Should I_d exceed the model's maximum input size[4], we split d into sub-documents $s_d = (s_1, ..., s_n)$ and concatenate each sub-document with $\mathcal{P}$ which yields n model inputs I_{s_d} that do not exceed the model's maximum input size. For simplicity, we will henceforth leave out the index d and it should thus be assumed that we always refer to a particular document d unless otherwise stated.

Given I_s, the model generates outputs O_s which we then parse into key-value pairs P_s with a parser $\mathfrak{P}$. Note that there may be different values for the same key in the parsed key-value pairs of different sub-documents i and j. Formally, this can be expressed as $P_i, P_j \in P_s, i \neq j, \neg\forall k \in K : P_i(k) = P_j(k) = v_{k_i} = v_{k_j}$, where $P(k)$ yields the corresponding value of k in P. This necessitates some form of *unification* of sub-document key-value pairs P_s. Unification is the last step in our pipeline and yields a final prediction P.

[4] All models considered in this work are based on the Transformer architecture introduced in Vaswani et al. [2017]. The computation required for the attention mechanism with this architecture is $\mathcal{O}(L^2)$ where L is the length of the sequence, which quickly becomes infeasible [Kitaev et al., 2020]. We show the exact input limitations of the considered models in Table 6.

2 Related Work

2.1 Information Extraction

As the name implies, Key Information Extraction (KIE) is part of the broader field of Information Extraction (IE) whose "goal [...] is to make the text's [implicit] semantic structure explicit" [Grishman, 2015]. This is a markedly broad goal with a plethora of sub-tasks, including but not limited to named entity tagging, named entity recognition, named entity linking, coreference resolution, and relation extraction.

Notable work in relation extraction, which from the above-mentioned sub-tasks is most similar to KIE, was conducted by Zhang et al. [2017] who introduce a large-scale relation extraction data set, TACRED, which "is one of the largest and most widely used sentence-level relation extraction data sets" [Stoica et al., 2021]. Additionally, Stoica et al. [2021] find issues with TACRED and perform a re-annotation of the entire TACRED data set which they call Re-TACRED.

In 2019, Huang et al. [2019] introduce SROIE, i.e. KIE on annotated receipts as part of the ICDAR 2019 Robust Reading Competition. In general, publicly available data sets for KIE are limited and include Kleister Charity/NDA [Stanisławek et al., 2021], CORD [Park et al., 2019], FUNSD [Jaume et al., 2019], XFUND [Xu et al., 2022], EPHOIE [Wang et al., 2021], WildReceipt [Sun et al., 2021], NIST SFRS [Garris, 2017], EATEN [Guo et al., 2019], and Project DeepForm[1].

Information Extraction, and specifically KIE, is of great interest in the business domain which is reflected by much of the above-mentioned work being conducted by researchers at private companies. Related to this, we note that some work in KIE is evaluated on publicly unavailable data sets [Baviskar et al., 2021, Zhao et al., 2019, Schuster et al., 2013, Palm et al., 2017], potentially in order to retain a competitive advantage. Finally, KIE can be classified as a task under the broader term of *Document Intelligence* which we discuss in the next section.

[1] https://wandb.ai/deepform/political-ad-extraction/benchmark

2.2 Document Intelligence

According to Cui et al. [2021], “Document Intelligence is a relatively new research topic that refers to the techniques for automatically reading, understanding, and analyzing business documents.” which started in the early 1990s. Layout analysis, table recognition, and KIE are notable sub-tasks in this domain. It is generally considered to be a very challenging task “due to diversity of layouts and formats, low-quality scanned document images, and [...] complexity of template structure” and is motivated by reducing time-consuming manual extraction [Cui et al., 2021].

We give a brief overview of approaches in Document AI and refer to Cui et al. [2021] for a comprehensive survey of models, tasks and benchmark data sets. Historically, models for document analysis were rule-based and required templates that encoded the layout information of a given document [Cui et al., 2021]. Crucially, this led to these models being very sensitive to changes in layout.

With the advent of deep neural networks and the pre-train/fine-tune paradigm, “accuracy of Document AI tasks [greatly improved]” [Cui et al., 2021], specifically by incorporating visual features into the pre-training stage of Transformer-based [Vaswani et al., 2017] multi-modal models. Notably, Xu et al. [2020] propose general document pre-training with LayoutLM. Since then, the “typical framework is to treat document images as a pixel grid and add text features to the visual feature map” Cui et al. [2021] for the task of key information extraction (also called visual information extraction) and Stanisławek et al. [2021] claim that “spatial information is essential for properly understanding these kinds of [visually rich] documents”.

Finally, we briefly discuss work most similar to ours. Ni et al. [2022] frame the task of relation extraction (RE) and classification (RC) as a sequence-to-sequence generation task and present a novel generative model. Similarly, Alt et al. [2019] extend the original OpenAI GPT Transformer for RE. For RC, Han et al. [2022] explore prompt tuning for pre-trained language models. Finally, Josifoski et al. [2021] consider the task of closed information extraction where sets of triplets are to be extracted from unstructured text under constraints from a Knowledge Base schema and auto-regressively generate these triplets in textual form. To the best of our knowledge, our work is the first to tackle the task of key information extraction from visually rich documents using text-only input and generative large language models in an auto-regressive way.

3 Data Sets

3.1 Kleister Charity

Introduced in Stanisławek et al. [2021], *Kleister Charity* is a data set of 2788 financial reports from English and Welsh charity organizations, with a combined total of 61,643 pages. We show an example page of a report in Figure 2. The reports are "a mixture of born-digital and (mostly) scanned" [Stanisławek et al., 2021] PDF files, which mandates the use of *optical character recognition* (OCR) to extract the textual information, which is additionally included with the data set. Specifically, the text for each report is supplied as extracted by three different OCR systems, namely, Microsoft Azure Computer Vision API, Tesseract [Smith, 2007] and Amazon Textract. Stanisławek et al. [2021] also include results of several baseline systems which use the text as extracted by Microsoft Azure CV. To ensure best possible comparisons with our results, we exclusively use the supplied text as extracted by Microsoft Azure CV, rather than extracting the text ourselves.

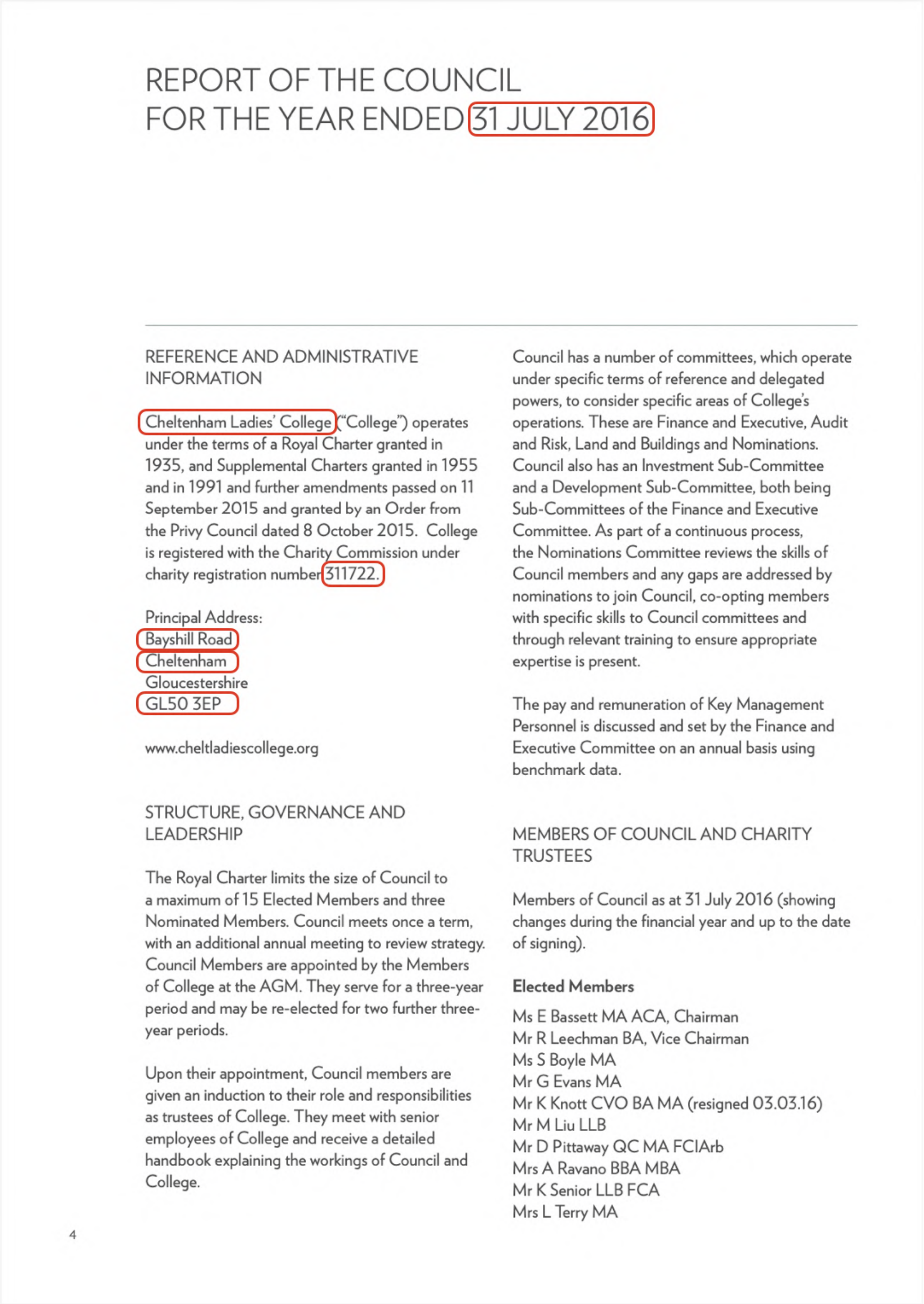
The data set is split into train, development, and test sets with 1729, 440, and 609 documents respectively. The language used in all documents is English. As we do not train or otherwise fine-tune the weights of any model, we discard the training set, and focus only on the development and test sets.

The task can be expressed in terms of our definition of KIE in section 1.1. In this case, the keys of interest K are the same for all documents d, which is the textual representation of the financial reports. Table 2 shows the keys of interest and their prevalence, i.e., the fraction with a designated gold value that is not empty.

We also show the distribution of tokens[1] in the Kleister Charity data set on both development and test set in Figure 3. We can observe that the distributions are similar and that both sets contain outliers. The longest document in the data set has 294,640 tokens, which corresponds to roughly 235,712 words when assuming that one token corresponds to 4 characters of text and that a word consists of 5 characters on average. For reference, Harry Potter and the Sorcerer's Stone has 76,444 words[2].

[1] As determined by the tokenizer of GPT-2. We use this tokenizer for all token counts in this work.

[2] According to `https://wordcounter.io/blog/how-many-words-are-in-harry-potter`.



REPORT OF THE COUNCIL
FOR THE YEAR ENDED 31 JULY 2016

REFERENCE AND ADMINISTRATIVE INFORMATION

Cheltenham Ladies' College ("College") operates under the terms of a Royal Charter granted in 1935, and Supplemental Charters granted in 1955 and in 1991 and further amendments passed on 11 September 2015 and granted by an Order from the Privy Council dated 8 October 2015. College is registered with the Charity Commission under charity registration number 311722.

Principal Address:
Bayshill Road
Cheltenham
Gloucestershire
GL50 3EP

www.cheltladiescollege.org

STRUCTURE, GOVERNANCE AND LEADERSHIP

The Royal Charter limits the size of Council to a maximum of 15 Elected Members and three Nominated Members. Council meets once a term, with an additional annual meeting to review strategy. Council Members are appointed by the Members of College at the AGM. They serve for a three-year period and may be re-elected for two further three-year periods.

Upon their appointment, Council members are given an induction to their role and responsibilities as trustees of College. They meet with senior employees of College and receive a detailed handbook explaining the workings of Council and College.

Council has a number of committees, which operate under specific terms of reference and delegated powers, to consider specific areas of College's operations. These are Finance and Executive, Audit and Risk, Land and Buildings and Nominations. Council also has an Investment Sub-Committee and a Development Sub-Committee, both being Sub-Committees of the Finance and Executive Committee. As part of a continuous process, the Nominations Committee reviews the skills of Council members and any gaps are addressed by nominations to join Council, co-opting members with specific skills to Council committees and through relevant training to ensure appropriate expertise is present.

The pay and remuneration of Key Management Personnel is discussed and set by the Finance and Executive Committee on an annual basis using benchmark data.

MEMBERS OF COUNCIL AND CHARITY TRUSTEES

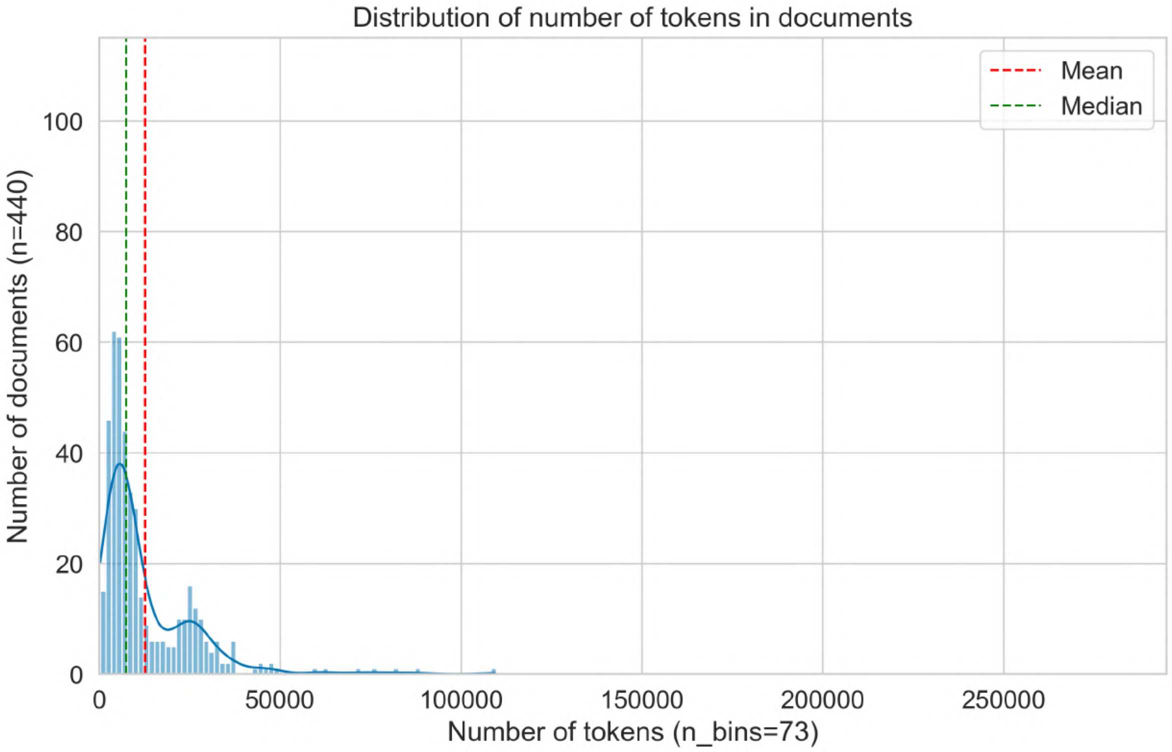
Members of Council as at 31 July 2016 (showing changes during the financial year and up to the date of signing).

