Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Jon Steinar Folstad

Transformer Pre-Trained Language Models and Active Learning for Improved Blocking Performance in Entity Matching

Master's thesis in Computer Science
Supervisor: Jon Atle Gulla
Co-supervisor: Nils Barlaug
August 2021

NTNU
Norwegian University of Science and Technology

Jon Steinar Folstad

Transformer Pre-Trained Language Models and Active Learning for Improved Blocking Performance in Entity Matching

Master's thesis in Computer Science
Supervisor: Jon Atle Gulla
Co-supervisor: Nils Barlaug
August 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Entity matching (EM) aims to reduce the entropy between two different data sources by identifying which records refer to the same real-world entity. Typically, many proposed blocking approaches require sufficient human expert involvement and/or a large amount of labeled data, however often unavailable for EM applications to achieve useful models. In this work, we propose TopKDAL, a deep learning-based approach targeting a low-resource setting through a combination of active learning (AL) with pre-trained transformer language models (TPLM). TPLMs are a promising approach towards hands-off blocking to obtain semantically meaningful sentence embeddings and the ability to learn where to pay attention between the records. Doing so, TPLMs unveil similarities between entities. We incorporate active learning to select informative examples to fine-tune a TPLM and to cope with labeled data scarcity. In this way, we investigate how to reduce the required labeling effort while maintaining the model accuracy and the blocking performance.

Experiments on five EM benchmark datasets showed the effectiveness of TopKDAL with respect to pair completeness (PC), reduction rate, and running time. We found active learning strategies yield better results with an order of magnitude fewer labeled examples compared to a supervised Baseline trained on all available data. TopKDAL demonstrates best performance with Imbalanced-Partition-2 and Balanced-Uncertainty. Balanced-Uncertainty consumes an initial balanced training set, which contributes to kick-start the active learning performance and reduces the risk for cold start problems. However, it is an extra overhead required to unlock the potential with a balanced starting strategy. Towards mitigating biases, Random-P/N yield competitive performance towards the more advanced query sampling strategies when it is trained initially on an imbalanced initial training set. Our proposed TopKDAL requires no design decisions from a human and features are learned from the data. Fine-tuning hyperparameters are still recommended to optimize the model performance.

Sammendrag

Entitetsgjenkjenning har som mål å redusere entropien mellom to ulike datakilder ved å identifisere hvilke record som refererer til de samme entitetene i virkeligheten. Vanligvis krever mange foreslåtte blokkeringsmetoder tilstrekkelig menneskelig domenekunnskap og/eller en stor mengde merkede data, imidlertid er dette ofte utilgjengelig for applikasjoner i entitetsgjenkjenning for å oppnå nyttige blokkeringmodeller. I dette arbeidet foreslår vi TopKDAL, en dyplæringsbasert tilnærming som er rettet mot en situasjon med begrenset mengde merkede data ved å kombinere aktiv læring med forhåndstrente transformer språksmodeller. Disse språkmodellene er en lovende tilnærming for å oppnå semantisk meningsfulle embeddings og muligheten til å lære hvor man skal fokusere mellom records. Ved å gjøre det avslører transformer modellene likheter mellom entitetene. Vi bruker aktiv læring for å velge informative eksempler for å finjustere en transformer språkmodell og for å takle knapphet på merkede data. På denne måten undersøker vi hvordan arbeidet med merking av data kan reduseres, samtidig som modellnøyaktigheten og blokkeringsytelsen opprettholdes.

Eksperimenter med fem referansedatasett for entitetsgjenkjenning viser effektiviteten til TopKDAL med hensyn på pairs completeness (PC), reduction rate (RR) og kjøretid. Vi fant at aktive læringsstrategier gir bedre resultater med en størrelsesorden færre merkede eksempler sammenlignet med en supervised baseline som er trent på alle tilgjengelige data. TopKDAL oppnådde den beste ytelsen med Imbalanced-Partition-2 og Balanced-Uncertainty. Balanced-Uncertainty trenes basert på et balansert treningssett i starten, som bidrar til å forbedre den aktive læringsytelsen og redusere risikoen for kaldstart-problemer. Imidlertid kreves dette ekstra implementasjon for å muliggjøre potensialet med en balansert start strategi. For å redusere biases ble Random-P/N-strategien trent med et ubalansert treningssett som gir konkurransedyktig ytelse mot de mer avanserte prøvetaking strategiene. Vår foreslåtte TopKDAL krever ingen menneskelige designbeslutninger, og features læres fra dataene. Finjustering av hyperparametere anbefales fortsatt for å optimalisere modellytelsen.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU), as a part of the degree of Master of Science and the course at the Department of Computer and Information Science (IDI) under supervision of PhD Fellow Nils Barlaug and Professor Jon Atle Gulla.

First of all, I would like to thank my supervisors for their persistent guidance, enthusiastic encouragement and useful critiques of this work are highly appreciated. It has been exciting and challenging to work with this type of industrial research project related to unlocking the potential of active learning and transformer pretrained models (TPLM) for blocking. This deep active learning approach intersects the research fields entity matching, deep learning, and active learning. Thank you Jon Atle Gulla and Norwegian Research Center for AI Innovation (NorwAI) for letting me participate, and be a part of your research projects with Cognite.

In addition, I would also like to thank Nils Barlaug for his grateful help, feedback and an always-open office door for discussions related to deep active learning-based modeling in this thesis.

Jon Steinar Folstad

Trondheim, August 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the motivation, problem outline and goals of the thesis. Then, our contributions, approach, and main results are presented. Lastly, the structure of the thesis is described.

1.1 Background and Motivation

During the last decade, data-generating has boomed due to their primarily distributed way of being produced. Companies of any size, individual users, sensors, and automatic extraction tools have contributed a constantly increasing volume of diverse, heterogeneous, inconsistent, and noisy information. Today, the rise of big data poses new challenges. *Volume* requires techniques to scale to millions of entities, while *variety* calls for entity matching techniques that can cope with schema heterogeneity. One challenge related to big data in entity matching is that more extensive datasets require efficient parallel techniques. More heterogeneity involves unstructured, unclean, incomplete data and diverse data types. Data quality is reduced further due to the frequent use of heterogeneous names, abbreviations, and missing values. Consequently, determining matches between records is difficult because relevant data can be spread over thousands of data sources.

A common obstacle is that data resides in many different systems which do not share a standard data format, common attribute names, or unique identifier between the entities. Working across several data sources is essential to gain value outtakes from the company's data. For exemplifying the problem domain, a classic situation industrial companies can face in their business is the crucial aspect of identifying and establishing the relationship between an object in 3D drawings and its related sensors' time series data. For instance, the 3D drawings showing the pump components might be stored in data source A at a factory such as an oil platform. Their related time-series data from pump components' pressure, temperature, and power sensors are stored in another data source B. The problem arises if these two systems A and B do not have any unique identifier which indicates their relationships. On the other hand, it will be an open question for the reader why the industrial companies have not stored all their data in one data platform to be prepared

for utilizing their data in more applications and decisions. Such applications can predict the optimal combination of pump parameters to minimize the environmental emissions of CO_2 or support the field engineer during pump maintenance with available time series data for the surrounding components belonging to the pump itself. In many industrial companies, the solution has been to manually find and combine related data between data sources. This task can be extremely time-consuming at the factory with potentially thousands of sensors and equipment. It showcases the call for new data integration methods related to industry settings. This problem domain is known as entity matching (EM). It aims to reduce the entropy, leveraging the value of data source silos by identifying which records from two different data sources refer to the same real-world entity. The problem domain of the entity matching system might seem to be solvable and straightforward at first glance. However, the examples of matches and non-matches illustrated in Table 1.1 show the opposite, and it symbolizes only as an example of several challenges the industry is facing in terms of data integration. It is, however, far from nontrivial to systematically make matches out of this panoply of information in data sources at a large scale.

Table 1.1 shows the four possible combinations of matching pairs based on the given records in Table A and Table B. As observed from the table, it can be hard and ambiguous to identify which records refer to the same entity based on the attribute $'name'$ caused by no apparent unique identifier is present. To exemplify, "010-10823-00", "101093600" and "101082300" could be examples of possible identifiers. Unfortunately, some records do not have that types of unique numbers at all. The records can also have several words in common, e.g., "garmin vehicle suction cup", even without being a matching record pair. As seen from the given gold standard, record $A1$ and record $B1$ are labeled as match, which means these records refer to the same real-world entity. These records represent examples of the challenges our proposed approach is facing when all information is stored in one string instead of being categorized into several attributes. This problem is not present in relational databases since both tables A and B would have an attribute column with common unique identifiers indicating any relations between the records.

Today, many proposed entity matching techniques demand very high resources that limit their applicability to large-scale problems, except when utilizing powerful cloud infrastructure. Fussy matching approaches cause it, basically caused by having a quadratic complexity to compare all records to be matched with each other. Multiple blocking and matching algorithms can be applied in combined workflows to achieve sufficient matching quality, which requires more resources to solve the problem. In entity matching, blocking techniques are typically the key component for unlocking entity matching tasks on large and very large datasets, such as industrial datasets.

Our research will focus on the blocking process, also known as the candidate selection step. Many people think entity matching is a binary classification problem without necessarily reflecting on the blocking process as a crucial part of achieving matching. Therefore, it is essential to explain the purpose of blocking and matching. The entity matching workflow is a system consisting of two parts: (1) The *candidate selection step* determining the entities worth comparing, and (2) the *candidate matching step*, or simply matching comparing the selected entities to determine whether they represent the same real-world object or not. While the candidate matching step involves a time-consuming pairwise comparison of entities, the candidate selection step can be divided into two categories

Table 1.1: The illustration represents the entity matching problem in identifying matches from non-matches in the textual public dataset Abt-Buy. (a) Table A shows records in Abt, and (b) Table B shows records in Buy. (c) Table Matches addresses the annotation between the four possible combinations of pairs as the gold standard. The objective of blocking is to collect as many matching examples as possible while removing non-matching examples from the initial set consisting of all possible combinations of record pairs. It is equivalent to reduce the size of Cartesian Product, A x B.

(a) Table A: Abt

id	Name
A1	garmin 010-10823-00 black nuvi 660 vehicle suction cup mount 0101082300
A2	garmin vehicle suction cup mount 0101093600

(b) Table B: Buy

id	Name
B1	garmin suction cup mount 010-10823-00
B2	garmin vehicle suction cup mount

(c) Matches: Abt-Buy

Pid	Name
A1-B1	True
A1-B2	False
A2-B1	False
A2-B2	False

depending on their purpose. First, blocking aims to identify entity pairs that are likely to match, where the candidate set is reduced to only perform comparisons between the most likely matching entities. On the other hand, filtering aims to remove pairs that are guaranteed to no match quickly, resulting in a candidate set containing more certain matching entities. Despite blocking and filtering has a common goal, blocking might operate without knowledge of the matching step and filtering needs because two entities match if their similarity measure exceeds a predetermined threshold[Papadakis et al., 2019].

Hence, the motivation for this work comes from the demand for more efficient blockers on large-scale datasets to achieve a suitable candidate set upfront before performing matching. We hope to unveil beneficial relationships between entities in a low-resource setting by combining transformer pre-trained language models with active learning for blocking. This is also the primary starting point for designing our blocking approach.

This thesis aims to evaluate blocking to cheaply remove obvious non-matches from matches before more detailed and expensive record pair comparisons in terms of matching are performed on the datasets. That said, the first hurdle of blocking has been to understand and

get an overview of the datasets and then subsequently selecting the subset of the data, that is of interest, by using traditional blocking techniques. In terms of machine learning, this has usually involved using researchers or domain experts to handcraft features manually. Therefore, fortunately, it is highly beneficial when deep learning can extract features automatically out-of-the-box. Deep neural networks can learn which features are essential and which ones are less or not important at all. As a result, a suitable combination of features are discovered faster than a human doing feature engineering, especially for complex tasks.

1.2 Research Goals and Questions

The research in this thesis aims to develop a deep active learning-based approach for blocking investigating the combination of using pre-trained language models (TPLM) and active learning strategies. The goal of this thesis is to evaluate if there are any rewards related to combining active learning with TPLMs as a strategy when performing the candidate selection step or so-called blocking. The performance is evaluated by reducing the size of the candidate set while maximizing the pairs completeness (PC) score at a low labeling budget.