Elected Members

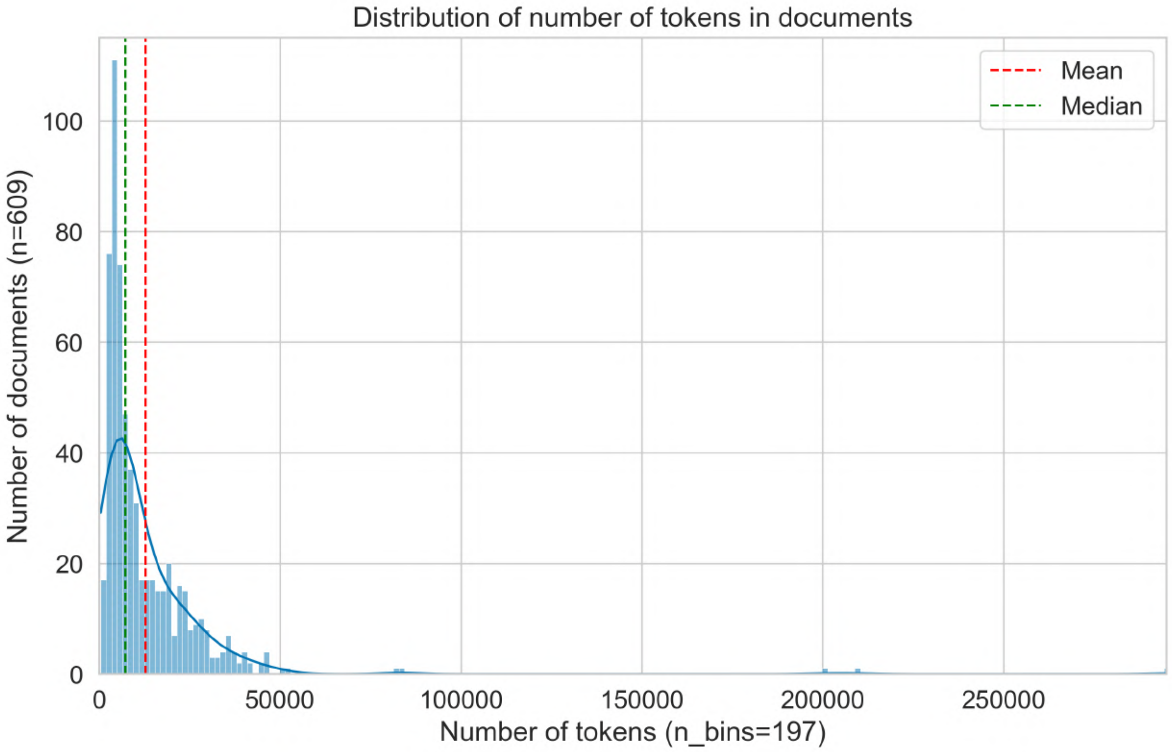
Ms E Bassett MA ACA, Chairman
Mr R Leechman BA, Vice Chairman
Ms S Boyle MA
Mr G Evans MA
Mr K Knott CVO BA MA (resigned 03.03.16)
Mr M Liu LLB
Mr D Pittaway QC MA FCIArb
Mrs A Ravano BBA MBA
Mr K Senior LLB FCA
Mrs L Terry MA

4

Figure 2: Example page out of a 64-page financial report of the Kleister Charity data set with marked values of some keys of interest.



(a) Token distribution in development set (μ=12633, σ=13253).



(b) Token distribution in test set (μ=12746, σ=19274).

Figure 3: Token distribution of documents in the Kleister Charity development and test sets.

Key	**Prevalence**	
	Development	Test
Address (post town)	0.959	0.956
Address (post code)	0.968	0.970
Address (street)	0.886	0.920
Charity Name	1.000	1.000
Charity Number	0.993	0.989
Annual Income	0.986	0.979
Period End Date	1.000	1.000
Annual Spending	0.986	0.970

Table 2: Prevalence of keys of interest in the documents of the Kleister Charity data set according to the gold standard. Rounded to 3 decimal places.

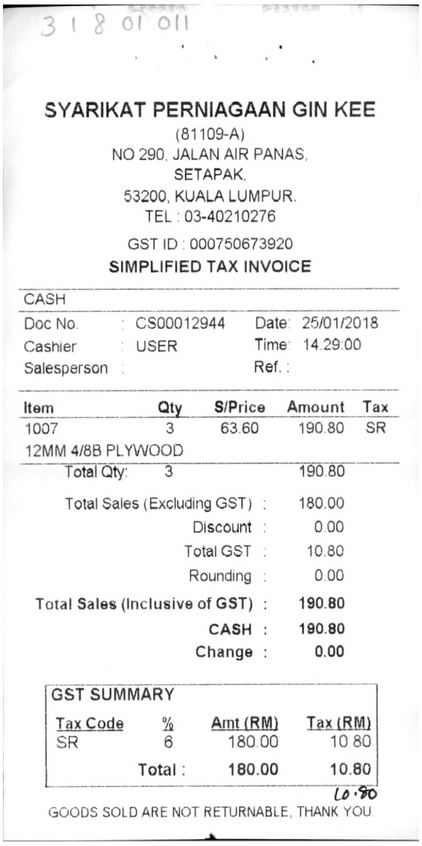
3.2 SROIE

The SROIE data set is a collection of 987 scanned receipt images[3] and annotations created for the ICDAR2019 competition [Huang et al., 2019] which comprises three tasks. We consider in this work task 3, KIE from scanned receipts, with a test set of 347 images, which include OCR and bounding box annotations. We discard all other sets, as we do not train or otherwise fine-tune the weights of any model. For our model input, we use the provided OCR annotation to ensure best possible comparisons with other results. Finally, we discard the bounding box information, as we do not use any layout information for our methods. We show an example receipt of the SROIE data set in Figure 4.

Analogously to the task of KIE on Kleister Charity, this task can be expressed in terms of our formal definition in section 1.1. Here, the keys of interest K are “Company name”, “Date of Receipt”, “Address of Company”, and “Total”. Contrary to Kleister Charity and according to the gold standard of SROIE, all keys have an associated value in all receipts and d is the OCR’d text of a receipt instead of a financial report.

While the task on Kleister Charity and SROIE is identical, we note that the text of an average financial report from Kleister Charity is approximately 36 times longer than that of a receipt from SROIE. We show the distribution of the number of tokens in the documents of the SROIE data set in Figure 5.

[3] Huang et al. [2019] state that 1000 receipt images are available, however, on the official homepage of the competition, accessible at `https://rrc.cvc.uab.es/?ch=13`, only 987 images are available for download.



3 1 8 01 011

SYARIKAT PERNIAGAAN GIN KEE
(81109-A)
NO 290, JALAN AIR PANAS,
SETAPAK,
53200, KUALA LUMPUR.
TEL : 03-40210276

GST ID : 000750673920
SIMPLIFIED TAX INVOICE

CASH

Doc No.	:	CS00012944	Date:	25/01/2018
Cashier	:	USER	Time:	14:29:00
Salesperson	:		Ref. :	

Item	Qty	S/Price	Amount	Tax
1007	3	63.60	190.80	SR
12MM 4/8B PLYWOOD				
Total Qty:	3		190.80	

Total Sales (Excluding GST)	:	180.00
Discount	:	0.00
Total GST	:	10.80
Rounding	:	0.00
Total Sales (Inclusive of GST)	:	**190.80**
CASH	:	**190.80**
Change	:	**0.00**

GST SUMMARY

Tax Code	%	Amt (RM)	Tax (RM)
SR	6	180.00	10.80
Total :		180.00	10.80

10.80

GOODS SOLD ARE NOT RETURNABLE, THANK YOU

Figure 4: Example receipt of the SROIE data set.

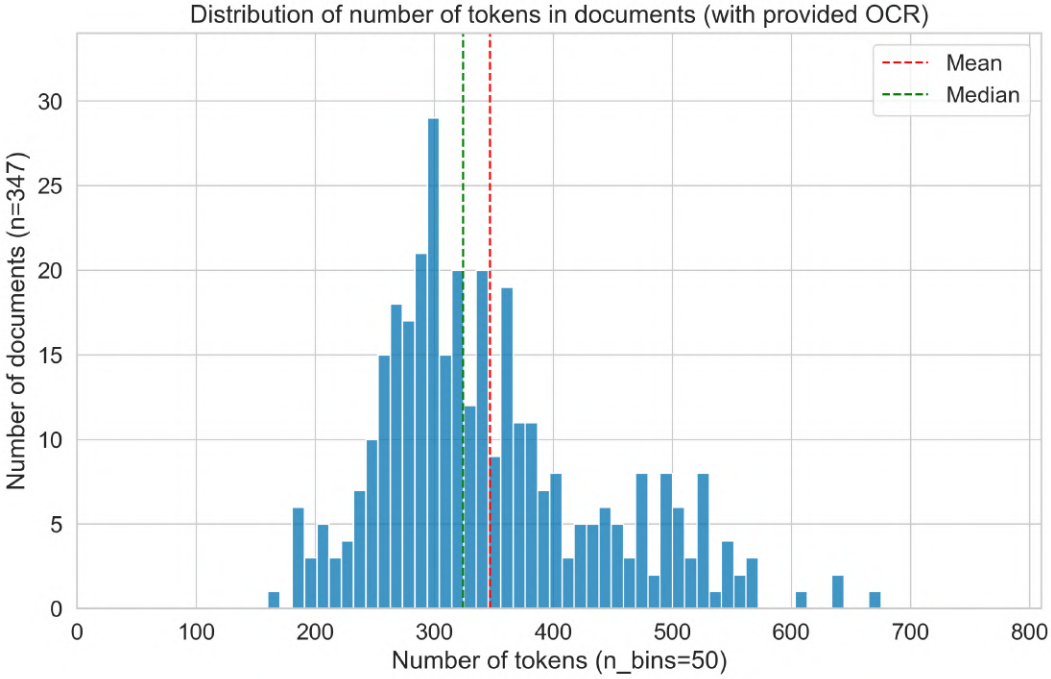


Figure 5: Token distribution of documents in the SROIE test set (μ=346, σ=93).

4 Methods

4.1 Baselines

In order to have a common point of reference, we first describe a general baseline applicable to any KIE task. Furthermore, we present a specific baseline, which is tailored to the task on the Kleister Charity data set and uses knowledge of the keys. Both baselines are rule-based.

General Baseline

With our general baseline, we intentionally refrain from sophisticated methods and instead focus on simplicity and interpretability. The main idea is to search for the key in the text of the document with some degree of fuzziness that is proportional to the length of the key. We define fuzziness in terms of the Levenshtein distance considered in Levenshtein et al. [1966]. For instance, searching for the value of the "Period End Date" key, the maximum allowed Levenshtein distance for a match is the character length of the key multiplied with an error percentage, e.g., for an error percentage of 0.18, the maximum allowed distance is $15 \cdot 0.18 = 2.7 \approx 3$. Additionally, our search for the best match is case-insensitive. Given that a best match is found, we then select the first named entity after it, within some character window w, as the value of the key. If there is no named entity within the window, we consider the key to not have an associated value. We describe this approach more precisely in Algorithm 1.

A glaring shortcoming of this approach is the inability to find key-value pairs when a given key is not explicitly present in the document. Additionally, this approach is not very robust as the desired key may be expressed in form of an abbreviation, e.g., "Charity No." instead of "Charity Number", which escapes the fuzziness constraint even with relatively high error percentages. Moreover, this approach heavily relies on the quality of the named entity recognition, which in turn depends on the OCR quality, essentially creating a propagation chain of errors. Finally, due to the lost layout information, another source of error arises when the value for a key appears only outside the character window w.

Algorithm 1 General Baseline

```
function GET_BEST_MATCH_SPAN($input$, $key$, $error\_pct$)
    $key\_length \leftarrow |key|$
    $max\_errors \leftarrow round(key\_length * error\_pct)$
    $match\_span \leftarrow regex.search(k, input, e < max\_errors)$
    if $match\_span$ then
        return $match\_span$
    end if
end function
function PREDICT($input$, $keys$)
    $ner\_tags \leftarrow get\_ner\_tags(input)$
    $output \leftarrow \emptyset$
    for $key$ in $keys$ do
        $match\_span \leftarrow get\_best\_match\_span(input, key, error\_pct)$
        if $match\_span$ then
            $first\_entity \leftarrow get\_first\_entity(match\_span, ner\_tags, window)$
            if $first\_entity$ then
                $output[key] \leftarrow first\_entity$
            end if
        end if
    end for
    return $output$
end function
```

Key	Synonyms	Named Entity
Address (post town)	N/A	GPE, LOCATION
Address (post code)	N/A	N/A
Address (street)	N/A	FACILITY
Charity Name	N/A	ORGANIZATION, NORP
Charity Number	“Charity Registration No”, “Charity No”	CARDINAL
Annual Income	“Income”, “Total Income”	MONEY, CARDINAL
Period End Date	“Period End”, “Year Ended”	DATE
Annual Spending	“Spending”, “Total Spending”, “Expenditure”	MONEY, CARDINAL

Table 3: Synonyms and expected named entity types of keys for Kleister Charity specific baseline.

Kleister Charity Specific Baseline

In addition to the general baseline, we implement a baseline that is specific to the Kleister Charity data set. The general approach is the same as with the general baseline, however, there are some key differences.

First, we use synonyms for the keys to increase robustness, e.g., instead of just searching for "Charity Number", we additionally search for "Charity Registration No" and "Charity No". The choice of these synonyms is motivated by considering documents in the development set manually. Second, for the "Address" keys, we do not use the approach from the general baseline. Instead, we first search for the post code with a regular expression and then use a given match as the anchor point for finding the street and the post town utilizing knowledge on how addresses are usually formatted. Third, for all other keys, we employ a type validation scoring system which rewards a given entity for a key if it fits an expected named entity type. For instance, the entity for "Period End Date" should be a `DATE`.

We show the synonyms and expected named entity types for the specific baseline in Table 3. For more information on the named entity types see chapter 2.6 in Weischedel et al. [2011]. Additionally, we describe our implementation of the specific baseline more precisely in Algorithm 2. The algorithm uses the same `get_best_match_span()` shown in Algorithm 1. Note that for the specific baseline algorithm, we assume the address keys to be in order like so: "Address (post code)", "Address (street)", "Address (post town)". The specific retrieval functions are left out for brevity, but we give a short description of how they work here:

- `find_post_code()`: Uses a regular expression[1] to return the first match of a UK post code in the input.
- `find_street()`: As UK addresses generally contain the street before the post code, this function returns the first line within 60 characters prior to the post code that contains "street", "avenue", "road", or "place".
- `find_post_town()`: Returns the first named that is a `GPE` or `LOCATION` within a window. Given that both, a post code and a street, have been found, the window is set to be in between those. If only a post code has been found, the window is set to 60 characters prior to the post code and the post code itself.

[1] Taken from `https://stackoverflow.com/a/51885364/11694746`.

Algorithm 2 Kleister Charity Specific Baseline

```
function PREDICT($input$, $keys$)
    $ner\_tags \leftarrow get\_ner\_tags(input)$, $output \leftarrow \emptyset$
    for $key$ in $keys$ do
        if $key =$ “Address (post code)” then
            $post\_code$, $post\_code\_idx \leftarrow find\_post\_code(input)$
            $output[key] \leftarrow post\_code$
        else if $key =$ “Address (street)” then
            if $output \neq \emptyset$ then          ▷ Can only find the street if we found a post code
                $street$, $street\_idx \leftarrow find\_street(input, post\_code\_idx)$
                $output[key] \leftarrow street$
            end if
        else if $key =$ “Address (post town)” then
            if $output \neq \emptyset$ then
                $post\_town \leftarrow find\_post\_town(input, street\_idx, post\_code\_idx)$
                $output[key] \leftarrow post\_town$
            end if
        else
            $candidate\_key\_matches \leftarrow \emptyset$, $candidate\_entities \leftarrow \emptyset$
            for $synonym\_key$ in $synonyms[key]$ do
                $match\_span \leftarrow get\_best\_match\_span(input, synonym\_key, error\_pct)$
                if $match\_span$ then
                    $first\_entity \leftarrow get\_first\_entity(match\_span, ner\_tags, window)$
                    if $first\_entity$ then
                        $candidate\_key\_matches.append(match\_span)$
                        $candidate\_entities.append(first\_entity)$
                    end if
                end if
            end for
            $best\_entity \leftarrow \emptyset$, $best\_entity\_score \leftarrow 0$
            for $candidate\_entity, candidate\_key\_match$ do
                $entity\_score \leftarrow 0$
                if $type(candidate\_entity) \in expected\_type[key]$ then
                    $entity\_score \leftarrow entity\_score + 3$
                end if
                $entity\_score \leftarrow entity\_score - levenshtein(candidate\_key\_match, key)$
                if $entity\_score > best\_entity\_score$ then
                    $best\_entity \leftarrow entity$, $best\_entity\_score \leftarrow entity\_score$
                end if
            end for
            $output[key] \leftarrow best\_entity$
        end if
    end for
    return $output$
end function
```

4.2 Pipeline approach with LLM

In this section, we delve deeper into the methodology of our pipeline approach outlined in section 1.3. We begin by explaining how the model input is constructed, including the attachment of the prompt and examples. We then provide details on the LLM we utilize and show how generations are parsed and how key-value pairs are unified when working with sub-documents in order to not exceed a given model's maximum input size.

Construction of Model Input

As the first step in the pipeline, a model input I is constructed by concatenating the textual representation of a document d, T, with a prompt $\mathcal{P}$ which contains the keys of interest K. More specifically, in all of our experiments, we use the following fixed prompt $\mathcal{P}$:

```
Extract <K>, "<|stop key|>" from the document above. If you can’t find a
key-value pair in the document set the value to "null".
Key: Value
<FIRST_KEY>:
```

To account for the possibility that the correct value for a given key is the empty string ε, as defined formally in section 1.1, i.e., the value cannot be determined from the document, we include the sentence “If you can’t find a key-value pair in the document set the value to 'null'.” in the prompt.

With this prompt, we also aim guide the model to generate the key-value pairs in a structured format, as in:

```
<KEY_1>: <VALUE_1>
<KEY_2>: <VALUE_2>
```

and so on. This structure allows for easy parsing of the generated output. The inclusion of `<FIRST_KEY>:` in the prompt, in addition to the abstract `Key: Value`, aims to further emphasize the desired structure for the key-value pairs.

The use of the `<|stop key|>` in the prompt and as a stop sequence in a given LLM aims to signal the end of the generated text and prevent the model from continuing to produce additional tokens.

Attaching few-shot examples

As per sub-hypothesis b, we want to test whether few-shot examples improve performance. Specifically, we attach shots by prepending them to the model input I_d, like so:

```
Find below the OCR'd text of an example document:
###
<SHOT_1>
###
Find below the OCR'd text of an example document:
###
<SHOT_2>
###
...
Find below the OCR'd text of a *new* document:
###
<I_d>
```

A single shot consists of the textual representation T of a document, concatenated with $\mathcal{P}$; this is equivalent to how model inputs I are constructed. Additionally, it contains the gold key-value pairs in the desired format, including the `<|stop key|>` with no value.

For the Kleister Charity data set, we select the document in the training set with the least amount of tokens and its gold standard for our one-shot experiments. We motivate this by the fact that a random document from the training set would far exceed the maximum input size of any of our models. The gold standard has incorrect information for some keys of that document, which we manually correct. We show the text of the shot and the corresponding and corrected gold standard in appendix A.1.

In contrast, for the experiments on the SROIE data set, we select two random receipts from the training set and their corresponding gold standard. We show the text of the shots and their corresponding gold standard in appendix A.2.

Sub-documents

Given the issue of I exceeding a model's maximum input size, we split d into sub-documents $s = (s_1, ..., s_n)$, with some overlap. The overlap is motivated by the possibility of splitting a document right where important information for the correct value of some key, e.g., the value itself or, more generally, the context of a key-value pair may be, which would then be split across two sub-documents. The overlap between sub-documents is arbitrarily set to 20 tokens for all experiments. Formally, this also implies that s is *not* a partition of d. The model inputs on sub-document level are constructed in the same way as previously explained and we construct the sub-document level inputs I_s to be as close to the maximum input size that a given model can handle.

LLM used for Generation

We now describe in detail the models, configurations, and limitations we use for generations based on I, i.e., the second step of the pipeline that produces outputs O. The three model families we consider are Flan-T5 [Chung et al., 2022], GPT-NeoX [Black et al., 2022], and InstructGPT [Ouyang et al., 2022]. We give an overview of the specific models that we study, together with their parameter sizes and how we access the models, in Table 4. The specific models cover a wide range of parameter sizes, which we motivate with sub-hypothesis c, which states that we expect to observe a power-law relationship between performance and parameter size of Transformer models. Note that we continue to refer to the specific models by their associated model families for better legibility.

Model Family	**Specific Model**	**Parameter Size**	**Accessed via**
Flan-T5	`flan-t5-xl`	3 billion	Hugging Face Inference API
GPT-NeoX	`gpt-neox-20b`	20 billion	TextSynth API
InstructGPT	`text-davinci-003`	175 billion	OpenAI API

Table 4: Overview of considered LLM, their parameter size and the avenue of accessing them.

In order to minimize other factors, we leave all hyperparameters, except for the temperature during sampling (cf. sub-hypothesis a), fixed and give their values in Table 5. Any other hyperparameters that are specific to the models are set to their default values[2].

[2] For Flan-T5 see `https://huggingface.co/docs/transformers/v4.25.1/en/main_classes/text_generation#transformers.GenerationMixin.generate.inputs`, for GPT-NeoX see `https://textsynth.com/documentation.html#completions` and for InstructGPT see `https://beta.openai.com/docs/api-reference/completions/create`.

We also specify the limitations of all models regarding their maximum context window and maximum input tokens in Table 6. The context window describes the limit for both input and output tokens of a LLM. We compute from this the maximum input window which leaves ample space for 160 tokens for the output for all LLM. We choose to limit the maximum input tokens of Flan-T5 to 1792 as we were unable to reliably prevent out-of-memory errors with more tokens.

Flan-T5

Flan-T5 [Chung et al., 2022] is a LLM that has been finetuned on a collection of data sets phrased as instructions. The models are based on T5 [Raffel et al., 2020] and considerably improve upon their performance in a variety of benchmarks. T5, in turn, is based on the encoder-decoder Transformer [Vaswani et al., 2017] architecture and pre-trained on a span-corruption objective "loosely inspired by SpanBERT [Joshi et al., 2020]".

GPT-NeoX

Motivated by the fact that LLM "are almost universally the protected intellectual property of large organizations, and are gated behind a commercial API, available only upon request, or not available for outsider use at all", Black et al. [2022] introduce GPT-NeoX-20B, a LLM whose architecture largely follows that of GPT-3 [Brown et al., 2020].

InstructGPT

Similarly to the Flan models introduced in [Chung et al., 2022], InstructGPT [Ouyang et al., 2022] models are also instruction-finetuned, however, they use a method called reinforcement learning from human feedback (RLHF) [Christiano et al., 2017, Stiennon et al., 2020] and are based on GPT-3 [Brown et al., 2020]. GPT-3 is pre-trained by causal language modelling (i.e., a MLE objective of predicting the next token given all previous tokens) and is also based on the Transformer [Vaswani et al., 2017] architecture.

The model considered in this research, `text-davinci-003`, is an improvement over `text-davinci-002`, an InstructGPT model according to the OpenAI model index[3]. We note that we were unable to find information on how this improvement is achieved and how it relates to the InstructGPT 175B model presented in Chung et al. [2022].

Hyperparameter	**Value**
max_generated_tokens	256
temperature	(0, 0.1, 1)
top_p	0.9
top_k[4]	40
presence_penalty[5]	0
frequency_penalty	0
stop_sequence	`\n<|stop key|>`

Table 5: Hyperparameters of considered LLM during inference.

Model	**max_context_window**	**max_input_tokens**
Flan-T5	Only limited by memory[6]	1792
GPT-NeoX	2048	1792
InstructGPT	4000	3840

Table 6: Overview of input constraints in terms of number of tokens of considered LLM.

Parsing the Model Output

Given some output O, the parser $\mathfrak{P}$ turns the raw generations of a LLM, into key-value pairs P. The parser assumes that the value for any given key is in between what immediately follows it and before the next key (independent of line breaks). If the next key is not in the model output the parser searches for the next key after that one and so on until a key if found or the end of the model output is reaches. This value is then cleaned up by removing line breaks and any leading or trailing whitespace.

[3] Accessible at `https://beta.openai.com/docs/model-index-for-researchers`.

[4] This parameter is not supported by OpenAI for the InstructGPT models. We were unable to find information on whether this value is used and, if so, to which value it is set.

[5] Note that presence_penalty and frequency_penalty are not supported by Hugging Face for Flan-T5. There is, however, a repetition_penalty, which is disabled in all of our experiments.

[6] This is because the model that Flan-T5 is based on, T5 [Raffel et al., 2020], uses relative positional embeddings.

If a value for a key consists only of whitespace, i.e., the next key follows without any ASCII characters in between, or the value starts with “null”, the value for the key is set to the empty string ε, i.e., we have determined that the value was not found in the given (sub-)document. Additionally, for evaluations on the Kleister Charity data set, we further parse the generated values for some keys. For the “Period End Date” key, we parse it to ISO8601 format as the gold standard is not robust to other date formats. Importantly, this makes our results non-deterministic even for our rule-based baselines and greedy sampling (temperature equal to zero) with LLM because the date parser in our immplementation sometimes resolves values relative to the day the program is run. For instance, if the value for the “Period End Date” is (mistakenly) retrieved as “three”, this is resolved to the date the program is run with the month set to March (the third month).

For the “Annual Income” and “Annual Spending” keys, the gold standard mandates that values are to be provided in British pounds without currency symbols and with a full stop as the separator symbol like so: `123456.78`. We accordingly parse retrieved values for these keys by first ignoring any non-numeric characters (except for the full stop), which effectively also removes currency symbols, then by removing leading zeros, and finally by attaching `.00` to the retrieved value if it does not already contain a full stop.

Unification

Given the usage of sub-documents, the parser yields one key-value pair for each sub-document and key. Since the retrieved value for a key may differ between sub-documents, we must unify these candidate values to arrive at a final key-value pair for the whole document d. We do this by simply taking the most frequently occurring value of a particular key in the sub-documents, ignoring any values that represent the empty string value ε. For instance, given we used five sub-documents and the retrieved values for the key “Charity Number” in those sub-documents are “12345”, “54321”, “null’, “null” and “12345”, we set the final value of that key to “12345”. If there is no majority value, we set the value to the first non-empty string value in the order of the sub-documents.

5 Experiments

5.1 Design

Our experiments are designed with the aim of obtaining results that provide insight into the impact of variables that are of interest for our sub-hypotheses, i.e., temperature, few-shot examples, and parameter size of LLM on our method. Due to the high financial cost associated with running our experiments, we cannot try every possible combination of the aforementioned variables. Instead, we first focus on the development set of Kleister Charity and start by examining the impact of temperature for each considered LLM in the zero-shot setting, i.e., no examples are given as part of the input. The best performing temperature for each LLM is then used in the one-shot setting. Finally, the best performing combination of temperature and shot setting for each LLM on the development set is used on the test set of Kleister Charity.

For the SROIE data set, we use the best performing temperature on the Kleister Charity development set in the zero-shot setting and conduct experiments focused on studying the impact of few-shot examples on performance. This is done by directly testing the zero-shot, one-shot, and two-shot performance of each LLM on the test set.

To ensure robust results, all experiments are run three times and results are reported as averages over three runs with sample standard deviations, unless otherwise stated.

5.2 Kleister Charity

We do not present our results on the development and test sets using the evaluation script provided by Stanisławek et al. [2021] because we suspect possibly incorrect evaluation results in edge cases, slightly incorrect calculations of micro-averages, and unclear reporting of macro- or micro-averaged F1 scores. We report these issues in the provided repository on GitHub[1].

[1] `https://github.com/applicaai/kleister-charity/issues/6` and `https://github.com/applicaai/kleister-charity/issues/5#issuecomment-1368842929`

Results with the provided evaluation script are shown in appendix A.1 for better comparison to the results in Stanisławek et al. [2021].

In addition to these issues, the evaluation script is limited with respect to our specific research hypothesis (cf. definition of "satisfactory performance") which prompts us to carry out our own evaluation to better asses our methods and investigate the effect of sub-documents. We explain terms related to our additional evaluation, including our definition of correctness, in Table 7.

For the baselines, we report the ratio of finding matches for the keys of interest in the documents in Table 8 and accuracy in Table 9 for the best setting of window w and error percentage we found on the development set. For Flan-T5, GPT-NeoX and InstructGPT, we report scores pertaining to retrieving gold "null" values in tables 10, 12, and 14 in addition to the accuracy in tables 11, 13, and 15 respectively. Furthermore, we include histograms of the number of non-null values per key for all three LLM in their best setting in Figure 6. Moreover, for GPT-NeoX and InstructGPT, we include plots depicting the average number of collisions per key in Figure 7 and accuracy in Figure 8 with respect to the number of sub-documents. We do not include plots for Flan-T5 for the latter two in this chapter but in appendix A.1, in addition to auxiliary plots for models in their respective best settings. Finally, we illustrate the accuracy with respect to the parameter sizes of the considered LLM in Figure 9.

Term	**Explanation**
Correctness	Case-insensitive string match with minor normalization detailed in appendix A.1.
Accuracy	Ratio of correctly retrieved key-value pairs over all gold key-value pairs whose value is not "null". This is directly in line with our definition of "satisfactory performance".
Trivial Unification	All non-null candidate values for a given key are the same.
Collision	At least two non-null candidate values for a given key are different.
Full Collision	All non-null candidate values for a given key are different.
"null" Metrics	Consider only gold "null" with respect to a (whole) document.

Table 7: Explanations of terms used in additional evaluation.

Key	**Development Set**		**Test Set**	
	General	**Specific**	**General**	**Specific**
Address (post town)	0.000	-	0.000	-
Address (post code)	0.002	-	0.007	-
Address (street)	0.005	-	0.003	-
Charity Name	0.225	0.225	0.167	0.167
Charity Number	0.750	0.977	0.714	0.977
Annual Income	0.048	0.977	0.020	0.984
Period End Date	0.107	0.975	0.064	0.984
Annual Spending	0.009	0.941	0.008	0.946
Average	0.143	0.819	0.123	0.811

Table 8: Ratio of finding a string match for the keys of interest with error_percentage=0.18 of general and Kleister Charity specific baselines on Kleister Charity. Note that values for "Address" keys for specific baseline are not retrieved by string match, but with approach detailed in section 4.1. No ranges are given, as the baselines are deterministic if run on the same day (cf. section 4.2). Average is micro-averaged. Results rounded to three decimal places.

Key	**Development Set**		**Test Set**	
	General	**Specific**	**General**	**Specific**
Address (post town)	0.000	0.112	0.000	0.103
Address (post code)	0.000	0.594	0.000	0.638
Address (street)	0.000	0.195	0.000	0.268
Charity Name	0.036	0.036	0.028	0.028
Charity Number	0.563	0.739	0.583	0.771
Annual Income	0.000	0.094	0.000	0.134
Period End Date	0.000	0.064	0.000	0.089
Annual Spending	0.000	0.018	0.000	0.014
Average	0.077	**0.232**	0.078	**0.256**

Table 9: Accuracy of general and Kleister Charity specific baselines on Kleister Charity. Baselines use window w=40 and error_percentage=0.18. No ranges are given, as the baselines are deterministic if run on the same day (cf. section 4.2). Average is micro-averaged over predicted key-value pairs. Results rounded to three decimal places.

Key	**Setting**			
		Zero-Shot		**One-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=0.0**
Precision	0.027±0.000	0.027±0.000	0.027±0.000	0.026±0.000
Recall	0.928±0.013	0.931±0.005	0.900±0.013	0.804±0.000
F1	0.052±0.001	0.052±0.000	**0.053±0.001**	0.050±0.000

Table 10: "null" metrics of Flan-T5 on Kleister Charity development set. Ranges are sample standard deviations over three runs. Values are micro-averaged. Results rounded to three decimal places.

Key	**Development Set**				**Test Set**
		Zero-Shot		**One-Shot**	**One-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=0.0**	**t=0.0**
Address (post town)	0.156±0.000	0.152±0.004	0.136±0.015	**0.265±0.000**	0.287±0.000
Address (post code)	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
Address (street)	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
Charity Name	0.000±0.000	0.000±0.000	0.000±0.000	**0.016±0.000**	0.015±0.000
Charity Number	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
Annual Income	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
Period End Date	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
Annual Spending	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
Average	0.019±0.000	0.019±0.000	0.017±0.002	**0.035±0.000**	0.037±0.000

Table 11: Accuracy of Flan-T5 on Kleister Charity. Best accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places.

Key	**Setting**			
		Zero-Shot		**One-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=1.0**
Precision	0.028±0.000	0.028±0.000	0.040±0.010	0.013±0.003
Recall	0.876±0.000	0.876±0.005	0.299±0.059	0.113±0.022
F1	0.055±0.000	0.055±0.000	**0.071±0.017**	0.023±0.005

Table 12: “null” metrics of GPT-NeoX on Kleister Charity development set. Ranges are sample standard deviations over three runs. Values are micro-averaged. Results rounded to three decimal places.