Goal *Within the context of transformers for blocking in entity matching, evaluate to what extent active learning can reduce the labeling effort.*

In order to achieve this goal we have to (1) construct a model architecture based on Transformers with respect to ignore non-matching record pairs in the datasets in question from the candidate set, and (2) build the model architecture to enable unlocking the potential of active learning as a strategy to reduce the labeling effort for data-hungry deep learning models, and (3) evaluate the performance related to our proposed approach and identify their challenges. We defined three main research questions that need to be answered.

1.2.1 Application of Transformers

RQ1 *How can Transformers be used to improve the blocking performance in entity matching?*

The first research question addresses how TPLM can be leveraged for enabling efficient hands-off blocking tasks in entity matching. We wish to find a model architecture leveraging TPLM to improve the blocking performance and prepare this blocking approach for active learning strategies.

1.2.2 Application of Active Learning

RQ2 *How does active learning with transformers perform for blocking with respect to labeling effort, i.e. the number of labels?*

Transformers pre-trained language models are data-hungry deep learning models and require more labeled data and training time than classical machine learning models. This research question aims to address how the blocking performance is dependent on the amount of labeled data in a low-resource setting when active learning is combined with TPLM. In this case, a low-resource setting is defined to have a labeling budget within 1000 labeled instances.

1.2.3 Challenges with Transformers and Active Learning

RQ3 *What are the challenges in combining active learning and Transformer pre-trained models (TPLM) with respect to blocking in an entity matching system?*

The last research question addresses the challenges related to using active learning with TPLMs in the blocking step in EM. This aspect is important to better understand how our proposed approach impacts the blocking task, and useful to identify further work in this field of research.

1.3 Research Contributions

This thesis provides two main contributions of the work, and they are as follows:

C1 We present TopKDAL, a blocking approach based on pre-trained language models (LMs) such as BERT. To the best of our knowledge, TopKDAL is the first blocking solution that leverages deep active learning (DAL) with pre-trained Transformer-based LMs to obtain deeper language understanding for the candidate selection step in a low-resource setting. TopKDAL learns progressively from the updated input training sets selected by active learning query sampling strategies within a limited labeling budget of 1000 examples. We evaluate the effectiveness of TopKDAL on five benchmark EM datasets with varying degrees of textual and structured difficulty and domains. Across various real-world datasets, TopKDAL yields an improvement on PC scores by 0.7-3% for imbalanced initial training set and 1.2-1.9% for balanced initial training set over active learning-based baseline sampling randomly. It demonstrates the effectiveness of TopKDAL with DistilBERT as TPLM. Our best performing approaches outperform non-active learning baseline by using an order of magnitude

fewer labels. Imbalanced-P-2 consumes 360-840 labeled examples and Balanced-P-4 consumes 200-700 labeled examples of a labeling budget within 1000 examples to reach or surpass the performance the model achieves by training on the whole dataset. In this case, it considers the hard datasets Amazon-Google, Walmart-Amazon, and Abt-Buy. Towards hands-off blocking in EM, our proposed TopKDAL requires no design decisions from a human and features are learned from the data. Fine-tuning hyperparameters are recommended to optimize the model performance. Input data still needs to be labeled by an human. This approach is applicable independently of the matching step in an entity matching system.

C2 Our experimental results show that a balanced initial seed as a starting strategy can effectively tackle the cold start problem, even when the label budget is limited. With availability using a balanced instead of an imbalanced initial training set, a kick-start is instantiated for the model performance gaining a higher initial PC score as reward. However, the experiments show that the initial difference in PC score between training set distributions becomes a marginal difference after active learning over at most 1000 labeled data. Given an imbalanced starting strategy and Random-P/N, a balanced random sampling of positive and negative examples to mitigate biased learning seems to be competitive with more advanced active learning strategies, which depends on searching a more guided learning by selecting high-confident and low-confident examples.

1.4 Approach

This thesis aims to evaluate how active learning, in combination with TPLMs, can be applied for blocking in entity matching. To the best of our knowledge, we are the first to combine active learning with Sentence Transformers for performing the candidate selection step in blocking.

Our research questions are answered by designing a blocking approach involving two components: (1) Blocking strategy and (2) active learning strategy. Experiments by combining these two strategies are applied to evaluate how the query sampling strategies can reduce the labeling effort for blocking while maintaining the model performance. After each active learning iteration, the Sentence Transformer restarts the model instead of training the transformer model using only newly selected examples. Within a limited labeling budget of 1000 labels, the active sampling strategies try to choose better examples for the model to improve the pair completeness (PC) score and reduce the total time consumption.

The query strategies include existing classical active learning methods, Uncertainty, Partition-2, and Partition-4, in which the two latter is created for deep learning and active learning. Additionally, we test the query sampling strategies High Confident Positives and Negatives, Random Positives and Negatives, and Partition-4 with a positive boost to challenge Uncertainty, Partition-2, and Partition-4. Query sampling experiments are restricted to the two TPLM's DistilBERT and RoBERTa to see how their model size impacts time

usage. Balanced or unbalanced initial training sets are applied in the experiments to test the impact initial class distributions have on blocking performance. The experiments are evaluated on five public datasets, commonly used to benchmark EM systems. The results are benchmarked against the non-active learning supervised baseline, and the active learning-based supervised baseline.

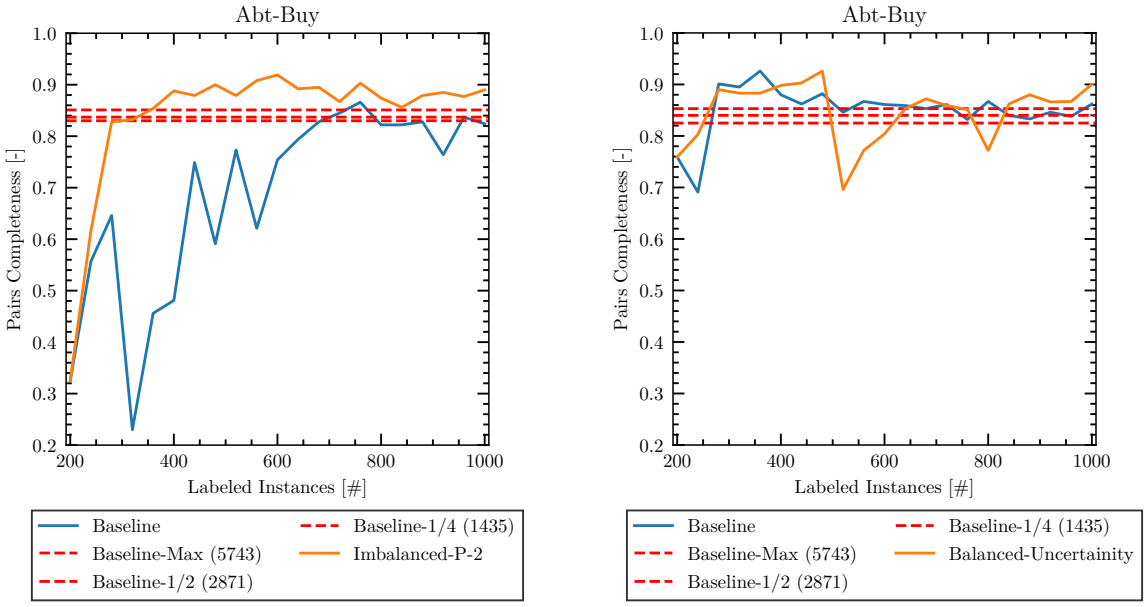
1.5 Results

Figure 1.1 showing active learning strategies yield better results than a supervised Baseline. In our experiments, TopKDAL demonstrated best performance with Imbalanced-Partition-2 and Balanced-Uncertainty to gain the highest PC score across five real-world datasets. After 1000 labeled instances, Imbalanced-Partition-2 and Balanced-Uncertainty outperformed Baseline across the five datasets on average with 0.027 and 0.018, respectively, where Balanced-Uncertainty yielded on average a PC score of 0.952 and outperformed Imbalanced-Partition-2 with 0.012 higher PC score (1.3%). Balanced-Unceratinty consuming an initial 50/50 distribution training set contributing to kick-start the active learning performance and reduce the risk for cold start problems.

On the dataset Abt-Buy, we found that the combinations Imbalanced-Partition-2 and Balanced-Uncertainty outperformed Baseline-Max performance in PC score after approximately 360 labeled instances and approximately 260 labeled instances, respectively. As seen from Figure 1.1, this improvement corresponds to 2.1 % and 1.5 % of total 17223 labeled instances for Abt-Buy. DistilBERT as TPLM reached an iteration time of on average 2-3 minutes. Combined with a limited labeling budget reduced to an order of magnitude fewer labeled examples than the Baseline-Max, TopKDAL proves that the combination of Sentence Transformers and active learning strategies is a valid option to consider for blocking in EM.

The semi-supervised strategy Partition-2 aiming for a balanced set of low confident and high confident examples delivered a significant improvement in PC score over the AL iterations, although it started with imbalanced initial training set 0.144 behind Balanced-Uncertainty. After 20 AL iterations, Imbalanced-Uncertainty ended up only on average 0.012 behind Balanced-Uncertainty, despite for the fact that the automatically labeled high-confidence examples were exposed for incorrectly labeling. A weakness to consider with Balanced-Uncertainty is volatile performance due to the fact it samples low confident examples over AL iterations.

Our approach TopKDAL shows the challenge to mimic the actual similarity between record pairs in the model architecture when combining TPLM with informative active learning query sampling strategies. In our approach, when leveraging the Softmax function to normalize the output of a network to a probability distribution over the predicted output classes, it affects the doubt record pairs negatively. In addition, our work supports a call for more research related to optimizing initial class distributions in the training set, query sampling strategies, pre-trained language models, and hyperparameters properly, as a unified system to maximize blocking performance.



(a) Imbalanced Initial Seed and Partition-2

(b) Balanced Initial Seed and Uncertainty

Figure 1.1: Iterative PC score for (a) Imbalanced-Partition-2 and (b) Balanced-Uncertainty with respect to the amount of labeled instances for Abt-Buy dataset. DistilBERT has been used as TPLM. Baseline-Max, Baseline-1/2 and Baseline-1/4 indicate the PC score after 100%, 50% and 25% of the dataset total size was labeled. On average across the datasets, (a) Partition-2 yielded the best performance on an imbalanced initial seed, and (b) Uncertainty performed best on a balanced initial seed. Both approaches outperformed Baseline-Max after 1000 labeled instances. For Abt-buy dataset, Imbalanced-Partition-2 and Balanced-Uncertainty outperformed Baseline-Max performance in PC score after approximately 360 labeled instances and approximately 260 labeled instances respectively, corresponding to 2.1 % and 1.5 % of the total size of Abt-Buy dataset with 17223 labeled instances. As observed, the PC score differences between the query sampling strategies and Baseline were greater in magnitude when starting with an imbalanced initial seed. However, Balanced-Uncertainty approach has unstable PC score performance over the AL iterations as a weakness.

1.6 Thesis Outline

This thesis is composed of eight chapters. A short description of what each chapter contains is as follows:

Chapter 1 - Introduction The first chapter explains the background and motivation of the thesis, in addition to describing the goals and the underlying research questions this thesis seeks to answer. Research contributions and the approach for proceeding

to solve these questions are presented. Last, the overall results and structured of the thesis are described.

Chapter 2 - Background This chapter presents a theoretical overview of the methods used in this thesis. This involves terminology and concepts related to entity matching, Transformer pre-trained language models, and active learning strategies. In addition, evaluation metrics for evaluating our approach are presented.

Chapter 3 - Related work The third chapter gives an overview of publish papers related to our research in this thesis.

Chapter 4 - Data The published datasets used in the experiments for validating our proposed approach are described here.

Chapter 5 - Method This chapter presents our design choices, proposed approach, the experimental setup, and evaluation metrics for evaluating our approach.

Chapter 6 - Results This chapter shows the results from the the experiments based on our proposed approach and observations among these results.

Chapter 7 - Discussion This chapter discusses the results, comparisons, and interpretations. It also discusses some of difficulties and challenges.

Chapter 8 - Conclusion and Further Work The final chapter concludes our thesis with respect to the research questions outlined in the introduction. Last, thoughts in terms of recommendations and suggestions for further work is presented.

Chapter 2

Background Theory

2.1 Entity Matching

2.1.1 Entity Matching Problem

Entity matching (EM) has been extensively studied for decades since it was first reported in 1946 [Dunn, 1946, Christophides et al., 2020, Christen, 2012a], and it is a crucial process for data integration and data cleaning [Köpcke and Rahm, 2010]. In the literature, the term *entity matching*[1] is often referred by many different names. Consequently, it has resulted in inconsistency as some refer to the same problem, while others are slight variations, specializations, or generalizations of the EM problem [Barlaug and Gulla, 2020].

The problem definition can be derived by defining two data sources A and B. A has the attributes $(A_1, A_2, ..., A_n)$, and each attribute has records which are expressed as $a = (a_1, a_2, ..., a_n) \in A$. Similarly, B has the attributes $(B_1, B_2, ..., B_n)$, and each attribute has records denoted as $b = (b_1, b_2, ..., b_n) \in B$. Furthermore, the problem definition can be divided into finding matches within each data source or finding matches across two data sources. However, for the two entity collections A and B, if a_i and b_i refer to the same real-world entity, they match successfully, and the entity records can be denoted as $a_i \equiv b_i$. In this work, the objective is to find matches across two data sources.

Definition 2.1.1 (Entity Matching)**.** *In entity collections A and B, record pairs match if $a_i = b_i$ refer to the same real-world entity. Matching entities are also called duplicates. The objective of Entity Matching (EM) is to find all matching entities within an entity collection or across two or more entity collections* [Papadakis et al., 2019].

A data source is defined as a set of attributes with records, while a record is a tuple

[1]Entity matching has many similar name variations. EM is also referred to as entity matching, data matching, string matching, fuzzy join, duplicate detection, re-identification, entity resolution, data linking, approximate string matching, similarity join, merge-purge, record linkage, reference reconciliation, fuzzy matching, deduplication, object identification or duplicate identification [Barlaug and Gulla, 2020].

following a defined schema of attributes. Although attributes can have metadata (e.g., a name) associated with themselves, it does not affect the equality between the attributes. So, the tuple of attributes $(A_1, A_2, ..., A_n)$ can be expressed as the schema of the data source A and correspondingly for B. Furthermore, two record sets can have different schemas, and the attributes can have any data type other than strings.

The entity matching (EM) methods can be classified into categories, Clean-Clean EM, Dirty-Clean EM, and Dirty-Dirty EM, given two input entity collections, A and B. In Clean-Clean EM, both A and B are duplicate-free entity collections. In Dirty-Clean EM, A is a duplicate-free entity collection, and B is a dirty entity collection. In Dirty-Dirty EM, both A and B are dirty entity collections. Duplicate-free is present if there is no duplicate record in A and B. Otherwise, it is said to be dirty.

2.1.2 Blocking vs. Matching Problem Domain

Figure 2.1 shows two sets of entities denoted as A and B, assuming two distinct datasets, where there are *1:1*, *1-n*, *m-1*, *n-m*, or *none* relationships between entities in A and B. $M = A \cap B$ is the intersection set of matched entities appearing in both A and B, and $U = (A \cup B) \backslash M$ is the set of non-matched entities appearing in either A or B, but not in both. When putting U and M together, it cover the entire entity space as illustrated in Figure 2.1. In edge cases, when there are two duplicate datasets denoted as A and A', B can be replaced by $'A$. Doing so, entity matching across two duplicate datasets may also call for an upfront blocking process.

First, the non-matching entities appearing in the space U are what the blocking techniques seek to identify and then ignore as much as possible of this space.. In entity matching, the space U is only a collection of non-matching entities. The lower and upper boundary of the space U is derived by Equation 2.1

$$||A| - |B|| \leq |U| \leq |A| + |B| \tag{2.1}$$

Second, the maximum possible number of matching entities are represented to the size of the smaller set of A or B. With respect to performing blocking and matching, the purpose is to keep as must matching entities as possible in the entity space M. In practice, this situation is represented when the smaller set is a proper subset of the larger one, which also results in the minimum number of non-matched entities. Moreover, the lower boundary of M is expressed when no entities appear in both sets, i.e. the minimum number of matched entities is zero. In this situation the number of non-matched entities corresponds to the sum of the entities in both sets. The lower and upper boundary of the space M is expressed formally Equation 2.2.

$$0 \leq |M| \leq min(|A|, |B|) \tag{2.2}$$

To summarize, the blocking algorithms aim to filter out U to simplify the work for the matching algorithms to classify the record pairs within M if and only if $M > 0$.

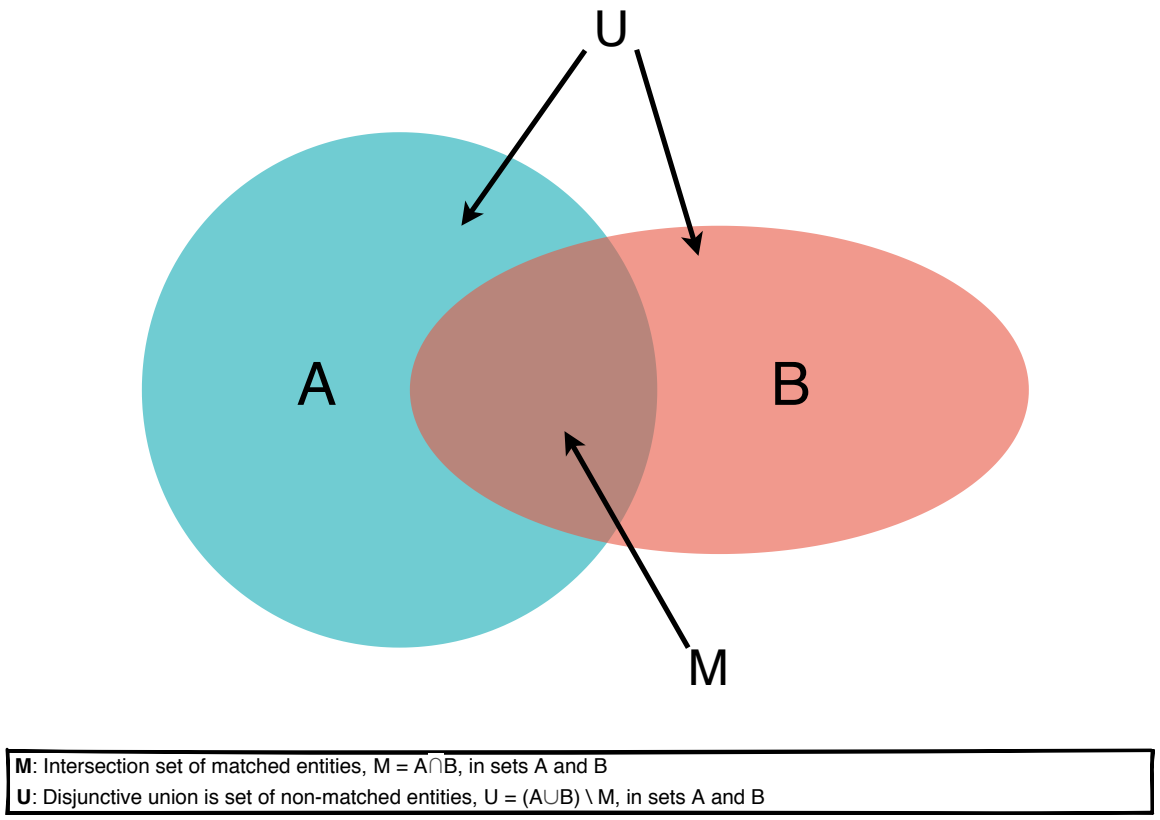


Figure 2.1: Illustration of entity matching problem domain with two sets of entities A and B. Entities appear in both sets are showed in the intersection M, and entities appear in either A or B are showed in the disjunctive union U. The edge case when there are 1-1 and/or 1-n relations across two duplicate datasets A and A', A' can replace B in the illustration. Doing so, the need for blocking is still present, especially for a large-scale dataset.

2.1.3 Time Complexity

In a situation where two data sets, A and B, are to be matched, potentially each record from A has to be compared with all records from B. The number of possible record pair comparisons are equal to the Cartesian product of the size of the two data sets, $|A| \times |B|$. Moreover, when deduplicating one of the two datasets, the number of possible record pairs is $|A| \times (|A| - 1)/2$. In an entity matching system, the performance bottleneck is typically the expensive comparison of attributes between pairs of records. For large datasets, it can often be unfeasible to perform comparisons between all pairs due to the computational complexity of $\Theta(n^2)$.

For instance, the naive brute force approach by performing pairwise comparisons on two datasets consisting of 1000 000 records each results in 10^{12} (1 trillion) record pair comparisons. If each comparison uses $1\mu s$, the computational time is approximately 11,6 days. On the other hand, it is unlikely that all pairs match in these two datasets. By reducing the number of comparisons to 10^9 (1 billion) based on blocking criteria, the computation time can be reduced to approximately 17 minutes if each comparison still uses $1\mu s$. Thus,

the motivation of finding efficient blocker(s) are high in entity matching problems.

2.1.4 Challenges in Entity Matching

Table 2.1 shows some typical challenges faced during binary classification in entity matching. These modified datasets about iPad products are generated from two online electronic product stores. There is only a match between 'A2' and 'B1' among four possible record pairs across datasets A and B. A match indicates records corresponding to the same entity in the real world, while the other three pairs are classified as non-matches.

No common identifier Each dataset has no identifier, often referred to as primary keys, to automatically join rows across the datasets.

Concatenated attributes Dataset A has concatenated several attribute fields compared to Dataset B. To exemplify, the attributes 'Name' and 'Type' in A contains information corresponding to the attribute values for the fields 'Name', 'Model' and 'Type' in B.

No unified schema The datasets can have the same attribute name, but they contain different attribute values. The attribute 'Name' addresses the challenge of which attribute fields should be used to find matching record pairs.

Inconsistent attribute data types Dataset A has defined the 'Price' field as an integer and Dataset B as a float.

Missing / additional information Datasets contain often missing values or additional information making the datasets dissimilar. In this case, the field 'Producer' are used in the schema B, but it is excluded in A. Missing values are referred to as 'NA', observed in the field 'Price' in B. Missing or additional information can result in difficulty to identify a matching record pair.

Quadratic complexity Comparisons between two datasets introduce a Cartesian product of possible record pairs in magnitude. As seen from Table 2.1, there exists only one matching record pairs among four possible pairs. As described in Section 2.1.3, this time complexity increases for each row item added to the datasets. For instance, by increasing the number of examples in each dataset with two records, the number of possible matches increases with 16

Table 2.1: Illustration of entity matching challenges faced between Dataset A and Dataset B regarding iPad products from Apple. Inconsistent schemas and attribute values contribute to difficulties in comparisons between Dataset A and Dataset B for identifying matching record pairs. In this case, the rows A2 and B1 refers to the same iPad model, while the other iPad models have non-matches.

(a) Dataset A

Pid	Name	Price	Type
A1	iPad Air 10.5" Retina display with True Tone 4th gen.	$599	tablet
A2	iPad 12.9-inch Liq. Retina True Tone Pro gen. 5th	$1099	smart tablet

(b) Dataset B

Pid	Name	Model	Type	Producer	Price
B1	iPad Pro	5th gen.	smart tablet 12.9"	Apple	NOK10990.00
B2	iPad mini	5th generation	7.9-inch tablet	Apple	NA

(c) Matches: Dataset A-Dataset B

Pid	Match
A1-B1	False
A1-B2	False
A2-B1	True
A2-B2	False

2.2 Entity Matching Process

In entity matching, there exists many different workflows due to no agreed process about how to perform the step-wise task[Barlaug and Gulla, 2020, Konda et al., 2016]. Figure 2.2 shows entity matching reference model. It is often described as a process consisting of five major steps with corresponding subtasks and subproblems[Barlaug and Gulla, 2020].