Key	**Development Set**				**Test Set**
		Zero-Shot		**One-Shot**	**Zero-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=1.0**	**t=1.0**
Address (post town)	0.088±0.000	0.058±0.050	**0.155±0.010**	0.002±0.003	0.191±0.015
Address (post code)	**0.031±0.000**	0.020±0.018	0.030±0.007	0.002±0.003	0.058±0.016
Address (street)	**0.031±0.000**	0.020±0.017	0.027±0.009	0.001±0.001	0.051±0.018
Charity Name	0.143±0.000	0.144±0.007	**0.365±0.017**	0.007±0.002	0.444±0.003
Charity Number	0.096±0.000	0.060±0.052	**0.153±0.024**	0.038±0.009	0.236±0.014
Annual Income	**0.012±0.000**	0.007±0.006	0.005±0.001	0.001±0.001	0.012±0.007
Period End Date	0.134±0.000	0.095±0.082	**0.345±0.030**	0.127±0.012	0.447±0.004
Annual Spending	0.000±0.000	0.001±0.001	**0.002±0.001**	0.001±0.001	0.002±0.002
Average	0.067±0.000	0.051±0.023	**0.138±0.009**	0.023±0.000	0.183±0.005

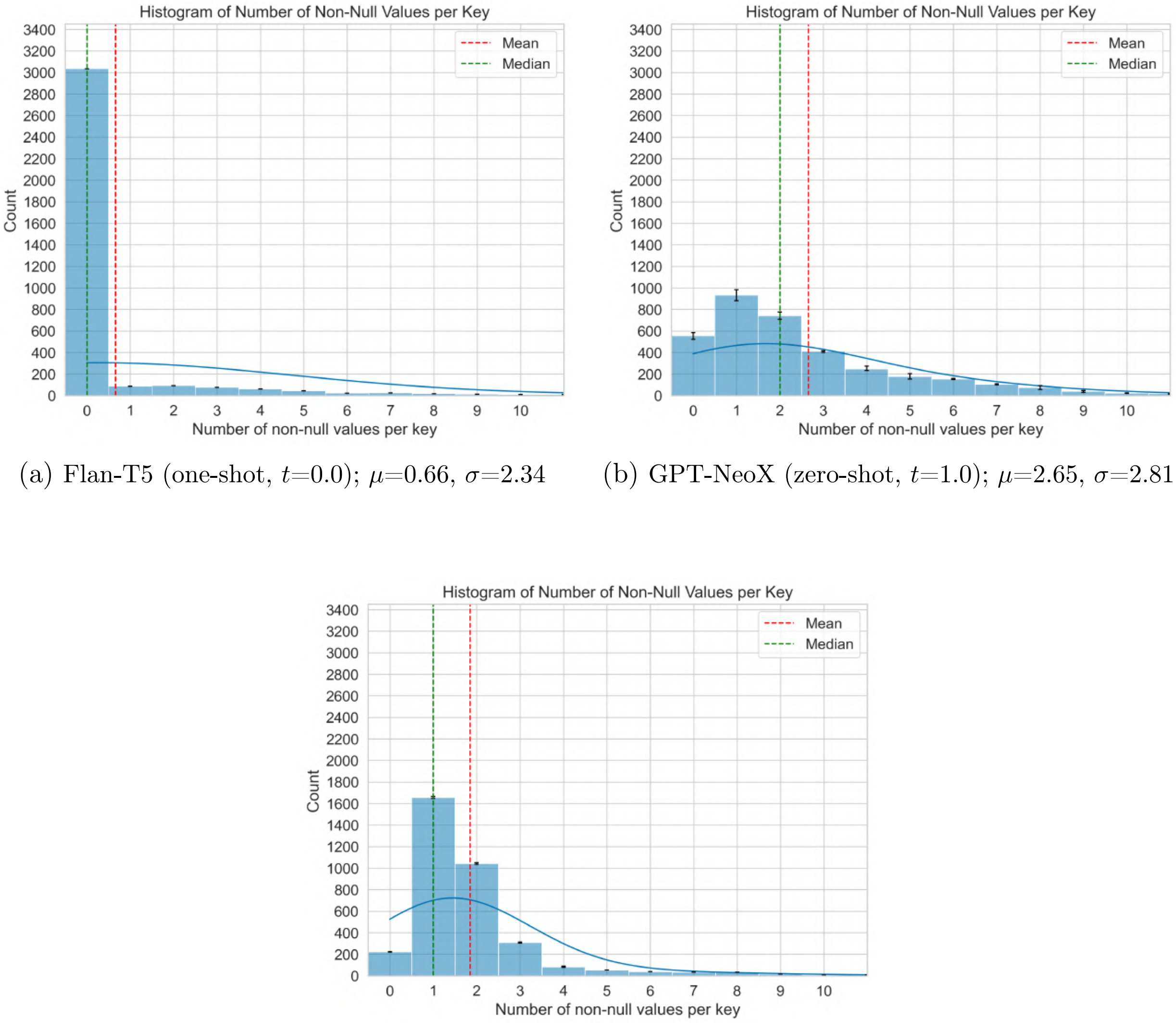
Table 13: Accuracy of GPT-NeoX on Kleister Charity. Best accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places.

Key	**Setting**			
		Zero-Shot		**One-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=0.1**
Precision	0.348±0.004	0.352±0.005	0.103±0.010	0.162±0.006
Recall	0.299±0.000	0.299±0.000	0.419±0.032	0.378±0.010
F1	0.322±0.002	**0.323±0.002**	0.165±0.015	0.226±0.007

Table 14: "null" metrics of InstructGPT on Kleister Charity development set. Ranges are sample standard deviations over three runs. Values are micro-averaged. Results rounded to three decimal places.

Key	**Development Set**				**Test Set**
		Zero-Shot		**One-Shot**	**One-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=0.1**	**t=0.1**
Address (post town)	0.791±0.001	0.790±0.001	0.598±0.017	**0.806±0.001**	0.812±0.002
Address (post code)	**0.673±0.001**	0.671±0.003	0.449±0.009	0.644±0.003	0.701±0.003
Address (street)	0.368±0.001	0.368±0.004	0.285±0.016	**0.491±0.001**	0.550±0.003
Charity Name	0.725±0.000	0.723±0.001	0.694±0.018	**0.767±0.003**	0.846±0.000
Charity Number	0.826±0.002	0.863±0.003	0.735±0.005	**0.938±0.000**	0.908±0.001
Annual Income	**0.452±0.004**	0.446±0.009	0.212±0.016	0.386±0.001	0.484±0.004
Period End Date	0.957±0.000	0.956±0.001	0.845±0.006	**0.959±0.002**	0.958±0.001
Annual Spending	**0.480±0.001**	0.469±0.007	0.194±0.031	0.443±0.007	0.495±0.003
Average	0.663±0.000	0.665±0.001	0.505±0.008	**0.682±0.000**	0.722±0.000

Table 15: Accuracy of InstructGPT on Kleister Charity. Best accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places.

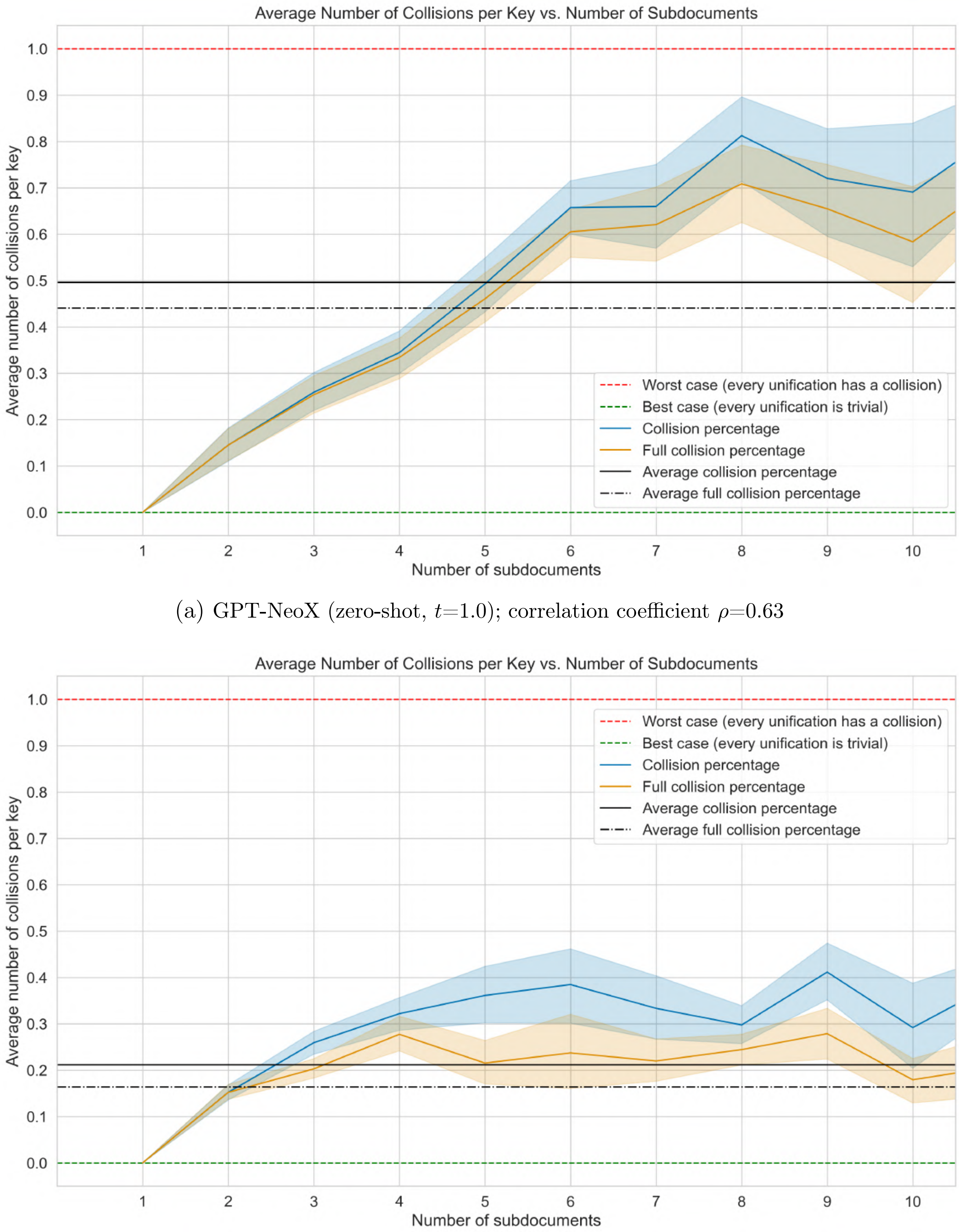


(a) Flan-T5 (one-shot, t=0.0); μ=0.66, σ=2.34

(b) GPT-NeoX (zero-shot, t=1.0); μ=2.65, σ=2.81

(c) InstructGPT (one-shot, t=0.1); μ=1.85, σ=1.69

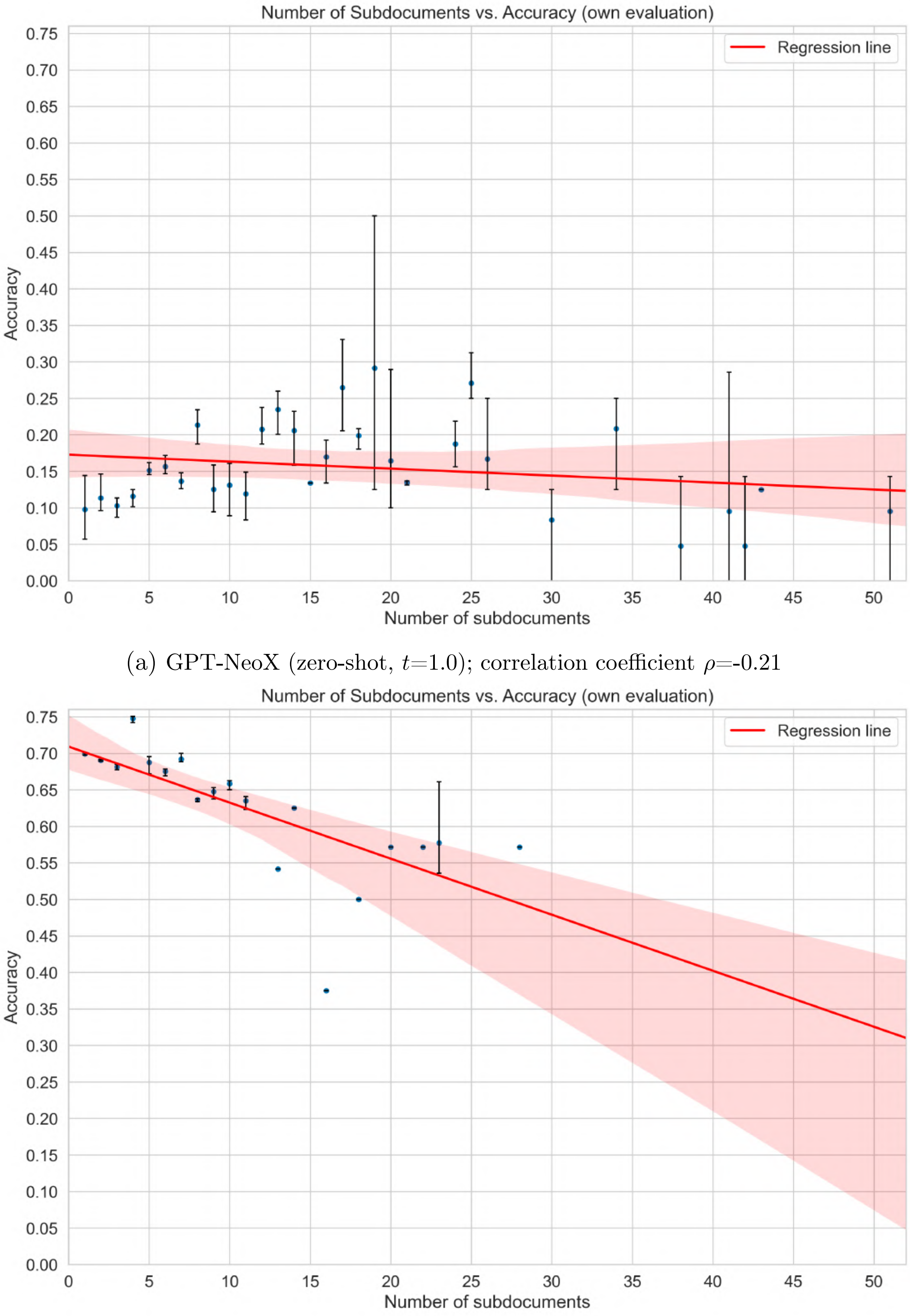
Figure 6: Histograms of number of non-null values per key over the sub-documents on Kleister Charity development set for best setting of Flan-T5, GPT-NeoX, and InstructGPT. Error bars represent minimum and maximum over three runs.



(a) GPT-NeoX (zero-shot, t=1.0); correlation coefficient ρ=0.63

(b) InstructGPT (one-shot, t=0.1); correlation coefficient ρ=0.42

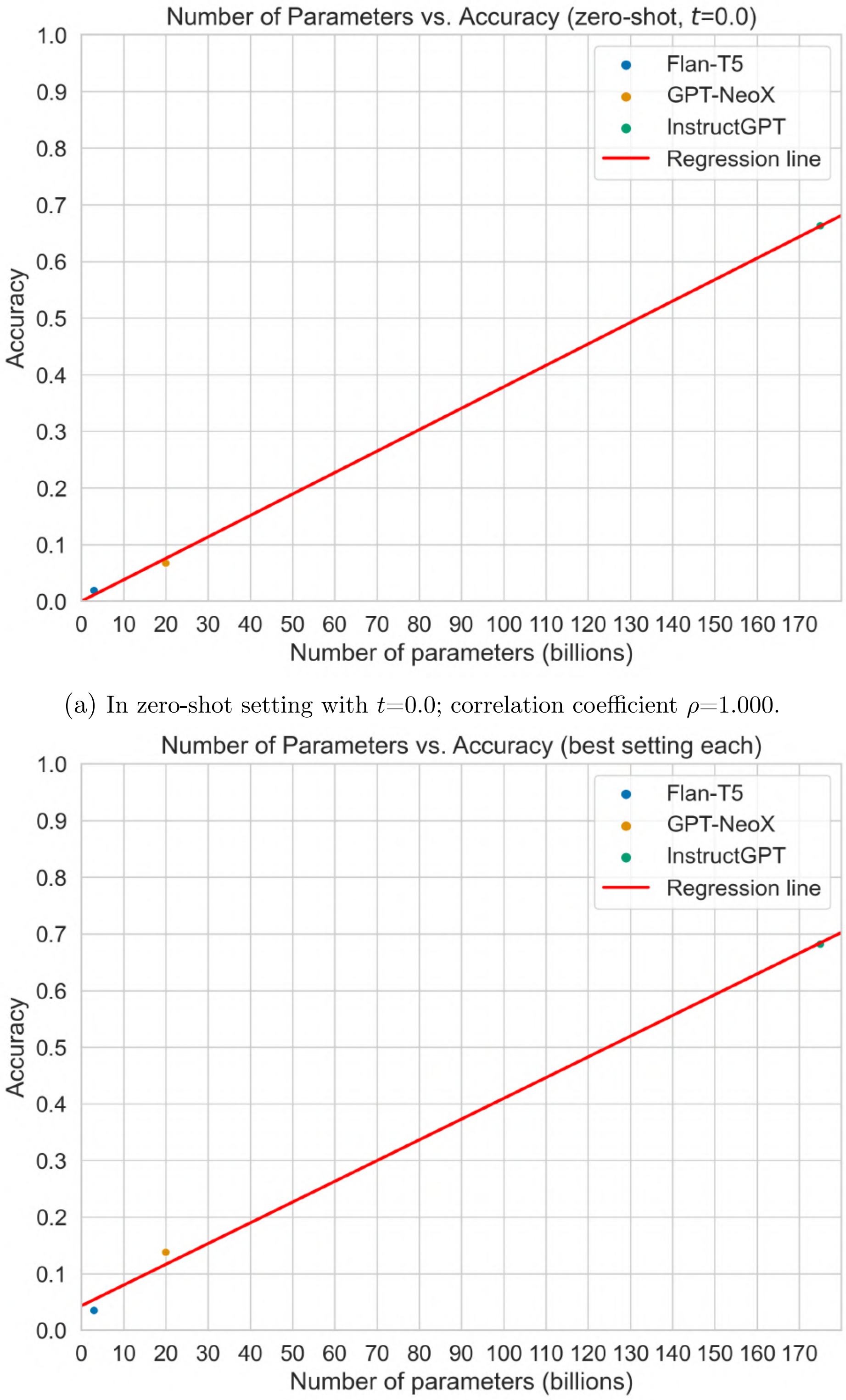
Figure 7: Average number of collisions per key versus number of sub-documents on Kleister Charity development set for best setting of GPT-NeoX and InstructGPT. Shaded area represents 95% confidence interval of three runs. Note: in case of two sub-documents, every collision is by definition also a full collision.