The entity matching (EM) process refers to the problem definition introduced in Section 2.1.1, and blocking problem domain, as we investigate in this work, is presented in Section 2.1.2. Typically, the process assumes two data sources as input, but it could be generalized to multiple data sources. A single source, as previously described, can be managed as a special case in entity matching. We denote that this process is machine-oriented, and therefore, it is not involved any iterative human interactions or feedback loops in the EM workflow. However, based on the survey by Barlaug and Gulla [2020], there were not any human-in-the-loop techniques tightly coupled to the process. We therefore introduce the traditional EM workflow as a reference model.

Data preprocessing The first step is data preprocessing, essential to obtain consistent and similar formats across the data sources before they are used in downstream

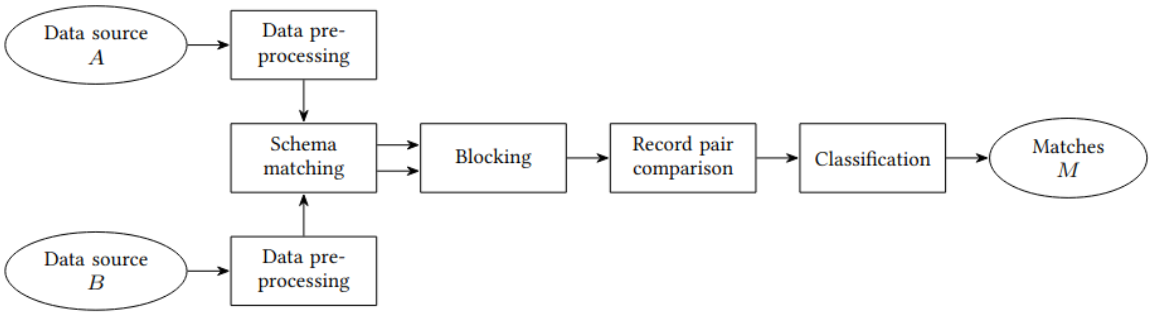


Figure 2.2: The traditional EM workflow can be described with five major steps according to overview by Barlaug and Gulla [2020]. Step 1) Data preprocessing. The data from two data sources A and B are preprocessed with cleaning and transformations to obtain a similar format. Step 2) Schema matching relates semantically similar attributes in the two datasets. Step 3) Blocking is used as a strategy to ignore non-matching record pairs to reduce the quadratic number of possible matching record pairs. Step 4) Record-pair comparison is computes as string metrics for all candidate pairs. Step 5) Classification defines whether a record pair is a match or non-match. Note that active learning with human-in-the-loop and TPLM aspects are not considered in the illustration. At high level, our research is considering the sub-problems and the sub-tasks from data sources to blocking output.

tasks. This step involves transformations such as removing unwanted characters, handling missing values, normalization values, and feature extraction. Preprocessing step needs to be customized based on the domain and the specific data sources. Due to this crucial step for data integration tasks, Barlaug and Gulla [2020] and Christen [2012a] highlight also several other data preprocessing aspects.

Schema Matching Schema matching is the task of identifying semantically related attributes, and which attributes should be compared to one another based on their attribute values and metadata [Barlaug and Gulla, 2020]. This step is often interleaved with the preprocessing step when transforming the data sources into the same schema format. Traditional, schema matching [Rahm and Bernstein, 2001, A. et al., 2017] can be investigated manually, supervised or unsupervised [Barlaug and Gulla, 2020].

Blocking Blocking is used as a strategy to reduce the quadratic number of possible matching record pairs. The objective is to remove as many negative matching record pairs as possible, while achieving a high pairs completeness score due to true positive matches successfully retrieved. In the end, the blocker outputs a candidate set consisting of possible matching record pairs, which often requires upfront to perform the record pair comparison and classification. A more detailed coverage of blocking methods are described in Chapter 3 and it can reviewed in the comprehensive survey of blocking by Papadakis et al. [2019].

Table 2.1 illustrates an entity matching process, in which all combinations of possible record pairs are present. A efficient blocking step would have ignored the record pairs A1-B1, A1-B2, and A2-B2 before a binary classification.

Record Pair Comparison In the record-pair comparison, when the number of candidate record pairs has been reduced to a doable amount, string metrics are computed for all candidate pairs. This pairwise comparison of $(a, b) \in C$ turns typically into a similarity vector S consisting of numerical values, which each value expresses the degree of similarity between records within two specific attributes[Barlaug and Gulla, 2020]. Different types of similarity measures can be applied for each record pair depending on their corresponding data types, such as age, time, and date. Similarity value is often a normalized value ranging between 0 and 1 [Konda et al., 2016]. Clue related to which attributes to compare is often seen during the schema matching step. Barlaug and Gulla [2020] highlight which proposed approaches have used record comparisons up-to-now.

Classification Lastly, binary classification is the last step. It is performed on all pairs in the candidate set to classify the record pairs as match or non-match based on the similarity vector S. Here, if $|S| = 1$, we can use a simple threshold. If $|S| > 1$, it requires more elaborate methods [Barlaug and Gulla, 2020]. In recent years, matching algorithms leveraging handcrafted rules, supervised and unsupervised machine learning approaches, deep-learning techniques, active learning strategies, and crowdsourcing have been used [Ren et al., 2020], in particular to improve the matching performance while reducing the run time. We refer to Barlaug and Gulla [2020] for further explanations related to proposed classification methods.

Table 2.1 shows an example of a binary classification task between dataset A and B, each source consists of two records. The dataset 'Matches' shows the corresponding labels obtained from the binary classification, and it is referred to as M in the reference model. The record pair A2-B1 is classified as a match.

As the purpose of this work is to investigate and improve the blocking performance of combining AL with TPLM, we focus on the blocking step in the entity matching process hereafter.

2.3 Active learning

Active Learning (AL) is natural to investigate as it has gradually received more attention during the recent years due to limits to the applicability of deep learning (DL) methods in low-resource entity matching scenarios [Kasai et al., 2019]. The performance of DL models suffer significantly compared to other traditional machine learning algorithms when only a limited amount of labeled data is available [Kasai et al., 2019]. DL requires instead the data acquisition and high-quality labeling of large datasets, which consumes a lot of manpower and high levels of domain expertise[Ren et al., 2020].

AL aims to reduce the labeling cost by orders of magnitude while retaining the powerful learning ability of Deep Learning (DL) and reaching target performance level accuracy more quickly. For this reason, Deep Active Learning (DAL) has emerged and the breakthroughs of DL have been focused on various fields with the large number of the publicly labeled datasets[Ren et al., 2020]. AL applies the DL model itself for selecting the data from the entire unlabeled dataset, which the model will learn most from. The only challenge is determining what those examples are. The purpose of active learning can be stated as

Definition 2.3.1 (Active learning)**.** *Active learning aims to maximize the performance of the machine learning or deep learning model's gain with respect to minimizing the required labeling of samples from the unlabeled datasets [Ren et al., 2020].*

In EM, active learning can be appropriate because the availability of unlabeled data is high and acquiring labeled data by the Oracle, i.e. a human annotator, is time-consuming or difficult to identify a suitable set of labeled data. In practice, the decision of query each specific label depends on whether the achievement of querying the label is greater than the cost of obtaining that information[Settles, 2009, Meduri et al., 2020].

2.3.1 Query Problem Scenarios

There are many active learning query sampling strategies proposed to select unlabeled instances for different problem scenarios. Settles [2009] categories these active learning (AL) scenarios into 1) pool-based active learning, 2) membership query synthesis, and 3) stream-based selective sampling.

Pool-based active learning [Lewis, 1995] is the most well-known scenario for active learning. In this sampling method, the algorithm tries to select the best query samples based on the evaluation and ranking of instances in the dataset. An overview of the pool-based AL process combined with deep learning model is showed in Figure 2.3. In the pool-based active learning setting, where there is a set of labeled data L and a set of unlabeled data U. The active learner algorithm is often initially trained on a fully labeled part of the training set to determine which instances would be most beneficial to add to the training set L before the model is trained on the updated dataset.

As real-world EM problems, the pool-based active learning is suitable to cover the potential matches in the unlabeled pool while the labeled examples are collected in the labeled set L. Each time the learner evaluates the unlabeled set U, the query sampling strategy selects a set of examples to be labeled by the Oracle before they are added to the labeled set L. The drawback is the amount of memory the method can require [Settles, 2009]. Pool-based sampling can hereafter be assumed if not otherwise is mentioned.

In *Stream-based selective sampling*, also known as selective sampling, the learner scans the set of instances sequentially to perform an independent judgment to decide if each instance in the data stream should query the Oracle to label the instance [Settles, 2009]. In this way, the sampling method assumes that retrieving an unlabeled instance is inexpensive,

while the labeling itself of the instances has a cost. Hence, the learner have to evaluate each instance informativeness measures or compute if the instances are located within the model's region of uncertainty. These instances are ambiguous to the learner and can then be queried. This approach has a natural disadvantage that involves lack of guarantee to stay within the labeling budget.

Membership query synthesis allows the learner to request any unlabeled instances within the input space for labeling. The learner can also generate synthetic data to be labeled by Oracle. In this way, the method is compatible with problems where it is easy to generate a data instance. The downside is that syntactic instances can be hard to label for the Oracle [Settles, 2009].

2.3.2 Active Learning Process

Active Learning is an iterative process used to select a set of examples by using a predefined query sampling strategy. These examples are passed to the Oracles to be labeled. A model is trained to enable prediction scores among the unlabeled data [Settles, 2009, Christen, 2012b, Kasai et al., 2019, Ren et al., 2020, Meduri et al., 2020].

Figure 2.3 shows this human-in-the-loop process. First, active learning starts with a few labeled samples, often known as an initial training set. Then, the model starts training on these labeled data before the unlabeled data is predicted. In each iteration, the Oracle labels the examples retrieved by the query sampling strategy. The newly labeled examples are then added to the set of labeled training data, which is used to train the model in the next iteration [Settles, 2009]. This AL loop continues until a predefined stop criteria are reached. This stop criteria can be either an upper boundary of maximum number of iterations, a performance target, time usage limitation, or no more unlabeled data is available [Settles, 2009].

In the early AL iterations, a challenge can typically be low model accuracy and low performance scores. However, the query sampling strategy can still select unlabeled examples even though it might be some mismatches related to predictions of the record pairs [Shao et al., 2019]. Section 2.4 and Section 5.3 present several different query sampling strategies.

2.3.3 Active Learning Definitions and Concepts

Figure 2.4 introduces the active learning concepts used in our experiments. ***Labeling budget*** is the maximum number of labeled examples in the training set after every AL iteration is finished. This labeling budget defines the upper boundary for the labeling work the Oracle needs to perform, The model aims to achieve target performance of Baseline-Max within that labeling budget, denoted with 1000 labeled instances in the illustration. ***Baseline-Max*** is the non-active learning based approach trained on 100 % of the total training set.

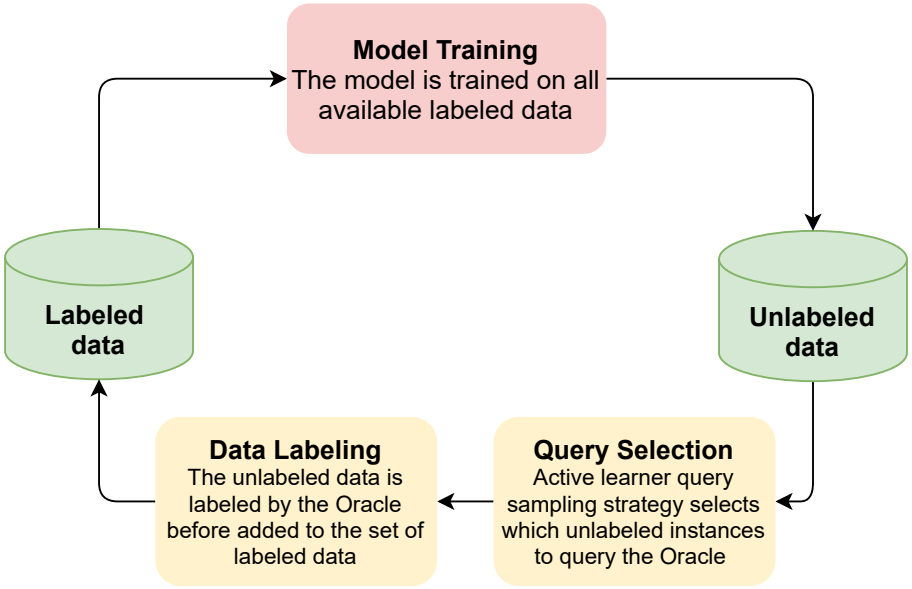


Figure 2.3: Workflow of pool-based active learning combined with deep learning model. First, model trains on a few labeled examples, often known as an initial training set. Then, the model performs prediction on unlabeled data. Next, an AL query algorithm applies to select examples from the unlabeled set to be labeled by the Oracle. The newly labeled data is added to the training set. Lastly, the model restarts and retrains based on the updated labeled training set. The AL loop continues until the stop criteria is reached.

As seen from the figure, the labeling budget is divided into the initial training set and an updated training set. ***The initial training set*** is the labeled data the pre-trained transformer language model (TPLM) is instantiated with, whereas the ***updated training set*** is the training data fed into the TPLM in each AL iteration.

For every AL iteration, the query sampling strategy selects and adds new examples into the updated training set. A ***passive query sampling strategy*** represents the performance of baseline given random sampling, whereas an ***active query sampling strategy*** indicates the performance of hand-picked active learning strategies. Our chosen active learning sampling strategies are described in Chapter 5. The size of the ***initial training set*** has to be fine-tuned properly to require a minimum amount of labeled instances before starting the active learning loop, as illustrated by the black arrows at 200 labeled instances. A suboptimal size and distribution of the initial training set can activate a ***cold start problem***, which means the model yields a performance score at approximately zero on the y-axis. Next, an unstable model might perform unusable probability predictions of the record pairs.

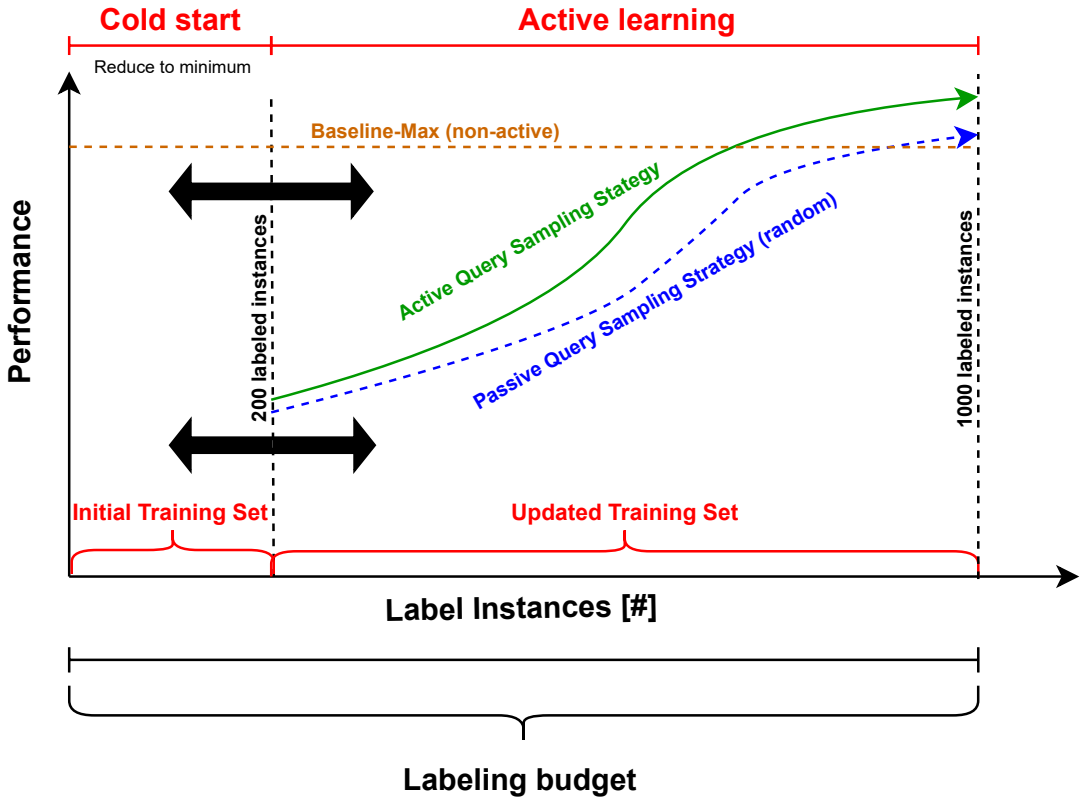


Figure 2.4: Illustration showing the most important AL concepts used in our experiments.

2.4 Active Learning Query Sampling Strategy

Query sampling is an essential strategy for acquiring training data in deep learning applications to achieve better performance with relatively fewer but representative training examples. In practice, it is often infeasible and expensive to obtain large amounts of manually labeled examples [Shao et al., 2019]. To seek to overcome this challenge, active learning has been studied with respect to how to effectively select fewer labeled examples to training set for the deep learning model while achieving similar or greater performance [Ren et al., 2020].

Historically, various active learning strategies have been proposed from different perspectives [Settles, 2009]. It involves uncertainty sampling [Yang et al., 2015], query-by-committee [Seung et al., 1992], and expected model change [Cai et al., 2017]. On the other hand, the research related to DAL methods focuses primarily only on the uncertainty-based methods. One reason is that DL requires least effort to be integrated with the uncertainty-based AL query sampling strategies [Ren et al., 2020]. By leveraging the information from the previous labeled examples, query sampling strategies tries to retrieve the most informative or representative examples from the unlabeled training set by posing queries and then asking labels from an Oracle [Deng et al., 2018, Maystre and Grossglauser, 2017].

Hence, we have also chosen to focus on uncertainty-based AL and informativeness to select unlabeled examples. It seems reasonable as uncertainty sampling[Lewis, 1995] is also the simplest and most commonly used strategy[Settles, 2009]. We argue that the likelihood of successfully unlock AL as a strategy with TPLM for blocking increases if we start with well-known query sampling methods. Furthermore, Ren et al. [2020] believes that uncertainty-based methods will continue to dominate in the future. To date, we refer to Ren et al. [2020] for a comprehensive review of DAL strategies covering other fields than EM. Chapter 5 describes our query sampling strategies in detail.

Unfortunately, which query strategies perform best depend on the datasets being sampled from and the deep learning learning model chosen. In common, none of strategies cover a one-fit-all solution - another challenge faced in active learning. Hence, it is still difficult for a specific task to determine its best-suited one. In the recent years, such limitation has introduced meta-learning algorithms[Hsu and Lin, 2015, Konyushkova et al., 2017].

2.4.1 Uncertainty Sampling

Uncertainty sampling is one of the most used active learning approaches [Lewis, 1995]. It is constructed to select the instances the model is the least certain of as the most informative instances based on probabilistic confidence [Settles, 2009]. The idea is to improve the model by learning more from uncertain labeled instances than certain labeled instances. The former is characterized as difficult instances to learn for the model, while the letter is more easy instances. Softmax can be used to normalize the output from the network to a probability distribution over predicted output classes. The instances with a probability score close to 0.5 are either low confident positives (LCP) or low confident negatives (LCN) dependent on the probability score is above 0.50 or not, respectively. A positive certain instance is represented by a probability score close to 1.0, and close to zero for a negative certain instance. Normally, the terms uncertain and certain instances correspond to low-confidence and high-confidence instances, respectively[Kasai et al., 2019].

The drawback related to uncertainty sampling is the selecting of samples that are similar to each other. These selected samples have often similar features [Yang et al., 2015]. Another issue of uncertainty sampling approaches is that they have no functionality to consider the diversity in the training set, typically present in imbalanced class distribution datasets[Ertekin et al., 2007].

Due to entity matching is a binary classification task, it can be derived that the uncertainty sampling models least confident sampling, margin sampling and entropy are equivalent in nature, while these models are not equivalent for multi-class tasks. Hence, we only present least confident sampling.

Least Confidence Sampling

Least confidence sampling selects the instance with lowest prediction score among its most likely label predictions. Let us define the query method as $\theta(x)$, the function defines how

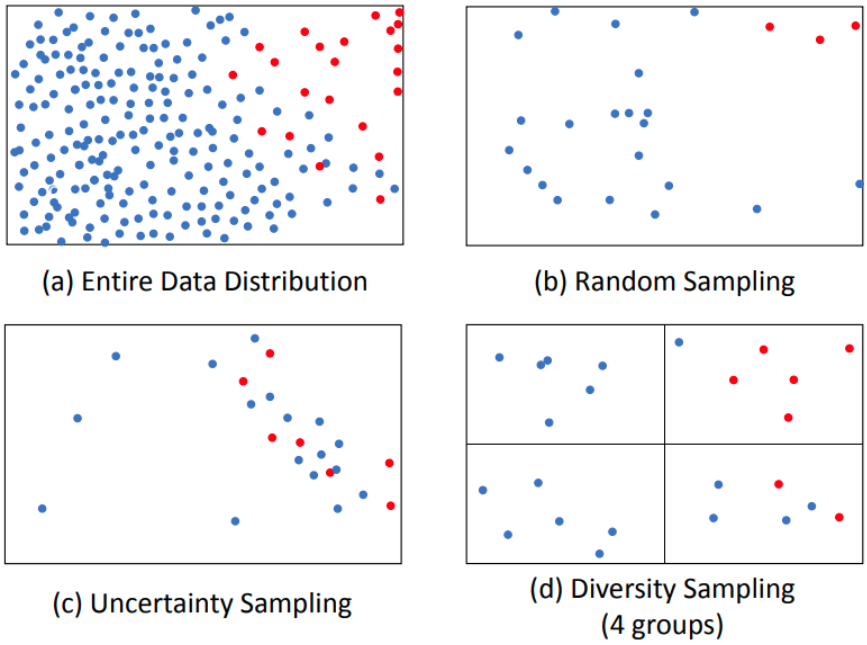


Figure 2.5: Comparison of different query sampling strategies. 24 examples are selected in each of strategies (b) random sampling, (c) uncertainty sampling, and (d) diversity sampling, where red colored dots are matches and blue colored dots are non.matches [Shao et al., 2019].

informative each instance x is. The least confident instances are selected to be labeled by Oracle, which can be derived by

$$\theta^{LC}(x) = \underset{x}{\operatorname{argmax}}\ 1 - P_\theta(\hat{y}|x) \tag{2.3}$$

where $\hat{y} = \underset{y}{\operatorname{argmax}}\ P_\theta(y|x)$ is the class label with highest probability under the model θ. y denotes all possible class labels, and x denotes all unlabeled instances in the unlabeled pool U[Settles, 2009]. The downside related to this query sampling strategy is that the model only leverage the instance with most likely class, the other instances are neglected. More detailed descriptions can be found in Settles [2009].

2.4.2 Diversity Sampling

Diversity sampling is another paradigm in active learning [Xu et al., 2007], and this approach is complementary to uncertainty sampling. It aims to select representative samples based on different types of examples with different features, as illustrated in Figure 2.5. Here, diversity sampling prefers dissimilar examples from different groups of classes in contrast to uncertain samples that are often similar to each other [Yang et al., 2015].

In situations where the deep learning model should select among n different groups, this diversity sampling approaches are beneficial. The downside of considering only strategies that achieve diversity in sampling may lead to increased labeling costs, especially if a significant number of currently selected samples have low information content. On the other hand, uncertainty sampling alone often leads to sampling bias, which means selected sample is not representative of the distribution of unlabeled datasets [Shao et al., 2019].

Several proposed query strategies have additional studied a combination of the uncertainty and diversity of query sampling strategies to seek for a balance of properties between these strategies [Zhdanov, 2019, Yin et al., 2017, Shui et al., 2020, Ash et al., 2020].

2.4.3 Meta-Learners

Meta-learning is an alternative to overcome the obstacles of finding the best query sampling strategies. Driven by the target to automate the query sampling process of active learning approaches, meta-learners seek to learn the best active learning strategy instead of using a particular pre-defined active learning strategy [Shao et al., 2019].