(a) GPT-NeoX (zero-shot, t=1.0); correlation coefficient ρ=-0.21

(b) InstructGPT (one-shot, t=0.1); correlation coefficient ρ=-0.69

Figure 8: Accuracy versus number of sub-documents on Kleister Charity development set for best setting of GPT-NeoX and InstructGPT. Shaded area of linear regression line represents 95% confidence interval estimated using bootstrap sampling. Error bars represent minimum and maximum over three runs.



(a) In zero-shot setting with t=0.0; correlation coefficient ρ=1.000.

(b) For best setting of each LLM; correlation coefficient ρ=0.998.

Figure 9: Accuracy versus parameter size of Flan-T5, GPT-NeoX, and InstructGPT on Kleister Charity development set.

5.3 SROIE

There are two significant differences of the SROIE task when compared to Kleister Charity that we illustrated previously in chapter 3 which we reiterate here to motivate the choice of presented results. First, all keys in all documents have a designated gold value (no empty values). Second, as the documents are much shorter, only three out of the total 347 documents have to be split into sub-documents, and only in the two-shot setting with Flan-T5 and GPT-NeoX. Hence, we leave out any "null" and sub-document related metrics.

As the provided evaluation script by Huang et al. [2019] does not include metrics on a per-key basis, we compute the accuracy according to our definition on a per-key basis, as well as on average, ourselves. We note that our definition of accuracy is identical to the recall as defined and computed by the provided script, however, we continue to refer to this metric as "accuracy" for consistency with our usage of this term on Kleister Charity. We present results, including the officially calculated F1 score used in Huang et al. [2019] to rank submissions, for Flan-T5 in Table 16, for GPT-NeoX in Table 17, and for InstructGPT in Table 18. Additionally we illustrate the accuracy of the three LLM with respect to the number of shots in Figure 10 and with respect to their parameter sizes in Figure 11. Finally, for reference, we note that our baseline scores **0.037** and **0.061** in accuracy and F1 respectively.

Key	**Zero-Shot**	**One-Shot**	**Two-Shot**
Company Name	0.026±0.000	**0.340±0.000**	0.164±0.000
Date of Receipt	0.000±0.000	0.000±0.000	0.000±0.000
Address of Company	0.000±0.000	0.000±0.000	**0.003±0.000**
Total	0.000±0.000	0.000±0.000	0.000±0.000
Average	0.006±0.000	**0.085±0.000**	0.042±0.000
F1	0.013±0.000	**0.135±0.000**	0.065±0.000

Table 16: Accuracy and F1 of Flan-T5 with best temperature setting (t=0.0) on SROIE test set. Best accuracy for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places.

Key	**Zero-Shot**	**One-Shot**	**Two-Shot**
Company Name	0.015±0.009	**0.539±0.030**	0.519±0.027
Date of Receipt	0.082±0.015	0.454±0.004	**0.472±0.012**
Address of Company	0.000±0.000	0.218±0.007	**0.231±0.023**
Total	0.035±0.005	0.471±0.031	**0.474±0.031**
Average	0.033±0.005	0.421±0.008	**0.424±0.021**
F1	0.037±0.005	0.427±0.008	**0.431±0.021**

Table 17: Accuracy and F1 of GPT-NeoX with best temperature setting (t=1.0) on SROIE test set. Best accuracy for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places.

Key	**Zero-Shot**	**One-Shot**	**Two-Shot**
Company Name	0.479±0.006	**0.929±0.002**	0.921±0.002
Date of Receipt	0.584±0.002	0.618±0.009	**0.906±0.004**
Address of Company	0.333±0.010	0.417±0.002	**0.551±0.007**
Total	**0.845±0.006**	0.793±0.006	0.741±0.000
Average	0.561±0.005	0.689±0.003	**0.780±0.001**
F1	0.561±0.005	0.689±0.003	**0.780±0.001**

Table 18: Accuracy and F1 of InstructGPT with best temperature setting (t=0.1) on SROIE test set. Best accuracy for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places.

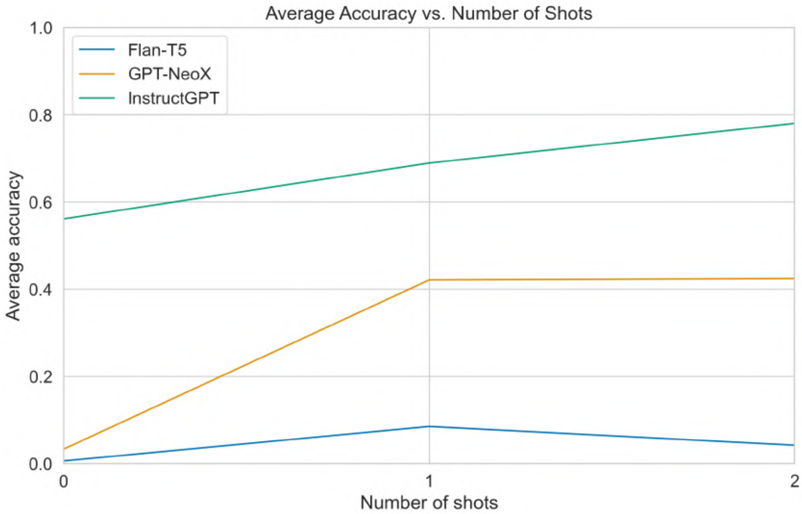
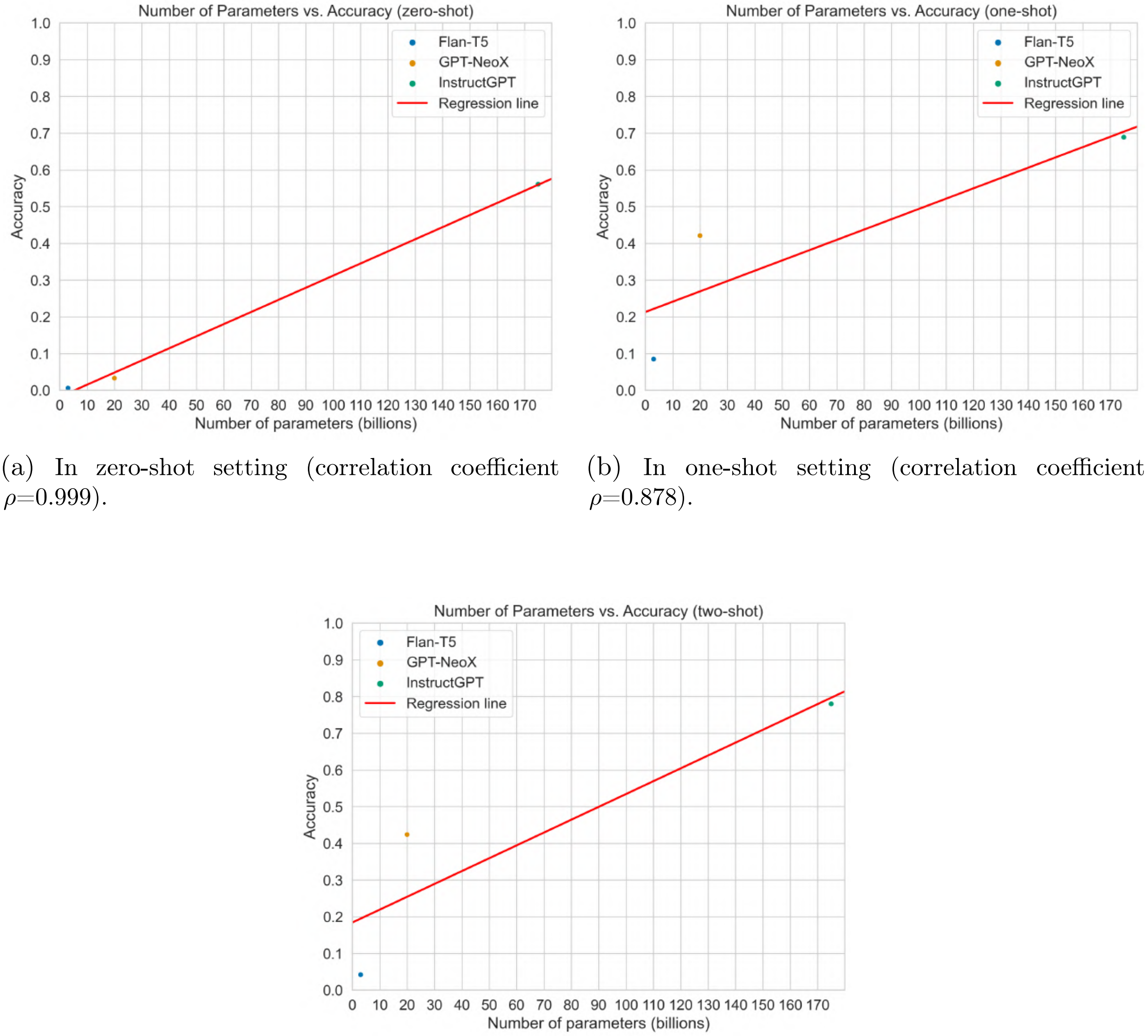


Figure 10: Average accuracy versus number of shots for best temperature setting of Flan-T5 (t=0.0), GPT-NeoX (t=1.0), and InstructGPT (t=0.1) on SROIE test set.



(a) In zero-shot setting (correlation coefficient ρ=0.999).

(b) In one-shot setting (correlation coefficient ρ=0.878).

(c) In two-shot setting (correlation coefficient ρ=0.899).

Figure 11: Accuracy versus parameter size for best temperature setting of Flan-T5 (t=0.0), GPT-NeoX (t=1.0), and InstructGPT (t=0.1) on SROIE test set.

6 Discussion

6.1 Results on Kleister Charity

Baselines

On the development set, the general baseline retrieves matches for the keys of interest in 14.3% of cases and in 22.8% of cases when excluding the address keys for comparison with the specific baseline which uses the approach detailed in section 4.1 for those keys. In contrast, the specific baseline retrieves matches for those keys in 81.9% of cases, which is the direct cause of using synonyms (cf. Table 3) which we initially motivated to increase robustness. Crucially, when not finding a string match for a given key, the value for it is said to be undefined according to the baselines. By considering this and the average prevalence of keys according to the gold standard as shown in Table 2, which is 97.2% for the development set, we can assert that the general baseline cannot exceed the accuracy of the specific baseline and that using synonyms does not hinder accuracy. The above statements hold true for the test set as well with slightly different ratios of finding matches.

The general baseline retrieves only values for “Charity Name” and “Charity Number” correctly at least once on the development and test set, further exemplifying the frail process of matching against keys explicitly in the text. The specific baseline finds more matches for keys by relative 259.2% and 317.1%, on the development and test set, respectively, which translates to an average relative improvement of 58.8% and 69.6% in accuracy for these keys. Retrieving values for the address keys with the special approach detailed in section 4.1 proves to be effective, improving accuracy from zero with the general baseline to an average of 0.300 and 0.336 for these keys on the development and test set respectively.

Finally, we repeat the average accuracy which we aim to beat with our more sophisticated pipeline approach as set by the specific baseline with **0.232** and **0.256** on the development and test set respectively.

Pipeline Approach

In this section, we discuss results with the pipeline approach and draw comparisons between the LLM used, which we subsequently discuss in more detail per model along with qualitative evaluations and error analyses. Note that we consider only results on the development set in this section.

Most notably, the average accuracy of Flan-T5 and GPT-NeoX is worse than the specific baseline in all settings, whereas the average accuracy of any setting of InstructGPT is *better* than it with $p < 0.01$ as determined by a one-sample t test rejecting the null hypothesis that the mean of any given three runs is equal to the fixed value of the specific baseline. This implies that the major factor with respect to accuracy is not the temperature during sampling or the zero- versus one-shot setting, but the LLM itself. Crucially, this does not leave the parameter size itself as the deciding factor, as the LLM differ in many other variables, e.g., their maximum input size, their training objective, and instruction-finetuning. Notwithstanding, Figure 9 depicts the accuracy versus the parameter size of the three LLM. We observe very high correlation coefficients of ρ=1.000, and ρ=0.998 in the zero-shot setting with temperature t=0.0, and with the best setting for each LLM, respectively. In essence, for the considered LLM, the relationship of parameter size and accuracy is almost perfectly linear, contrary to sub-hypothesis c, which suggests a power-law relationship.

Furthermore, there is no consistent negative relationship between accuracy and temperature across models as stipulated by sub-hypothesis a. In particular, the best performing temperature in the zero-shot setting is different for all three models and GPT-NeoX with t=1.0 (which we hypothesize would perform the worst out of the three) has approximately double the accuracy of t=0.1 and t=0.0 with $p < 0.01$ as determined by an unpaired t-test rejecting the null hypothesis that the means of the three runs of the respective temperature settings are equal.

While the one-shot setting does improve accuracy (cf. sub-hypothesis b) of Flan-T5 and InstructGPT with their respective best temperatures (with $p < 0.01$ as determined in the same way as previously), it only does so marginally by absolute 0.016, and 0.019 points. For GPT-NeoX, providing a single example hurts accuracy considerably and is even worse than the worst temperature setting in the zero-shot setting (with $p < 0.01$). We discuss the reason for this in a following section about GPT-NeoX specifically.

Finally, note that sampling with temperature t=0.0 is equivalent to greedy and accordingly deterministic sampling which explains the sample standard deviation being 0 in those cases. Conspicuosly, InstructGPT is the only model that does not reliably produce deterministic results (consider the non-zero sample standard deviations); we discuss this in section 6.3.

Flan-T5

Above all, Flan-T5 generally fails to recognize the desired structure of generating one key-value pair per line. Instead, Flan-T5 almost always generates just a single line for the first key "Address (post town)" (cf. accuracy by key in Table 11) and then generates an end-of-sequence token which terminates said generation. In this case, the parser considers the values for all other keys to be "null". Consider Figure 6, which shows the distribution of number of non-null values per key for Flan-T5 in the one-shot setting and temperature t=0.0, which is the best setting ($p < 0.01$) of Flan-T5 in both, "null" metrics and overall accuracy. Clearly, Flan-T5 almost always predicts "null" for the reason stated above. This issue is not alleviated when including an example, instead the precision and F1 for gold "null" values drop ($p < 0.01$) marginally (cf. Table 10).

The accuracy of Flan-T5, with this substantial issue, is nonetheless highly negatively correlated (ρ=-0.995) with the temperature during sampling. Also, the one-shot accuracy improves by 84.2% ($p < 0.01$) compared to the zero-shot accuracy with the same temperature. Notably, the one-shot accuracy for "Address (post town)" is higher than the accuracy of GPT-NeoX in any setting ($p < 0.01$) for this key.

GPT-NeoX

In contrast to sub-hypothesis a, which stipulates a negative relationship between temperature and performance, the accuracy of GPT-NeoX with temperature $t = 1.0$ is a relative improvement of 105.97% ($p < 0.01$) over t=0.0 in the zero-shot setting as shown in Table 13. With t=0.0 and t=0.1, GPT-NeoX predicts only few values at all, and resorts to predicting "null" in many cases, contrary to t=1.0. This is, however, not due to pre-mature termination of generation as in the case of Flan-T5, but due to predicting "null" for the respective keys as instructed by the prompt. The "null" metrics shown in Table 12 reflect this behavior, with precision for t=0.0 and t=0.1 at 0.028, and 0.876 for recall, while for t=1.0 the precision is 0.040, and the recall is considerably lower at 0.299 (all with $p < 0.01$), which is due to much fewer "null" values being predicted in general.