As we chose to focus on building a deep learning model in low resource setting, it is essential to choose correct active learning strategy and meta-learning algorithms seem to be beneficial. However, the drawback of these generalized meta-learning models is that they still require sufficient training data, whilst active learning typically starts with fewer labeled samples before it gradually adds more labeled samples over the AL iterations [Shao et al., 2019]. Consequently, a small number of labeled samples are applied to train a meta-learning model at the beginning, which unfortunately increases the probability of poor model performance in terms of instabilities and overfitting [Shao et al., 2019]. Those aspects were not wanted in our proposed approach and integrating meta-learning algorithms were left to further work.

2.4.4 Challenges of Active Learning

The purpose of active learning (AL) seems as a relevant strategy by reducing the labeling cost while maintaining the performance at a similar level. At the same time, there are some challenges related to the AL [Ren et al., 2020].

Processing pipeline inconsistency. There is inconsistency in the processing pipeline between AL and DL. Many AL methods are based on training of classifiers, and the query strategies are primarily utilized with respect to fixed feature representations. In deep learning, on the other hand, feature learning and classifier training are jointly optimized, and it should not be treated as two separate problem as explained in Wang et al. [2017].

Choice of query sampling strategy. One of the main challenges of AL is how to select the best query sampling strategy to apply. As a consequence, Ren et al. [2020] states that *"meta-learning algorithms for active learning are emerging as a promising paradigm*

for learning the best active learning strategy". However, the current meta-learning based active learning approaches still require a sufficient training set, in contrast to the nature of active learning which typically starts with a smaller number of labeled examples. Thus, poor performance in terms of instabilities and overfitting has been observed by using meta-learning model when trained on that smaller number of labeled examples [Ren et al., 2020].

Time usage labeling vs. AL iteration time. An Oracle is an essential key to unlock the potential of AL at all. A person needs to act as an Oracle, and in many situations the person is also replaced by a domain expert if the domain-specific dataset is challenging. On the other hand, the Oracle has a waiting time corresponding to the iteration time from the model start training until the query sampling strategy has selected the new samples passed over to the Oracle. Consequently, there is a trade-off of how long the iteration time can last and which training set size, hardware and DL model can be put together as combinations.

Implementation complexity. A DAL approach for blocking needs to be divided into two parts, namely, DL model training method and the AL query strategy on unlabeled dataset. Currently, there is no framework released to fill the gap between these parts for blocking. Regarding the AL query strategy, it is required to manage the input and output from the Oracle, and the support of query sampling strategies. The Oracle needs a user-friendly interface to label the examples for fulfilling having human-in-a-loop. Next, the new labeled examples have to be inserted into labeled training set L, and the DP model is retrained on the updated training set. Consequently, a DAL approach requires much more effort to configure compared to standard supervised learning. In supervised learning all the training data is initially labeled and used to train the model corresponding to one-step AL loop.

Feature extraction. In deep learning it is not needed to craft meaningful feature extraction according to many machine learning models. Feature Extraction is usually quite complex and requires detailed knowledge of the problem domain. DL with TPLM can learn an implicit representation of the raw data directly on their own, and the domain experts' work is therefore reduced to conducting labels on the examples in the AL iterations. The challenge is related to explainability of the DL model, namely, how to interpret whether the examples are matching or non-matching record pairs [Barlaug and Gulla, 2020].

2.5 Deep Learning

In the deep learning (DL) field, many architectures exist, and they are all based on Artificial neural networks (ANN) with specific optimizations for certain problems. Recurrent Neural Networks (RNN) are built for sequential data and are popular for Natural Language Processing (NLP). With the rise of deep learning in NLP, several works introduced new techniques such as word embeddings and attention to entity matching [Mudgal et al., 2018, Ebraheem et al., 2019]. One of the advantages of such deep learning models is no need for handcrafted features. However, it also requires a large amount of data to optimize a massive number of parameters in the network [Ren et al., 2020].

Recently, Zhao and He [2019] proposed a transfer-learning approach to entity matching (EM), which leverage pre-trained EM models from large-scale production knowledge bases. The next year, Brunner and Stockinger [2020] applied transformer architectures for entity matching. As described in Chapter 3, several EM papers have leveraged the advantages of transformers for the matching step and achieved state-of-the-art results. However, less attention has been on using transformers for doing the blocking. In our work, Transformers are applied in the experiments to solve the blocking problem in entity matching.

We will only focus on relevant DL aspects related to our performed experiments to limit the scope in the following sections.

2.5.1 Recurrent Neural Networks and Long Short-Term Memory (LSTM) Challenges

Recurrent neural network (RNN) architectures have dominated the the Natural Language Processing (NLP) field until Transformers were introduced [Brunner and Stockinger, 2020]. The most popular proposed approaches have been seq2seq [Sutskever et al., 2014, Brunner and Stockinger, 2020] and encoder-decoder architectures [Cho et al., 2014, Brunner and Stockinger, 2020], which consists of one RNN for the encoder and one RNN for the decoder. Even though RNN seems to be suitable for building representations of sequential text data, Long Short-Term Memory (LSTM) was introduced to mitigate several limitations.

In RNN, it is difficult to utilize the parallelization computation on GPUs caused by input data needs to be passed sequentially processed one after the other, i.e. the inputs of the previous state is needed to make any operation on the current state. This obstacle makes it difficult to take benefit of modern hardware, which results in longer training time than a simple feed forward network [Brunner and Stockinger, 2020]. In addition, RNN has a sub-optimal transferring of knowledge from the source sentence to the target sentence, also known as the bottleneck problem [Bahdanau et al., 2016]. Relevant information is lost when it is passed through a recurrent neural network in long sequences of recurrent connections. Lastly, RNNs struggles to retain longer term dependencies. This behavior is due to the Vanishing Gradient problem, and it can cause problems when the early parts of the input sequence contain important contextual information. In addition, long range dependencies are hard to learn with RNNs, even though LSTMs and GRU (gate recurrent unit) architectures should tackle it. Vaswani et al. [2017] described this effect by the path length between two tokens, e.g. two words in a sentence, which is the number of steps the signal has to flow through the network. The longer the path, the harder to learn a dependency.

To the rescue to solve the problem of Vanishing gradient and Long term dependency in RNN, Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] networks were proposed. It is a modified version of recurrent neural networks, which makes it easier to remember past data in memory. On the other hand, LSTM also struggles when sentences are too long due to the challenge of keeping the context of a word that is far away from the current word.

These challenges have been the motivation of the development of transformers, in particular parallelization of sequential data. With RNN encoder, a sentence is passed one word after the other. The current word hidden state has dependencies in the previous words hidden state. The word embeddings are generated one time step at a time. With transformer encoder, on the other hand, there is no concept of time step for the input data. All words of the sentence are determined by the word embeddings simultaneously [Hernández and Amigó, 2021].

2.5.2 Attention Mechanism

To overcome the bottleneck problem, [4] and [70] proposed a new technique called attention. Attention was introduced as a mechanism to master the bottleneck problem [Bahdanau et al., 2016, Vaswani et al., 2017]. Vaswani et al. [2017] proposed Transformer architecture that eliminates sequential processing and recurrent connections. It relies only on self-attention mechanism to capture the global dependencies between input and output. Transformer architecture achieved significant parallel processing and shorter training time without any recurrent component [Hernández and Amigó, 2021].

Attention involves answering what part of the input data is important to focus on. The analogy to understand self-attention is the brains' ability to understand how much each word is related to every word in the same sentence. This is represented in the attention vector and it is computed in the attention block. For every word we can have an attention vector generated, which captures the contextual relationships between words in the sentence. Each time the model predicts an output word, it only uses parts of an input where the most relevant information is concentrated instead of an entire sentence [Hernández and Amigó, 2021].

To exemplify, a word can be represented as a weighted combination of the words in its proximity. Brunner and Stockinger [2020] explain self-attention by using the word "it" and the sentence *"The animal did not cross the street because it was too tired"*. For the human brain, it is intuitive that the word "it" in a sentence can be represented by the words "The", "animal", and "street" in the same sentence. When the attention mechanism is applied, a language model can be trained to pay attention to relevant words in its proximity accordingly [Brunner and Stockinger, 2020].

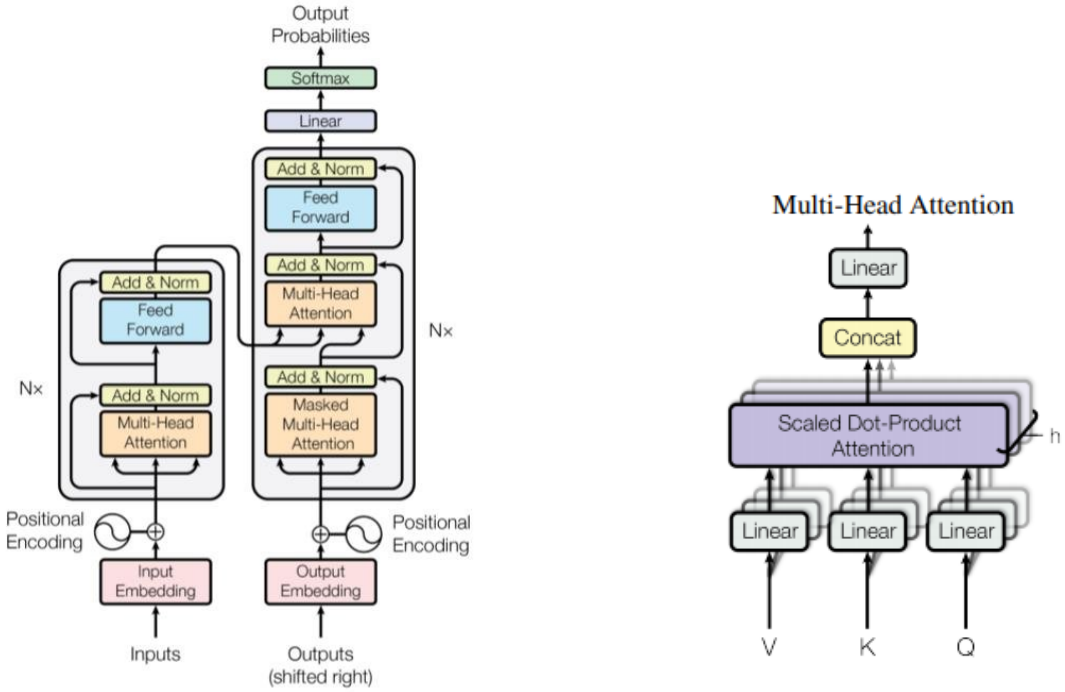
2.5.3 Transformers

Brunner and Stockinger [2020] was the first to approach entity matching with modern transformer architectures. In this architecture, the recurrent connections are redundant and can be removed. It also removes the computational bottleneck caused by the RNN's long backpropagation path. In this way, it allows for a much higher degree of parallelization and faster training. There are several advantages related to using transformers. It reduces the computational time when using pre-trained models instead of doing heavy training on low resource hardware. There are also more than 2,000 pre-trained models

published to transfer the learning ability from. In addition, the transformers utilizes attention mechanisms to overcome the drawback of information loss caused in previous approaches.

As input, the transformer applies a sequence of tokens. The first obstacle is to convert the sentences into relevant and useful sequence of tokens to be ingested into the model architecture of the transformer. There are two common modes to fine-tune a transformer, either paired mode and single mode [Jain et al., 2021]. While a paired mode is typical used for matching, single mode is often applied for blocking applications. For each token the transformer can output a fixed dimensional contextual embedding. The hundreds of million parameters used in Transformer are pre-trained using large amounts of unlabeled text corpus. In turn, each token is assigned an embedding that captures its semantics in the context of the current sentence. These highly contextual embeddings have demonstrated robustness to spelling mistakes, and abbreviations, and provide state-of-the-art performance on dirty datasets for entity matching (EM) tasks [Brunner and Stockinger, 2020, Li et al., 2020]. This improvement is achieved by inserting a task-specific layer on top of the transformer to obtain a fine-tuned EM task-specific objective.

Vaswani et al. [2017] introduces the model architecture Transformer build only with attention mechanisms. The challenge of the previously approaches are described in Section 2.5.1. Corresponding to Long Short-Term Memory(LSTM), Transformer is also an architecture to transform one sequence into another sequence using encoder and decoder. However, Transformer differs from the previous models because attention mechanisms can completely replace the recurrent network modules like GRU and LSTM[Brunner and Stockinger, 2020].



(a) Transformer **(b) Multi-Headed Attention**

Figure 2.6: Transformer architecture as illustrated by Vaswani et al. [2017].

In Vaswani et al. [2017], there can be multiple encoders and decoders stacked on top of each other. Encoder and decoder consists of $N = 6$ encoder blocks followed by the same number of decoder blocks. A encoder block and decoder block are represented inside Nx module illustrated to the left and right in Figure 2.6a, respectively.

In the encoder, it can be seen that each layer of those N=6 identical layers has 2 sub-layers. The first sub- layer is a multi-head self-attention mechanism, and the second is a position-wise fully connected feed-forward network [Vaswani et al., 2017]. There are residual connections around each sub-layer and a normalization layer to ensure stable training. The output data has the same dimension model of 512 for each sub-layers.

The decoder, on the other hand, consists of the 3 sub-layers masked multi-head attention, multi-head attention and a feed forward layer. These sub-layers are illustrated to the right in Figure 2.6a. Each sub-layer has layer normalization. Each decoder block is mostly identical to an encoder block. The only significant addition is a masked multi-head attention. The masked multi-head attention is used to mask the next token in the sentence to prevent the model from looking at the future word. Then, the model can predict the next word without looking at the original text word in the sequence. If not doing so, it would not be any learning, it would only output the next word.

In both the encoder and the decoder, the multi-head attention layer uses scaled dot product attention as a core of performing complex computations. This multi-head attention is showed in Figure 2.6b. All parameters of this multi-headed attention mechanism are learned through standard backpropagation. Feed forward neural network is applied to every one of the attention vectors. In practice, these feed forward nets are used to transform the attention vector into a form that is digestible by the next encoder block, decoder block or a linear layer, as showed in Figure 2.6a. At the end of the decoder, a Softmax and linear layer is added on top of the decoder. Here, the number of neurons in linear layers is equal to the number of words in the output sentence [Vaswani et al., 2017].

Raw text cannot be used directly, hence, the input and output are passed through the embedding layer to learn context-independent word embeddings for each token. Then, the positional encoding is concatenated to the embedded representation of n-dimensional vector space to give each word positional information between the words and sentences. This positional encoding are added after input embedding for the encoder and after the output embedding for decoder to remember the token's order in the sequence.

The downsides related to transformer architecture are limited of only handling fixed-length text strings. The raw text data needs to be split into a defined number of segments before being fed into the model as input. While the positional embedding is periodic and the multi-headed attention can be computed on any sequence length, Transformer models generally struggle when encountering sentences longer than those seen during training. The reason is that since it has never seen such long-term dependencies before, it cannot represent them correctly, and therefore, it underperforms [Dai et al., 2019].

2.5.4 Pre-trained Language Models

Transformer based Pretrained Language Models (TPLM) involves typically pre-trained learning models such as BERT [Devlin et al., 2019], DistillBERT [Sanh et al., 2020], XLNet [Yang et al., 2020], and RoBERTa [Liu et al., 2019]. These pre-trained learning models have demonstrated to work well on various domains and EM tasks [Brunner and Stockinger, 2020]. On the other hand, there exists other language models such as OpenAI's Generative Pretrained Transformers (GPT). GPT-3, an gigantic autoregressive language model pre-trained LM with an enormous 175 billion parameters, is a unidirectional language model unlike BERT models [Brown et al., 2020]. Currently, no published EM works have evaluated language models such as GPT-2 and GPT-3. However, GPT-3 incurs practical inconvenience caused by the model size and significant compute resources that is required [Brown et al., 2020].

Bidirectional Encoder Representations from Transformers (BERT) is a multi-layer bidirectional transformer encoder, pre-trained on 16 GB data consisting of 3.3 billion words from Wikipedia and BooksCorpus. BERT use Transformer, an attention mechanism that learns contextual relations between words in a text. Transformer has two separate mechanisms, an encoder that reads the text input and a decoder that produces a prediction for the task. In BERT, on the other hand, only the encoder mechanism is required because the purpose is to create a language model. During pre-training, the model is trained on unlabeled data over different pre-training tasks. Then, BERT model is fine-tuned by first initializing with the pre-trained parameters before the parameters are fine-tuned to a task-specific objective based on labeled data[Devlin et al., 2019].

Historically, the one-directional language model approaches, like LSTM, were limited to have the ability of reading text input sequentially, either left-to-right or right-to-left, that also could bias tokens towards a certain meaning until a complete sentence was reached. The introduction of Transformers unlocked the potential of reading in both directions at once so-called bidirectionally[Devlin et al., 2019]. BERT takes advantage of this bidirectional technique during pre-trained when using Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). Language Modeling is the task of predicting the next token based on a given sequence of tokens. In masked language modeling, a percentage of input tokens are randomly masked with a [MASK] token, and it is only those masked tokens that are predicted bidirectionally. In this way, the model learns to reconstruct the original data using the entire sequence of tokens. As the input representation for the encoder, the sequence can be a single sentence or a pair of sentences separated by the separator token [SEP] and starting with a special classification token [CLS]. In addition, positional embeddings are used to remember the position of a token in the sentence.

In BERT, a Next Sentence Prediction (NSP) language task is added to the training process. In this step, BERT is trained to perform Next Sentence Prediction tasks that require an understanding of the relationship between sentences. The model receives two sentences A and B as input, and it has to predict if the sentence B is the next sentence for A [Devlin et al., 2019].

RoBERTa introduces improved performance by using the more training data combined with correct hyperparameters [Liu et al., 2019]. They showed that BERT was signifi-

cantly undertrained and proposed minor changes to improve performance to match or exceed existing models. To exemplify, RoBERTa yields 2-20% improvement over BERT by increasing the training data to five English-language corpora with a total size of over 160 GB of text, 16 GB BERT and 144 GB additional data. In addition, Liu et al. [2019] demonstrated that longer training duration and the additional data led to the better performance. They also found that the learning rate was increased when the batch size was increased above 256, which was BERT's batch size. Another important changes to improve the performance were to remove the next sentence prediction (NSP) loss and to use the model input maximum size of 512 tokens, i.e. full attention span, during pre-training to better learn long-range dependencies [Brunner and Stockinger, 2020].

DistilBERT represents a BERT distillation, a smaller, cheaper and faster pre-trained language model than BERT instead of aiming at improving the model results itself[Sanh et al., 2020]. There are several approaches to reduce the model size with respect to maintain the performance approximately unchanged: 1) Quantization, 2) weights pruning, or 3) distillation. DistilBERT has chosen knowledge distillation, where a smaller model is trained based on learning from the original model behaviorSanh et al. [2020]. In comparison to the original BERT model, DistilBERT has only 3% degradation from BERT at the same time reducing the model size by 40% and improving to be 60% faster.

2.6 Evaluation Metrics

We are interested in the following performance measures to evaluate our proposed approach. Once the entity blocker algorithm has been applied, an approach for evaluating the performance of blocking schema s is needed. There are several ways to evaluate the entity blocking algorithm [Christen, 2012b, Papadakis et al., 2015, Shao et al., 2018, Papadakis et al., 2019]. First, the blocking performance can be characterized by its effectiveness and its efficiency[Efthymiou et al., 2020].

Definition 2.6.1 (Effectiveness)**.** *Entity matching aims to achieve a high-quality match result with respect to recall and precision. This means all real corresponding entities and no others entities should be included in the result. In blocking, this quality measure is measured by using Pair Completeness (PC) and Pair Quality (PQ).*

Definition 2.6.2 (Efficiency)**.** *Entity matching aims to improve the efficiency of EM process by adopting blocking algorithms to reduce the search space. EM should be fast even for voluminous datasets. In blocking, this quality measure is measured by using the reduction ratio (RR).*

Before defining the blocking performance quality measures, it is required to define the terminology. A pair of tuples that are placed into the same block is called true positive (TP), if it refers to a match. Otherwise, it is defined to be false positive (FP). Furthermore, a pair of records is called a false negative (FN) if it refers to a match while the record pairs are placed into two different blocks. Thus, TP(s), FN(s), and FN(s) are denoted to the numbers of true positives, false positives, and false negatives in the blocks generated by s, respectively. Further, the quality measures of blocking algorithms are defined and described.

Definition 2.6.3 (Blocking Performance)**.** *Given block collections A and B, blocking aims to cluster similar entities into a block collection bc such that PC(bc), PQ(bc) and RR(bc) are simultaneously maximized.*

Reduction Ratio (RR) quantifies the extent to which the blocking scheme reduces the number of candidate pairs. In other words, it indicates the relative reduction of the comparison search space. The quality of the reduction is not taken into account in RR, meaning how many record pairs from the symmetric difference U, $A \triangle B$ and how many from matches from intersection M, $A \cap B$ are removed by blocking[Christen, 2012a]. It is illustrated in Figure 2.1. RR can be expressed by

$$RR = 1 - \frac{N_b}{|A| \times |B|}, RR \in [0, 1] \tag{2.4}$$

where N_b is the number of record pairs produced by a blocking algorithm, meaning the number of record pairs not removed by blocking. The term $|A| \times |B|$ is the Cartesian product or the total number of comparisons.

An RR close to 1.0 indicates that candidate set has been significantly reduced, while an RR close to 0.0 indicates that the reduction achieved by the blocking scheme was small. RR can be computed although the ground truth of the datasets are unknown.

Pair Completeness (PC) of a blocking scheme s is the number of true positives TP(s), meaning TP(s) + FN(s), which measures the rate of true matches remained in blocks. PC can be calculated by using

$$PC = \frac{TP(s)}{TP(s) + FN(s)}, PC \in [0, 1] \tag{2.5}$$

where TP(s) is the number of true positives and FN(s) is the number of false negatives in the block. Pair completeness can be seen as being analogous to recall. The PC value is between 0 and 1, which is directly proportional to the effectiveness of the algorithm. The ground truth of the datasets are required to compute PC.

Pair Quality (PQ) of the blocking schema is the number of true positives TP(s) divided by the total number of record pairs that are placed into the same block, meaning TP(s) + FP(s), which measures the rate of true positives in blocks. PQ can be expressed by

$$PQ = \frac{TP(s)}{TP(s) + FP(s)}, PQ \in [0, 1] \tag{2.6}$$

where TP(s) is the number of true positives and FP is the number of false positives in the block. The ground truth of the datasets are required to compute PQ. A high PQ value means blocker is efficient and generates mostly true matched record pairs. On the other hand, a low PQ value means a large number of non-matches are also generated.

F_1-score is defined as the the harmonic mean of pairs completeness (PC) and pair quality (PQ) [Simonini et al., 2016, Christen, 2012b]. This metric can be expressed by,

$$F_1(bc) = \frac{2 \times PC(bc) \times PQ(bc)}{PC(bc) + PQ(bc)} \tag{2.7}$$

This quality measure can be useful to compare block collections that present different values of both PC and PQ.

Unfortunately, there are a trade-off between the quality measures for selecting blocking schemes caused by they are often competing. For instance, PC and PQ are negatively correlated in many applications, as well as RR and PQ [Christen, 2012b]. There are also a trade-off between RR and PC, in other words, between the number of removed record pairs and the number of missed true matches. As a consequence of no blocking algorithms are perfect, the blocking process will remove record pairs from M, both true matches and true non-matches. Several factors such as dirty data, misspellings, natural variations, synonyms, missing values, etc, are affecting the quality measures. Furthermore, it is important to recognize that blocking will also influence the quality measures used in the matching process.

Shao et al. [2018] states there are different blocking schemes being preferred in different situations. For instance, a crime investigation aims to match individuals to a large databases of people, a high PC would be desired to increase the likelihood that potential criminals are included for investigation. On the other hand, in public health studies aiming to find matches corresponding to a patient with certain medical conditions, a high PQ is required to only include patients that do have the medical condition under study [Christen, 2012a].