Remarkably, the accuracy of GPT-NeoX in the one-shot setting is worse than the accuracy of any temperature in the zero-shot setting ($p < 0.01$) as shown in Table 13. This is due to GPT-NeoX repeating the example solution in its generations; for 423 documents the "Address (post town)" gold value in the example is repeated, similarly in 425 cases, the "Address (post code)" and "Charity Name", and in 426 cases the "Address (street)" value of the example is repeated. Other keys are "null" in the gold standard of the example (see appendix A.1) and are not repeated. We conclude that GPT-NeoX in most cases fails to generalize the example to new documents with temperature t=1.0.

Comparing the number of non-null values per key in sub-documents for the best setting in Figure 6 to InstructGPT, we observe that GPT-NeoX predicts on average 0.8 more non-null values per key than InstructGPT, and also predicts "null" for a whole document (equivalent to zero non-null values per key) more often. Note that is not structurally explained by the fact that GPT-NeoX has a smaller input window which yields more sub-documents, as the content of combined sub-documents is invariant with respect to the number of said sub-documents with exception of the overlap between sub-documents.

Next, we consider Figure 7a which illustrates how collisions are related to the number of sub-documents for the best setting of GPT-NeoX. The collision and full collision percentages rise approximately linearly with the number of sub-documents up to six sub-documents and slightly taper off with more sub-documents. Furthermore, the collision and full collision percentages across the depicted number of sub-documents are less separated than for InstructGPT.

Finally, Figure 8a depicts the relationship between accuracy and number of sub-documents which is only correlated slightly negatively (ρ=-0.21). This suggests that a larger maximum input window, and assuming all other things equal, would only marginally improve accuracy of GPT-NeoX, as even with just one sub-document (i.e., the document fits into the model), the accuracy is on average not considerably higher than with more sub-documents. Strictly speaking, this implication also assumes that correctly retrieving any single value is generally not dependent on the whole sub-document but only on little context around it, which may not necessarily be true.

InstructGPT

As shown in Table 15, InstructGPT does not strictly exhibit worse accuracy with higher accuracy, i.e., in the zero-shot setting the average accuracy with t=0.1 is 0.665 and thus slightly higher than with t=0.0 where it is 0.663 (p = 0.026). Accuracy with t=1.0, however, is considerably worse ($p < 0.01$) than the lower temperatures at 0.505 and has higher sample standard deviation. It is also considerably worse ($p < 0.01$) at correctly retrieving “null” values (F1 of 0.165 versus 0.322 with t=0.0 and 0.323 with t=0.1, respectively) as shown in Table 14.

Table 15 also exhibits that InstructGPT has higher accuracy in the one-shot compared to the zero-shot setting with the same temperature, but accuracy improves only marginally from 0.665 to 0.682, which translates to a relative improvement of 2.56% ($p < 0.01$). Notably, the accuracy is worse for the “Address (post code)” (by 0.027 absolute, $p < 0.01$) and “Annual Spending” (by 0.026 absolute, p = 0.0104) keys and worse still for “Annual Income” (by 0.06 absolute, $p < 0.01$). In that same vein, the average F1 score for “null” values decreases ($p < 0.01$) when attaching an example from 0.323 to 0.226, a relative change of -30.03%.

We hypothesize that this is directly related to the example we use. In support of this hypothesis, we show relative changes when going from the zero-shot to the one-shot setting in “null”-F1 for each key and whether the value for those keys is “null” in the attached example in Table 19. It exhibits, on a per-key basis, that gold “null” values in the example hurt performance on correctly retrieving “null” values, whereas gold non-“null” values improve recognition of “null” values. We leave to future work whether this applies more generally. Note that “Charity Name” and “Period End Date” are never “null” in the gold standard and hence “null”-F1 for these keys is undefined (cf. prevalence of keys in Table 2).

Next, we consider Figure 7b which illustrates how collisions are related to the number of sub-documents for the best setting of InstructGPT. Interestingly, and in contrast to GPT-NeoX, we observe that the collision and full collision ratios stay approximately constant after five sub-documents at approximately 0.35 and 0.25 respectively which is also reflected by the lower correlation at ρ=0.42 versus ρ=0.63 with GPT-NeoX. Lower correlation for these two variables is desirable for higher accuracy, because, in this setting, collision and full collision percentage correlate highly negatively with accuracy (ρ=-0.784 and ρ=-0.851). While the the collision percentage seems to be invariant to the number of sub-documents

Key	**Relative Change**	***p*-value**	**“null” in shot?**
Address (post town)	**150.0%**	<0.0001	No
Address (post code)	8.2%	0.0186	No
Address (street)	**45.8%**	0.0014	No
Charity Name	-	-	No
Charity Number	4.3%	0.1161	Yes
Annual Income	**-75.4%**	<0.0001	Yes
Period End Date	-	-	Yes
Annual Spending	**-66.7%**	<0.0001	Yes

Table 19: Relative changes in “null”-F1 by key of InstructGPT (t=0.1) from zero- to one-shot setting on Kleister Charity development set. Relative changes with $p < 0.01$ as determined by unpaired t-test marked **bold** and rounded to one decimal place.

past five sub-documents, we can observe in Figure 8b that nonetheless accuracy drops with more sub-documents (ρ=-0.69) also beyond five sub-documents. The combination of these two observations implies that longer documents are inherently more difficult and performance is likely not limited by the sub-document method itself. We leave to future work to examine this more closely in an ablation study by artificially constraining the maximum input window of InstructGPT.

Comparison to other work

We briefly compare the results of the best performing model, that is InstructGPT with temperature t=0.1 in the one-shot setting, on the test set of Kleister Charity to the results in Stanisławek et al. [2021]. Specifically, we reach an F1 score of **70.50±0.10**. Detailed results, as evaluated by the provided evaluation script, of InstructGPT, Flan-T5, and GPT-NeoX, can be found in appendix A.1.

The highest scoring method in Stanisławek et al. [2021] is achieved with LAMBERT [Garncarek et al., 2021] which scores **83.57** with unknown standard deviation over three runs. Additionally, other baselines are provided which all exceed our approach with InstructGPT. For context, it is important to note that all methods in Stanisławek et al. [2021] are trained on a training set of 1729 documents which are from the same distribution as the test set. Additionally, LayoutLM [Xu et al., 2020] and LAMBERT [Garncarek et al., 2021] use layout features.

Comparing the F1 score of our method on a per-key basis to LAMBERT [Garncarek et al., 2021], it is apparent that the three keys with the lowest accuracy are the same (“Address (street)”, “Annual Income”, and “Annual Spending”). Considering the data at hand, it is not surprising that the financial metrics are particularly difficult to retrieve, as financial reports usually contain many other numbers such as specific quarter results and income by source. Additionally, these numbers are usually laid out in a table, where layout information encodes useful information about the semantics of the numbers which is lost with our approach. This coincides with the fact that the three aforementioned keys are the keys where LAMBERT [Garncarek et al., 2021] offers the biggest absolute improvements (21.5, 21, and 21.6 points respectively) compared to InstructGPT.

Finally, we note that for the “Period End Date”, InstructGPT reaches 96.30±0.20 in F1, which is close to LAMBERT [Garncarek et al., 2021], which scores 96.80. Similarly for “Address (post town)”, InstructGPT reaches 79.20±0.20 (versus 83.70) and 90.70±0.10 (versus 95.80) for “Charity Number”.

6.2 Results on SROIE

For reference, the general baseline scores **0.037** in accuracy and **0.061** in F1 which we aim to beat with our more sophisticated pipeline approach.

Pipeline Approach

We discuss results with the pipeline approach in the zero-shot, one-shot and two-shot setting. Note that the temperature is fixed to the best performing temperature on the Kleister Charity development set in the zero-shot setting. Specifically Flan-T5 is set to t=0.0, GPT-NeoX to t=1.0, and InstructGPT to t=0.1. Note, that in this section we only consider the accuracy according to our definition, and not F1. Figure 10 visualizes the relationship of accuracy versus the number of shots and relates the considered LLM to each other.

For Flan-T5, the issue of almost always predicting only the first key persists in all settings (cf. Table 16), as observed also on Kleister Charity. Accuracy in the zero-shot setting is worse than the general baseline ($p < 0.01$) and improves by a factor of 13 ($p < 0.01$) in the one-shot setting, however, this is only an absolute improvement of 0.079. Markedly, accuracy in the two-shot setting is worse ($p < 0.01$) by absolute 0.043 points compared to the one-shot setting.

Similarly, GPT-NeoX in the zero-shot setting is also worse than the general baseline (cf. Table 17), however this is not statistically significant as computed by a one-sample t test ($p = 0.3$). Accuracy in the one-shot setting improves by a factor of 12 ($p < 0.01$) when compared to the zero-shot setting, which is an absolute improvement of 0.388. This is the largest relative and absolute increase in performance when adding an additional example in any setting for all considered models. The two-shot setting does not yield a statistically significant improvement in accuracy ($p = 0.8285$) compared to the one-shot setting.

InstructGPT outperforms ($p < 0.01$) all other models in all settings even in its worst ($p < 0.01$) setting, which is zero-shot (cf. Table 18). Accuracy increases with more shots ($p < 0.01$), with a relative improvement from the zero-shot to the one-shot setting of 22.8% and 13.2% from the one-shot to the two-shot setting, which suggests improvement tapering off with more shots. This, however, is not conclusive and experiments with more shots are needed to conclusively state or refute this observation. Note, that due to InstructGPT always predicting something for all keys in all documents, contrary to Flan-T5 and GPT-NeoX, the precision and recall are the same and consequently also the F1 score.

Finally, in Figure 11, we plot the parameter size of considered LLM against their accuracy in the three settings. We observe a very high correlation coefficient of ρ=0.990 in the zero-shot setting, and high correlation in the one-shot (ρ=0.878) and two-shot (ρ=0.899) settings. Notably, this is in contrast to sub-hypothesis c, which suggests a power-law relationship between performance and parameter size.

Comparison to other work

Now, in order to briefly compare to other work the results of our best performing model, that is InstructGPT in the two-shot setting, we consider the F1 score as computed by the provided evaluation script by Huang et al. [2019]. This is the metric used for ranking submissions[1]. We reach an F1 score of **0.780±0.001**. Considering submissions when the challenge was

[1] Ranking available at `https://rrc.cvc.uab.es/?ch=13&com=evaluation&task=3`.

first posed with a deadline, this would rank 8th out of 16 and considering all submissions, it would rank 61st out of 79 with StrucTexT [Li et al., 2021] topping the ranking Table with an F1 score of **0.987**.

6.3 Limitations

As mentioned earlier, with temperature t=0.0, contrary to GPT-NeoX and Flan-T5, InstructGPT does not yield deterministic generations in all cases (as can be seen from non-zero sample standard deviations). Because there is no public information on how exactly InstructGPT is provided via API, reasons for this remain unclear, which limits replicability of our results.

In addition to that, OpenAI do not publish any check-points or weights of any of their models, which limits replicability severely. We cannot exclude the possibility of OpenAI continuously optimizing models in accordance to user feedback, which is why we provide information on the time-frame, which might be a factor in the results, in which we ran the experiments, namely November/December 2022, and January 2023.

Finally, Brown et al. [2020] recognize that there are “some datasets where GPT-3 faces methodological issues related to training on large web corpora”. As the training data for GPT-3 based models, including InstructGPT, is not publicly known, we cannot know for certain that InstructGPT has never “seen” the Kleister Charity or SROIE data sets. This fundamentally limits our research hypothesis, which states that performance is evaluated *without training* which we can only assume, but not verify, for InstructGPT. Furthermore, replication of our work is, strictly speaking, impossible as InstructGPT has during the course of our work “seen” the data sets which we evaluate on.

7 Conclusion

7.1 Summary

In conclusion and per our research hypothesis, none of the herein considered pre-trained generative large language models reach satisfactory performance (accuracy $\geq$ 0.8) from text-only inputs and without training on Kleister Charity, nor on SROIE, two data sets that pose the task of KIE. Specifically, on the test sets, our best method achieves 0.722 and 0.780 in accuracy respectively. We do not definitively refute or confirm our hypothesis, as many variables have not been considered in this work which might improve performance. We leave this to future work which we discuss in more detail in section 7.2.

Next, we consider sub-hypothesis a, which states that sampling at higher temperatures negatively impacts performance due to increased diversity. By considering the results when varying the temperature in the zero-shot setting on Kleister Charity, specifically t=0.0, t=0.1, and t=1.0, we are unable to find a consistent negative relationship between accuracy and temperature *across* models. Specifically, for Flan-T5 in the zero-shot setting performance decreases ($p < 0.01$) when comparing the extremes, t=0 and t=1.0 and similarly, for InstructGPT performance decreases by relative 23.83% ($p < 0.01$). Notably, however, GPT-NeoX with t=1.0 reaches approximately twice the accuracy ($p < 0.01$) when compared to t=0.0. Thus, we refute this sub-hypothesis as we have presented statistically significant empirical evidence that this sub-hypothesis does not apply to GPT-NeoX in the zero-shot setting on Kleister Charity.

Sub-hypothesis b postulates that performance improves when providing few-shot examples during inference. Our results support this hypothesis for InstructGPT on Kleister Charity, where the one-shot performance improves on the zero-shot performance by 2.56% ($p < 0.01$) compared to the zero-shot setting, and on SROIE, where moving from zero- to one-shot, and from one-shot to two-shot increases ($p < 0.01$) performance by 22.8% and 13.2% respectively. For Flan-T5, accuracy increases from the zero- to the one-shot setting on Kleister Charity by absolute 0.016 ($p < 0.01$), and on SROIE by absolute 0.079 ($p < 0.01$), it decreases by absolute 0.043 ($p < 0.01$) when moving from the one-shot to the two-shot setting on SROIE. Results for GPT-NeoX suggest that the data set is relevant for whether performance improves or declines when providing few-shot examples. While adding a single example improves performance by an absolute 0.388 ($p < 0.01$) on SROIE, it decreases performance drastically by relative 83.3% ($p < 0.01$) on Kleister Charity. Combining our observations, we refute this sub-hypothesis as we have presented statistically significant empirical evidence that this sub-hypothesis does not apply to all models in all cases.

Finally, we concern ourselves with sub-hypothesis c. It states that there is a power-law relationship between performance and parameter size when not bottlenecked by compute or data set size as observed by Kaplan et al. [2020]. Across both data sets, Kleister Charity and SROIE, and various settings of temperature and number of examples, we observe very high linear correlation between parameter size of the considered models and their performance, especially in the zero-shot setting (ρ=1.000 and ρ=0.999). Importantly, we only consider three models from three different model families hence comparability with respect to just parameter size is limited. Interestingly, even with the observed linear relationship, for both data sets any setting of InstructGPT is better (p<0.01) than any setting of Flan-T5 and GPT-NeoX because of its number of parameters vastly exceeding these models by factor of 58 and 9 respectively. As we consider only three models with different parameter sizes which are not directly comparable, we are not confident in refuting or confirming this particular hypothesis with the data at hand.

7.2 Future Work

In order to increase the validity and generality of our findings, future work may examine different generative pre-trained LLM on other data sets in the KIE domain and with other examples during inference to study specifically the impact of examples on performance.

One promising avenue to increase performance of our method is prompt engineering [Singh et al., 2022, Arora et al., 2022, Chen et al., 2022] which shows impressive improvements on many tasks. Future work may then examine the impact of prompts on performance and investigate whether satisfactory performance can be achieved with different prompts.

Finally, future work may concern itself with KIE in other languages and with ablation studies on the maximum input window of InstructGPT by constraining it to the same size as that of Flan-T5 and GPT-NeoX.

7.3 Acknowledgments

We would like to thank Prof. Dr. Katja Markert for supervising this thesis. We would also like to acknowledge Levity AI for their generous funding of this research and Charul Daudkhane for offering her technical and academic expertise during the course of this thesis.

A Appendix

A.1 Kleister Charity

Normalization

The F1 score of the provided evaluation is based on exact string match which does not take into account overlap. For our own evaluation, we normalize values of three out of the eight keys as detailed below. Our motivation for this is similar to the one in Kim et al. [2022], however, contrary to normalization, they use Tree Edit Distance (TED) to evaluate their models on document information extraction.

- "Address (post town)"
  - `<Solution City>` and `City of <Solution City>` are set to be equivalent.
  - We allow for a Levenshtein [Levenshtein et al., 1966] distance of one. This allows for equivalency between e.g. `St.Mary`, `St._Mary`, `St_Mary`, and `St._Mary`.
- "Address (street)"
  - We delete spaces around hyphens, e.g. `7-14_Great_Dover_Street` and `7_-_14_Great_Dover_Street` are equivalent.
- "Charity Name"
  - We delete spaces around hyphens as with the "Address (street)" key.
  - We allow for a Levenshtein [Levenshtein et al., 1966] distance of one as with the "Address (post town)" key.
  - `<Charity Name>_Ltd`, `<Charity Name>_Ltd.`, and `<Charity Name>_Limited` are set to be equivalent.
  - `&` and the literal `and` are set to be equivalent.

Additionally, we set `’` (U+2019) to be equivalent to `'`.

Shot

Ushaw Moor Pre-school/Childcare Durham Road Ushaw Moor Durham DH7 7LF Telephone 01913737536 Annual General Meeting October Attending meeting: Catherine Winn, Julie Davison, Lindsley Davison, Deborah Mellis, Megan Bowery, Nikki Lowerson, Karen Smith, Janice Laight, Kayleigh Hughes, Abbie Syers Apologies from Lynsey Everett. Up date on school situation: Mr Truman has left the school after a along absent, and a new head is now in position, Mrs Maughan has had a chat with Julie, and the girls from the pre- school are more optimistic with the future links between us. As the school have a new head they will be due an ofsted and Mrs Maughan has put this as her priority but has already invited our children to attend their Christmas activities. Other ideas included: Stay and play days, with parents and shared outdoor activities. Pre-School: Karen reported the pre-school had now used up all their childcare spaces and would not be taking any September starters from the childcare setting. As the pre-school taking children doing 3o hours per week number of spaces were less. Julie is concerned about turning these younger away may have a impact on next year's intake. Childcare: Lindsley has concerns about the number of children using the childcare during the school holidays, as some days there are more staff than children. Catherine reminded the staff that they should take their due holidays outside of term time when the numbers are low. It was agreed that we would monitor the situation and maybe change opening hours. Next staff meeting to be arranged Agenda: Christmas activities

Key	Gold Value	Corrected Gold Value
Address (post town)	DURHAM	DURHAM
Address (post code)	DH7 7ND	DH7 7LF
Address (street)	9 ASH AVENUE	DURHAM ROAD
Charity Name	Ushaw Moor Pre-School	Ushaw Moor Pre-School
Charity Number	1072461	-
Annual Income	199658.00	-
Period End Date	2017-10-31	-
Annual Spending	189280.00	-

Table 20: Gold values with corrections of example used for one-shot setting on Kleister Charity.

Additional Evaluation

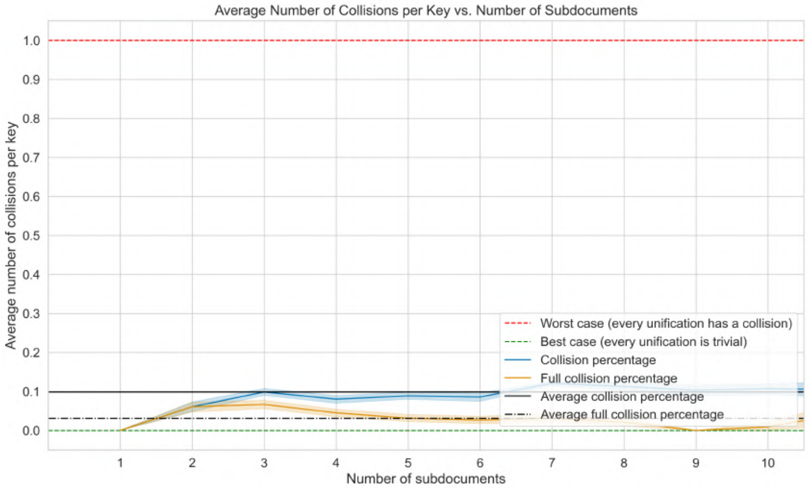


Figure 12: Average number of collisions per key versus number of sub-documents on Kleister Charity development set for best setting of Flan-T5. Shaded area represents 95% confidence interval of three runs. Note: in case of two sub-documents, every collision is by definition also a full collision.

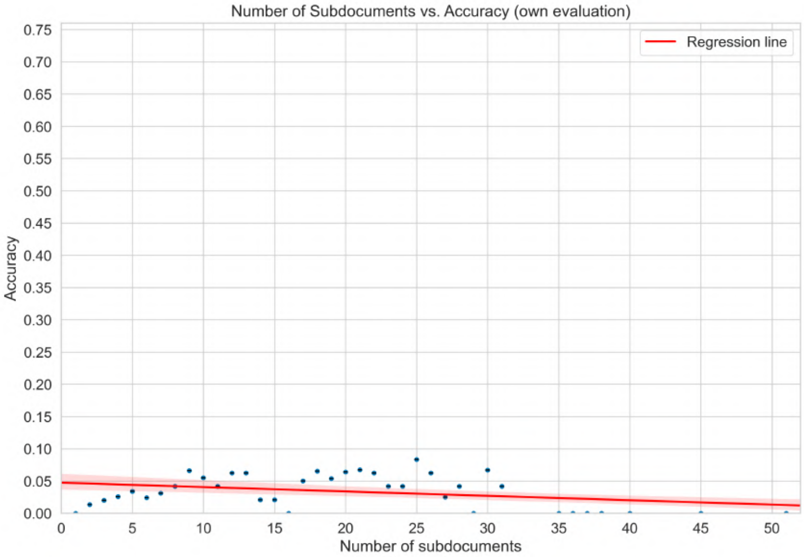


Figure 13: Accuracy versus number of sub-documents on Kleister Charity development set for best setting of Flan-T5. Shaded area of linear regression line represents 95% confidence interval estimated using bootstrap sampling. Error bars represent minimum and maximum over three runs.

Key	**Development Set**		**Test Set**	
	General	**Specific**	**General**	**Specific**
Address (post town)	0	18.2	0	13.9
Address (post code)	0	60.0	0	63.5
Address (street)	0	23.6	0	32.8
Charity Name	6.3	6.3	5.0	5.3
Charity Number	65.4	75.6	68.4	78.6
Annual Income	0	11.2	0	16.5
Period End Date	0.8	9.6	0	13.3
Annual Spending	0	2.9	0	2.1
Average	9.1	25.9	0	28.3

Table 21: F1 scores on Kleister Charity with provided evaluation script of general and Kleister Charity specific baselines. Baselines use window w=40 and error_percentage=0.18. No ranges are given, as the baselines are deterministic if run on the same day (cf. section 4.2). Average is macro-averaged over keys. Results rounded to one decimal place.

Key	**Development Set**				**Test Set**
		Zero-Shot		**One-Shot**	**One-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=0.0**	**t=0.0**
Address (post town)	22.4±0.0	22.2±0.5	15.9±1.5	**26.1±0.0**	28.6±0.0
Address (post code)	0±0.0	0±0.0	0±0.0	0±0.0	0±0.0
Address (street)	0±0.0	0±0.0	0±0.0	0±0.0	0±0.0
Charity Name	0±0.0	0±0.0	0±0.0	**2.8±0.0**	2.5±0.0
Charity Number	0±0.0	0±0.0	0±0.0	0±0.0	0±0.0
Annual Income	0±0.0	0±0.0	0±0.0	0±0.0	0±0.0
Period End Date	0±0.0	0±0.0	0±0.0	0±0.0	0±0.0
Annual Spending	0±0.0	0±0.0	0±0.0	0±0.0	0±0.0
Average	2.8±0.0	2.8±0.1	2.0±0.2	**3.6±0.0**	3.9±0.0

Table 22: F1 scores of Flan-T5 on Kleister Charity with provided evaluation script. Best scores on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is macro-averaged over keys. Results rounded to one decimal place.

Key	**Development Set**				**Test Set**
		Zero-Shot		**One-Shot**	**Zero-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=1.0**	**t=1.0**
Address (post town)	15.2±0.0	15.4±0.2	**15.8+1.1**	0.2±0.3	19.8±1.8
Address (post code)	**5.8±0.0**	5.7±0.2	3.3±0.8	0.2±0.3	6.4±1.7
Address (street)	**5.5±0.0**	5.0±0.3	3.0±1.1	0.1±0.2	5.4±2.1
Charity Name	19.8±0.0	19.8±1.3	**32.1±1.2**	0.8±0.2	36.6±1.1
Charity Number	**16.5±0.0**	15.5±0.1	16.3±2.5	4.9±1.4	25.0±1.9
Annual Income	**2.1±0.0**	2.0±0.2	0.7±0.2	0.2±0.3	1.5±0.8
Period End Date	23.0±0.0	23.9±1.2	**41.0±2.9**	17.1±1.2	50.8±0.2
Annual Spending	0±0.0	0.2±0.4	**0.3±0.2**	0.2±0.3	0.3±0.3
Average	11.0±0.0	10.9±0.1	**14.1±1.1**	3.0±0.1	18.2±0.8

Table 23: F1 scores of GPT-NeoX on Kleister Charity with provided evaluation script. Best scores on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is macro-averaged over keys. Results rounded to one decimal place.

Key	**Development Set**				**Test Set**
		Zero-Shot		**One-Shot**	**One-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=0.1**	**t=0.1**
Address (post town)	77.0±0.1	77.0±0.1	62.1±2.4	**79.5±0.2**	79.2±0.2
Address (post code)	**67.1±0.1**	66.9±0.3	47.8±1.0	64.9±0.4	70.1±0.3
Address (street)	34.0±0.2	34.3±0.4	28.0±1.9	**45.6±0.3**	52.8±0.3
Charity Name	60.0±0.0	59.9±0.1	57.5±1.7	**63.3±0.3**	68.7±0.2
Charity Number	83.6±0.5	87.4±0.6	75.3±1.1	**94.5±0.0**	90.7±0.1
Annual Income	**46.0±0.5**	45.4±0.8	23.0±2.0	43.2±0.2	53.7±0.3
Period End Date	95.7±0.0	95.6±0.2	86.9±0.5	**95.8±0.2**	96.3±0.2
Annual Spending	**48.7±0.1**	47.5±0.8	21.0±3.1	47.8±0.8	52.6±0.2
Average	64.0±0.1	64.2±0.1	50.3±1.1	**66.8±0.0**	70.5±0.1

Table 24: F1 scores of InstructGPT on Kleister Charity with provided evaluation script. Best scores on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is macro-averaged over keys. Results rounded to one decimal place.

Key	**Accuracy**		**Development Set**			**Test Set**
			Zero-Shot		**One-Shot**	**One-Shot**
		t=0.0	**t=0.1**	**t=1.0**	**t=0.0**	**t=0.0**
Address (post town)	Standard	0.156±0.000	0.152±0.004	0.136±0.015	**0.265±0.000**	0.287±0.000
	Lenient	0.175±0.000	0.173±0.002	0.170±0.017	0.361±0.000	0.412±0.000
Address (post code)	Standard	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	Lenient	0.032±0.000	0.032±0.000	0.032±0.000	0.032±0.000	0.028±0.000
Address (street)	Standard	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	Lenient	0.114±0.000	0.114±0.000	0.112±0.001	0.105±0.000	0.074±0.000
Charity Name	Standard	0.000±0.000	0.000±0.000	0.000±0.000	**0.016±0.000**	0.015±0.000
	Lenient	0.000±0.000	0.000±0.000	0.000±0.000	0.016±0.000	0.013±0.000
Charity Number	Standard	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	Lenient	0.007±0.000	0.007±0.000	0.007±0.000	0.007±0.000	0.011±0.000
Annual Income	Standard	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	Lenient	0.014±0.000	0.014±0.000	0.014±0.000	0.014±0.000	0.021±0.000
Period End Date	Standard	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	Lenient	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
Annual Spending	Standard	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000
	Lenient	0.014±0.000	0.014±0.000	0.014±0.000	0.014±0.000	0.03±0.000
Average	Standard	0.019±0.000	0.019±0.000	0.017±0.002	**0.035±0.000**	0.037±0.000
	Lenient	0.044±0.000	0.044±0.000	0.043±0.002	0.068±0.000	0.074±0.000

Table 25: Standard and enient ("oracle" considering *all* values predicted in sub-documents and perfect unification) accuracy of Flan-T5 on Kleister Charity. Best standard accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places.

Key	**Accuracy**	**Development Set**				**Test Set**
			Zero-Shot		**One-Shot**	**Zero-Shot**
		t=0.0	**t=0.1**	**t=1.0**	**t=1.0**	**t=1.0**
Address (post town)	Standard	0.088±0.000	0.058±0.050	**0.155±0.010**	0.002±0.003	0.191±0.015
	Lenient	0.120±0.000	0.120±0.002	0.229±0.011	0.083±0.003	0.251±0.006
Address (post code)	Standard	**0.031±0.000**	0.020±0.018	0.030±0.007	0.002±0.003	0.058±0.016
	Lenient	0.061±0.000	0.062±0.003	0.054±0.015	0.011±0.005	0.074±0.014
Address (street)	Standard	**0.031±0.000**	0.020±0.017	0.027±0.009	0.001±0.001	0.051±0.018
	Lenient	0.125±0.000	0.124±0.003	0.055±0.017	0.008±0.001	0.073±0.015
Charity Name	Standard	0.143±0.000	0.144±0.007	**0.365±0.017**	0.007±0.002	0.444±0.003
	Lenient	0.120±0.000	0.123±0.006	0.389±0.022	0.124±0.019	0.438±0.016
Charity Number	Standard	0.096±0.000	0.060±0.052	**0.153±0.024**	0.038±0.009	0.236±0.014
	Lenient	0.105±0.000	0.099±0.001	0.202±0.017	0.046±0.009	0.274±0.011
Annual Income	Standard	**0.012±0.000**	0.007±0.006	0.005±0.001	0.001±0.001	0.012±0.007
	Lenient	0.027±0.000	0.025±0.002	0.012±0.003	0.011±0.002	0.026±0.009
Period End Date	Standard	0.134±0.000	0.095±0.082	**0.345±0.030**	0.127±0.012	0.447±0.004
	Lenient	0.139±0.000	0.139±0.007	0.415±0.032	0.169±0.016	0.535±0.002
Annual Spending	Standard	0.000±0.000	0.001±0.001	**0.002±0.001**	0.001±0.001	0.002±0.002
	Lenient	0.016±0.000	0.015±0.003	0.008±0.001	0.011±0.003	0.011±0.003
Average	Standard	0.067±0.000	0.051±0.023	**0.138±0.009**	0.023±0.000	0.183±0.005
	Lenient	0.089±0.000	0.088±0.003	0.17±0.015	0.058±0.007	0.21±0.009

Table 26: Standard and lenient (“oracle” considering *all* values predicted in sub-documents and perfect unification) accuracy of GPT-NeoX on Kleister Charity. Best standard accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places.

Key	**Accuracy**	**Development Set**				**Test Set**
			Zero-Shot		**One-Shot**	**One-Shot**
		t=0.0	**t=0.1**	**t=1.0**	**t=0.1**	**t=0.1**
Address (post town)	Standard	0.791±0.001	0.790±0.001	0.598±0.017	**0.806±0.001**	0.812±0.002
	Lenient	0.788±0.001	0.787±0.001	0.624±0.009	0.798±0.001	0.800±0.001
Address (post code)	Standard	**0.673±0.001**	0.671±0.003	0.449±0.009	0.644±0.003	0.701±0.003
	Lenient	0.700±0.002	0.698±0.003	0.511±0.006	0.685±0.005	0.721±0.003
Address (street)	Standard	0.368±0.001	0.368±0.004	0.285±0.016	**0.491±0.001**	0.550±0.003
	Lenient	0.384±0.000	0.385±0.003	0.314±0.020	0.483±0.001	0.542±0.003
Charity Name	Standard	0.725±0.000	0.723±0.001	0.694±0.018	**0.767±0.003**	0.846±0.000
	Lenient	0.621±0.001	0.622±0.001	0.604±0.014	0.657±0.0	0.709±0.002
Charity Number	Standard	0.826±0.002	0.863±0.003	0.735±0.005	**0.938±0.000**	0.908±0.001
	Lenient	0.945±0.001	0.944±0.001	0.787±0.009	0.949±0.001	0.933±0.002
Annual Income	Standard	**0.452±0.004**	0.446±0.009	0.212±0.016	0.386±0.001	0.484±0.004
	Lenient	0.576±0.003	0.567±0.005	0.264±0.019	0.445±0.002	0.517±0.003
Period End Date	Standard	0.957±0.000	0.956±0.001	0.845±0.006	**0.959±0.002**	0.958±0.001
	Lenient	0.964±0.000	0.963±0.001	0.873±0.017	0.967±0.001	0.959±0.001
Annual Spending	Standard	**0.480±0.001**	0.469±0.007	0.194±0.031	0.443±0.007	0.495±0.003
	Lenient	0.592±0.001	0.586±0.004	0.248±0.036	0.500±0.002	0.545±0.002
Average	Standard	0.663±0.000	0.665±0.001	0.505±0.008	**0.682±0.000**	0.722±0.000
	Lenient	0.696±0.001	0.694±0.002	0.528±0.016	0.686±0.002	0.716±0.002

Table 27: Standard and lenient ("oracle" considering *all* values predicted in sub-documents and perfect unification) accuracy of InstructGPT on Kleister Charity. Best standard accuracy on development set for each row in **bold**. Ranges are sample standard deviations over three runs. Average is micro-averaged. Results rounded to three decimal places.

Key	**Setting**			
		Zero-Shot		**One-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=0.0**
ρ(# of non-null values, collision pct.)	0.999	0.991	0.999	1.000
ρ(# of non-null values, accuracy)	0.996	1.000	0.996	0.999
ρ(collision pct., accuracy)	0.997	0.988	0.997	0.999
ρ(full collision pct., accuracy)	0.998	0.931	0.998	0.999

Table 28: Correlations of Flan-T5 on Kleister Charity development set. Values are micro-averaged. Results rounded to three decimal places.

Key	**Setting**			
	Zero-Shot			**One-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=1.0**
ρ(# of non-null values, collision pct.)	0.392	0.440	0.408	0.990
ρ(# of non-null values, accuracy)	0.512	0.523	0.182	-0.474
ρ(collision pct., accuracy)	-0.205	-0.137	-0.141	-0.459
ρ(full collision pct., accuracy)	-0.345	-0.206	-0.611	0.371

Table 29: Correlations of GPT-NeoX on Kleister Charity development set. Values are micro-averaged. Results rounded to three decimal places.

Key	**Setting**			
	Zero-Shot			**One-Shot**
	t=0.0	**t=0.1**	**t=1.0**	**t=0.1**
ρ(# of non-null values, collision pct.)	-0.360	-0.356	-0.223	-0.463
ρ(# of non-null values, accuracy)	0.583	0.578	0.875	0.517
ρ(collision pct., accuracy)	-0.685	-0.654	-0.412	-0.784
ρ(full collision pct., accuracy)	-0.806	-0.752	-0.698	-0.851

Table 30: Correlations of InstructGPT on Kleister Charity development set. Values are micro-averaged. Results rounded to three decimal places.

A Appendix

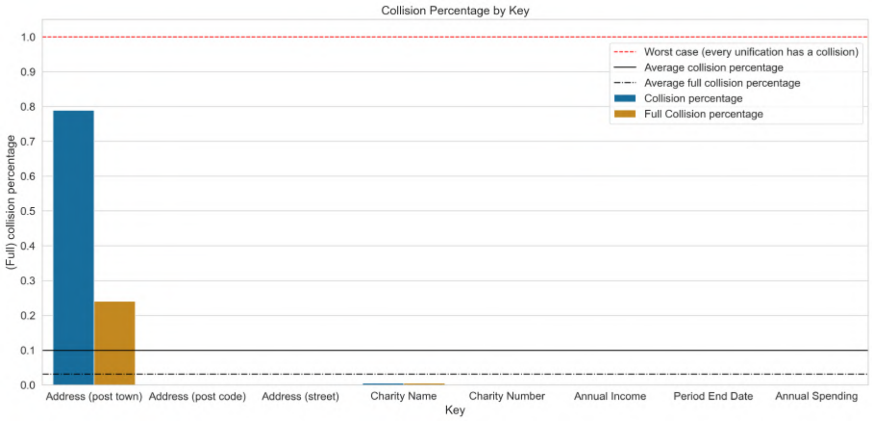


Figure 14: Collision percentages by key on Kleister Charity development set for best setting of Flan-T5. Error bars represent minimum and maximum over three runs.

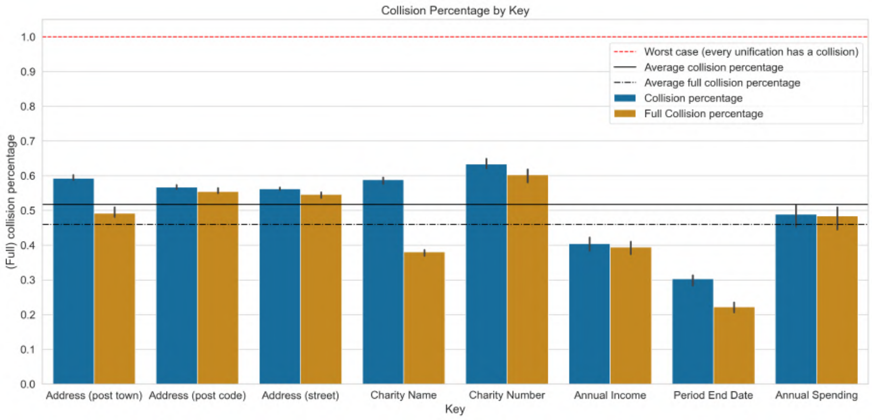


Figure 15: Collision percentages by key on Kleister Charity development set for best setting of GPT-NeoX. Error bars represent minimum and maximum over three runs.

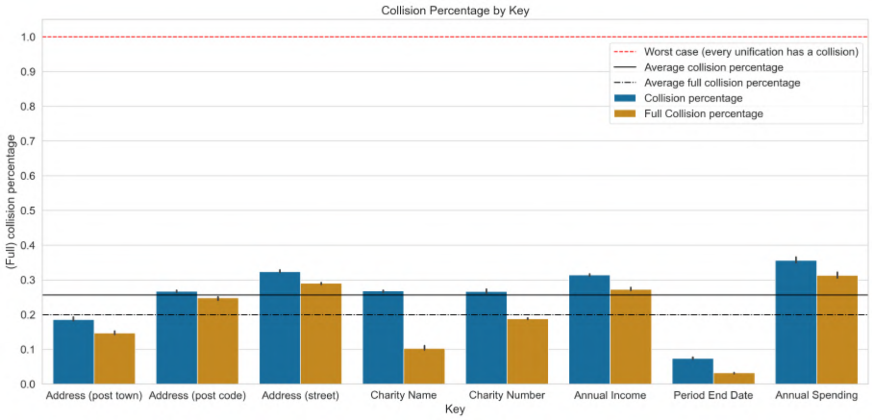


Figure 16: Collision percentages by key on Kleister Charity development set for best setting of InstructGPT. Error bars represent minimum and maximum over three runs.

A.2 SROIE

Shots

First Shot

SYARIKAT PERNIAGAAN GIN KEE (81109-A) NO 290, JALAN AIR PANAS, SETAPAK, 53200, KUALA LUMPUR TEL : 03-40210276 GST ID : 000750673920 SIMPLIFIED TAX INVOICE CASH DOC NO GOODS SOLD ARE NOT RETURNABLE, THANK YOU. : CS00012944 DATE: 25/01/2018 CASHIER 10.80 : USER TIME: 14:29:00 SALESPERSON: 180.00 REF. : ITEM QTY S/PRICE AMOUNT TAX 1007 3 63.60 190.80 SR 12MM 4/8B PLYWOOD TOTAL QTY: 3 190.80 TOTAL SALES (EXCLUDING GST) : 180.00 DISCOUNT : 0.00 TOTAL GST : 10.80 ROUNDING : 0.00 TOTAL SALES (INCLUSIVE OF GST) : 190.80 CASH : 190.80 CHANGE : 0.00 GST SUMMARY TAX CODE % AMT(RM) TAX(RM) SR 6 180.00 10.80 TOTAL :

Second Shot

POPULAR BOOK CO. (M) SDN BHD (CO. NO. 113825-W) (GST REG NO. 001492992000) NO 8, JALAN 7/118B, DESA TUN RAZAK 56000 KUALA LUMPUR, MALAYSIA AEON SHAH ALAM TEL : 03-55235214 27/02/18 21:22 TASHA SLIP NO. : 8020188757 TRANS : 204002 MEMBER CARD NO : 1001016668849 CARD EXPIRY : 31/05/18 DESCRIPTION AMOUNT DOCUMENT HOL A4 1466A-TRA 2PC @ 1.15 MEMBER DISCOUNT PB PVC A4 L-FLD PBA4L25 2PC @ 3.90 MEMBER DISCOUNT CANON CAL AS120V GREY MEMBER DISCOUNT NASI'APR16/SEASHORE [BK] 2.30 T -0.24 7.80 T -0.78 29.90 T -2.99 5.00 Z TOTAL RM INCL OF GST ROUNDING ADJ TOTAL RM CASH CHANGE 40.99 0.01 41.00 -51.00 10.00 ITEM COUNT GST SUMMARY T @ 6% Z @ 0% AMOUNT (RM) 33.95 5.00 6 TAX (RM) 2.04 0.00 TOTAL SAVINGS -4.01 BE A POPULAR CARD MEMBER AND ENJOY SPECIAL DISCOUNTS THANK YOU. PLEASE COME AGAIN. WWW.POPULAR.COM.MY BUY CHINESE BOOKS ONLINE WWW.POPULARONLINE.COM.MY

Key	**Gold Value**
Company Name	SYARIKAT PERNIAGAAN GIN KEE
Date of Receipt	02/01/2018
Address of Company	NO 290, JALAN AIR PANAS, SETAPAK, 53200, KUALA LUMPUR.
Total	39.75

Table 31: Gold values of the the first example used for one- and two-shot settings on SROIE.

Key	**Gold Value**
Company Name	POPULAR BOOK CO. (M) SDN BHD
Date of Receipt	27/02/18
Address of Company	NO 8, JALAN 7/118B, DESA TUN RAZAK 56000 KUALA LUMPUR, MALAYSIA
Total	41.00

Table 32: Gold values of the the second example used for the two-shot settings on SROIE.

Bibliography

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and C. V. Jawahar. ICDAR2019 competition on scanned receipt OCR and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, sep 2019. doi: 10.1109/icdar.2019.00244. URL `https://doi.org/10.1109%2Ficdar.2019.00244`.

Łukasz Borchmann, Michał Pietruszka, Tomasz Stanislawek, Dawid Jurkiewicz, Michał Turski, Karolina Szyndler, and Filip Graliński. Due: End-to-end document understanding benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Fengbin Zhu, Chao Wang, Wenqiang Lei, Ziyang Liu, and Tat Seng Chua. Rdu: A region-based approach to form-style document understanding. *arXiv preprint arXiv:2206.06890*, 2022.

Yulin Li, Yuxi Qian, Yuechen Yu, Xiameng Qin, Chengquan Zhang, Yan Liu, Kun Yao, Junyu Han, Jingtuo Liu, and Errui Ding. Structext: Structured text understanding with multi-modal transformers. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1912–1920, 2021.

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*, 2017.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *arXiv preprint arXiv:1811.02549*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://aclanthology.org/D07-1090`.

Daniel Jurafsky and James H Martin. Speech and language processing: An introduction to speech recognition, computational linguistics and natural language processing. *Upper Saddle River, NJ: Prentice Hall*, 2008.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Ralph Grishman. Information extraction. *IEEE Intelligent Systems*, 30(5):8–15, 2015. doi: 10.1109/MIS.2015.68.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. Position-aware attention and supervised data improve slot filling. In *Conference on Empirical Methods in Natural Language Processing*, 2017.

George Stoica, Emmanouil Antonios Platanios, and Barnabás Póczos. Re-tacred: Addressing shortcomings of the tacred dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13843–13850, 2021.

Tomasz Stanisławek, Filip Graliński, Anna Wróblewska, Dawid Lipiński, Agnieszka Kaliska, Paulina Rosalska, Bartosz Topolski, and Przemysław Biecek. Kleister: Key Information Extraction Datasets Involving Long Documents with Complex Layouts. In *Document Analysis and Recognition – ICDAR 2021*, pages 564–579. Springer International Publishing, 2021. doi: 10.1007/978-3-030-86549-8_36. URL `https://doi.org/10.1007%2F978-3-030-86549-8_36`.

Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. Cord: a consolidated receipt dataset for post-ocr parsing. In *Workshop on Document Intelligence at NeurIPS 2019*, 2019.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. Funsd: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 1–6. IEEE, 2019.

Yiheng Xu, Tengchao Lv, Lei Cui, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, and Furu Wei. Xfund: A benchmark dataset for multilingual visually rich form understanding. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3214–3224, 2022.

Jiapeng Wang, Chongyu Liu, Lianwen Jin, Guozhi Tang, Jiaxin Zhang, Shuaitao Zhang, Qianying Wang, Yaqiang Wu, and Mingxiang Cai. Towards robust visual information extraction in real world: New dataset and novel solution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2738–2745, 2021.

Hongbin Sun, Zhanghui Kuang, Xiaoyu Yue, Chenhao Lin, and Wayne Zhang. Spatial dual-modality graph reasoning for key information extraction. *arXiv preprint arXiv:2103.14470*, 2021.

Michael D Garris. Nist special database 2. nist structured forms reference set of binary images (sfrs), 2017.

He Guo, Xiameng Qin, Jiaming Liu, Junyu Han, Jingtuo Liu, and Errui Ding. Eaten: Entity-aware attention for single shot visual text extraction, 2019. URL `https://arxiv.org/abs/1909.09380`.

Dipali Baviskar, Swati Ahirrao, and Ketan Kotecha. Multi-layout unstructured invoice documents dataset: A dataset for template-free invoice processing and its evaluation using ai approaches. *IEEE Access*, 9:101494–101512, 2021.

Xiaohui Zhao, Endi Niu, Zhuo Wu, and Xiaoguang Wang. Cutie: Learning to understand documents with convolutional universal text information extractor, 2019. URL `https://arxiv.org/abs/1903.12363`.

Daniel Schuster, Klemens Muthmann, Daniel Esser, Alexander Schill, Michael Berger, Christoph Weidling, Kamil Aliyev, and Andreas Hofmeier. Intellix–end-user trained information extraction for document archiving. In *2013 12th International Conference on Document Analysis and Recognition*, pages 101–105. IEEE, 2013.

Rasmus Berg Palm, Ole Winther, and Florian Laws. Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 406–413. IEEE, 2017.

Lei Cui, Yiheng Xu, Tengchao Lv, and Furu Wei. Document ai: Benchmarks, models and applications. *arXiv preprint arXiv:2111.08609*, 2021.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2020. doi: 10.1145/3394486.3403172. URL `https://doi.org/10.1145%2F3394486.3403172`.

Jian Ni, Gaetano Rossiello, Alfio Gliozzo, and Radu Florian. A Generative Model for Relation Extraction and Classification, 2022. URL `https://arxiv.org/abs/2202.13229`.

Christoph Alt, Marc Hübner, and Leonhard Hennig. Improving Relation Extraction by Pre-Trained Language Representations. *arXiv preprint arXiv:1906.03088*, 2019.

Jiale Han, Shuai Zhao, Bo Cheng, Shengkun Ma, and Wei Lu. Generative prompt tuning for relation classification. *arXiv preprint arXiv:2210.12435*, 2022.

Martin Josifoski, Nicola De Cao, Maxime Peyrard, and Robert West. Genie: generative information extraction. *arXiv preprint arXiv:2112.08340*, 2021.

Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*, 2011.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. GPT-NeoX-20B: An Open-Source Autoregressive Language Model. *arXiv preprint arXiv:2204.06745*, 2022.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Łukasz Garncarek, Rafał Powalski, Tomasz Stanisławek, Bartosz Topolski, Piotr Halama, Michał Turski, and Filip Graliński. LAMBERT: Layout-Aware Language Modeling for Information Extraction. In *International Conference on Document Analysis and Recognition*, pages 532–547. Springer, 2021.

Chandan Singh, John X. Morris, Jyoti Aneja, Alexander M. Rush, and Jianfeng Gao. Explaining patterns in data with language models via interpretable autoprompting, 2022. URL `https://arxiv.org/abs/2210.01848`.

Simran Arora, Avanika Narayan, Mayee F. Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. Ask me anything: A simple strategy for prompting language models, 2022. URL `https://arxiv.org/abs/2210.02441`.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. KnowPrompt: Knowledge-Aware Prompt-Tuning with Synergistic Optimization for Relation Extraction. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 2778–2788, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3511998. URL `https://doi.org/10.1145/3485447.3511998`.

Geewook Kim, Teakgyu Hong, Moonbin Yim, Jeongyeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park, 2022. URL `https://arxiv.org/abs/2111.15664`.