Chapter 3

Related Work

In this chapter, we present some previous works relevant to our thesis. This thesis concerns the intersection of several distinct research fields, deep learning (DL), active learning (AL), and Entity Matching (EM). Hence, there are few previously published works related to our thesis as a whole. In blocking, Deep Active Learning (DAL) is not covered in the research to perform candidate set selection, except for the recently published work by Jain et al. [2021]. Currently, the proposed Ditto [Li et al., 2020], SentenceBERT [Reimers and Gurevych, 2019], DTAL [Kasai et al., 2019], and DIAL [Jain et al., 2021] are the most relevant work to our approach. However, several approaches, such as Zhang et al. [2020] and Ebraheem et al. [2019], suffer from not reporting their blocking performance results on the EM benchmark datasets presented in Chapter 4, which in turn would define a Baseline for comparing our proposed approach against. Ditto and DIAL focus instead to evaluate their approach as a unified entity matching system with a blocking-matching combination to achieve target performance with respect to F1-score [Li et al., 2020, Jain et al., 2021].

At a high level, these works can be separated into approaches that 1) use deep learning for blocking [Zhang et al., 2020, Ebraheem et al., 2019], 2) EM work leveraging TPLM as a strategy [Reimers and Gurevych, 2019, Li et al., 2020, Jain et al., 2021], and 3) EM approaches that unlocking the potential of active learning as a strategy [Kasai et al., 2019]. These three categories of related work will now be accounted for in turn.

3.1 Traditional Blocking Methods

Traditional blocking methods are of interest to better understand which blocking approaches have been extensively studied over several decades. Previous work solutions have tackled the blocking problem by proposing various methods such as handcrafted blocking rules, unsupervised clustering, crowdsourcing and machine learning [Christen, 2012b, McCallum et al., 2000, Fisher et al., 2015, Wang et al., 2016, Li et al., 2020].

In practice, a key-based blocking methods with human effort involvement are still mostly

used. These techniques use a key to partition the entities into separate blocks. This approach is beneficial to restrict matching to only the entities in the same block [Christen, 2012b]. Köpcke and Rahm [2010] categorizes traditional blocking algorithms between overlapping and disjoint methods. *Overlapping method* is determining overlapping blocks of entities. These methods can result in an entity to be matched to multiple blocks, in which leads to increased pair completeness compared to disjoint methods. These methods lead also to increased overhead. *Disjoint methods* build mutually exclusive blocks by assigning each entity to one block. It is typically implemented by using sorting or hash-based on the key. In these blocking methods, a sub-optimal key results in overselection of non-matching record pairs, implying decreasing efficiency. The situation is even worse if the particular key is sorting out the true matching record pairs, and as a result, it increases the fraction of non-matches in the candidate set C. The key is determined manually or semi-automatically based on the datasets. Traditional blocking methods also involve q-gram blocking [Papadakis et al., 2015], and sorted neighborhood [Hernández and Stolfo, 1995].

Another blocking strategy is known as filtering, which apply simple string similarity metrics and a threshold value to remove negative matches. An example is to use Jaccard similarity with a threshold value of 0.4.

Some other methods, MinHash blocking [Liang et al., 2014] is used to obtain a fuzzy match on attributes. However, it often results in many unaffordable candidate pairs. Meta-blocking [Papadakis et al., 2014a,b, 2016a, Simonini et al., 2016] tries to reduce Pair-Entity ratio (P/E) to achieve a small ratio of the number of candidate pairs to the number of entities by introducing, between blocking and matching, extra steps to prune the candidate pairs. However, their proposed approach is orthogonal to our work. We refer to Papadakis et al. [2016b, 2019] for a comprehensive comparison of existing blocking and filtering methods. Additionally, there is still a high demand for democratizing blocking by reducing the human involvement in labeling data, find correct blocking functions, performing feature engineering, and tuning parameters [Ebraheem et al., 2019].

3.2 Deep Learning for Blocking

Deep Learning has been used to tackle the blocking task in entity matching [Zhang et al., 2020, Ebraheem et al., 2019]. In particular, DeepER and AutoBlock are of interest to better understand what Ditto improved in their work.

Ebraheem et al. [2019] proposes DeepER, a system that needs less labeled data by using the knowledge of matching record pairs. They have designed an approach that considers both syntactic and semantic similarities based on distributed representation vectors (DR), a fundamental concept in deep learning (DL). Compared to the traditional blocking methods, feature engineering and parameter tuning are not required. Ebraheem et al. [2019] use recurrent neural networks (RNNs) with long short term memory (LSTM) hidden units to encode each record to a DR vector. For each record in the dataset A, their blocking approach retrieves the candidates with the highest cosine similarity scores from the dataset B by using an approximate nearest neighbor search on vector representations.

In this way, the candidate set is reduced to including records with their approximate nearest neighbors. In addition, they propose a locality sensitive hashing (LSH) based blocking approach that consider all the tokens with all attributes of a record to produce smaller blocks.

Zhang et al. [2020] introduced AutoBlock, it improves deep learning-based EM models by pre-training the EM model on an extra task of entity type detection. In contrast to DeepER, they implement multiple alignment layers for token-level, attribute-level, entity-level comparisons to incorporate finer-grained similarity calculation between records. AutoBlock assumes some knowledge about the data beforehand, and therefore strong attributes are used, such as model number for products to produce labeled data. Both AutoBlock and DeepER use Locality Sensitive Hashing (LSH) for retrieval.

A weakness related to AutoBlock is that it requires insight about the datasets to identify strong attributes, essential to produce labeled data. Advantageous, Transformers do not necessarily need to have any such assumptions beforehand to start learning the transformer network. Hence, an initial training set can instead be instantiated randomly, and an Oracle are labeling these instances as either matches or non-matches.

Lastly, Zhang et al. [2020] and Ebraheem et al. [2019] suffer from not reporting their blocking performance results on the EM benchmark datasets presented in Chapter 4. This leads to the difficulty of evaluating their weaknesses and blocking performance directly with our proposed approach and experimental results. In general, it is a call for establishing a comprehensive benchmark survey of blocking methods based on these EM benchmark datasets and a belonging test framework with proposed blocking methods to date, suggested as further work in Section 8.2.

3.3 Deep Learning with TPLM for Blocking

Li et al. [2020] investigates a similar approach for blocking as used in DeepER in their proposed solution called Ditto in a high-resource setting. Ditto is of interest and related to our work due to its application of TPLM for blocking. Ditto's solution consists of two high-level components to leverage of pre-trained language model (TPLM), encoding each record with Sentence-BERT [Reimers and Gurevych, 2019], and similarity search. A pre-trained language model (TPLM) is fine-tuned on labeled data according to Sentence-BERT [Reimers and Gurevych, 2019]. By fine-tuning the TPLM on labeled data, they obtain an embedding vector representing each record in Dataset A and Dataset B from the trained transformer model. Next, the candidate set is selected by doing a similarity search of likely matching pairs of records, retrieving the top-k approximate nearest neighbors by using blocked matrix multiplication [Abuzaid et al., 2019]. Li et al. [2020] reports that Ditto reduces the number of likely record pairs in the candidate set from 10 million to 0.6 million. A basic blocking method based on heuristic rules instead obtained 10 million record pairs. As a model assumption, they simplify their evaluation to only involve similarity search according to match each record with $k = 10$ to achieve the top ten most similar records in the candidate set. Additionally, Li et al. [2020] claim that their blocking approach in "*Ditto reduces the total time for matching from 6.49 hours to 1.69*

hours on a single-GPU machine", which shows an accelerating of the EM process by 3.8x with TPLM-based blocking. This observation also showcases the potential of performance improvements by integrating an efficient blocking step as a task in the entity matching process. However, Ditto does not report any blocking performance results with respect pairs completeness and reduction rate.

According to Li et al. [2020], they have chosen to retrieve $k = 10$ nearest neighbors for each record. The choice of k is an essential factor that influences the size of the candidate set. In addition, it affects the overall pair completeness of the system. A small candidate set can result in low pairs completeness and, and a large candidate set can inadvertently cause low pair quality (i.e. precision). The risk of missing out true matching record pairs increases when k decreases. In our approach, it also seems reasonable to use top-k nearest neighbors search instead of filtering based on a similarity threshold because identifying the threshold parameter for each dataset is quite difficult and impractical. In particular, the performance can deteriorates drastically as a comparison of dense attributes [Zhu et al., 2018].

On top of the transformer network for blocking, Ditto has added a classification layer with Softmax loss to unlock for co-embedding likely matching and non-matching record pairs. In this way, Softmax transforms the embedding vectors in the embedding space to normalize the transformer network's output to a probability distribution over predicted output classes. Consequently, the transformation might influence the difficult non-matches, often the doubt cases, by moving their representation in the embedding space from a doubt case to a more certain case. Ditto's model architecture seems to not compensate for the fact that many non-matching record pairs in datasets can have some degree of similarity. This disadvantage might affect a blocking approach combining TPLM and active learning strategies more negatively than a non-active learning approach with TPLM isolated.

There are several weaknesses related to Ditto's TPLM approach in a high resource setting. They assume labeled datasets are available. Unfortunately, they do not take advantage of an active learning strategies to reduce the labeling cost. Consequently, the availability of labeled data needs to be sufficient to fine-tune the TPLM properly for the task-objective. Additionally, Ditto did not report their PC score performance and reduction rate on the EM benchmark datasets.

3.4 Deep Active Learning with TPLM for Blocking and EM

Currently, Jain et al. [2021] published April 2021 the most recent work within deep active learning for a entity matching system. They propose Deep Indexed Active Learning (DIAL) based upon DTAL [Kasai et al., 2019]. Compared to AutoBlock, DIAL does not require any insight into the dataset beforehand. In contrast to Ditto, DIAL combines AL with TPLM to improve the performance in a blocking-matching combination in EM. As previously described, Ditto uses similarity search by blocked matrix multiplication [Abuzaid et al., 2019], and DIAL uses the similarity Search FAISS to retrieve nearest

neighbors [Johnson et al., 2021].

Jain et al. [2021] apply AL with TPLM and propose to integrate the blocking process in the active learning loop. The matching process is trained in an AL loop. DIAL differs to our research as we focus on solving the blocking process by combining AL with TPLM isolated from the matching process. Jain et al. [2021] claims that the blocking and the matching process can simultaneously leverage the advantages of using the newly labeled record pairs in the AL loop. In addition, they trained a committee on top of the transformer network to unlock the potential of the query strategy index-by-committee. From their method, they consider models for matching and blocking as separate systems, and they create a committee of multiple embeddings [Jain et al., 2021].

DIAL is evaluating their co-embed blocking and matching on several benchmark datasets as a unified entity matching system. It outperforms previous matching state-of-the-art results for entity matching, but it suffers from not reporting the blocking performance on the EM benchmark datasets. Additionally, they demonstrate how to leverage TPLM on a multilingual dataset [Jain et al., 2021]. Since DIAL was published in April 2021, it was intentionally not identified any weaknesses of their approach, and therefore, it is not addressed in our design choices.

3.5 Deep Transfer Active Learning for EM

Kasai et al. [2019] proposed approach Deep Transfer Active Learning (DTAL) to tackle EM in a low resource scenario, where the availability of labeled data is low. DTAL is of interest and related to our work due to its application of active learning in a low-resource setting. Contrary to Ditto, DTAL does not apply TPLMs in the matching process and does not address learning a blocker according to our research goal. Instead, DTAL combines transfer learning and active learning to target the low-resource setting for matching step in EM. The AL strategies used can be utilized in our work since DTAL uses independent TL and AL methods working even though the source datasets for TL are unavailable.

DTAL learns a transferable model from a high-resource setting, which is further adapted to a target dataset using active learning. Hence, their TL approach needs domain-related labeled target datasets, often with similar content as the source dataset, if TL should be applied. The approach learns a neural network architecture by integrating active learning strategies such as uncertainty sampling or partitioning sampling. These active learning strategies select informative examples from the target dataset to train the transferred model on. Combining TL and AL with traditional deep learning approaches demonstrates competitive performance with previous state-of-the-art matching methods on public benchmark datasets for EM with an order of magnitude fewer labels.

As mentioned in their work, Kasai et al. [2019] exploit AL query sampling strategy designed for DL models and EM as a task. The active learning strategies try to select a balanced set of positive and negative record pairs based on the model's predictions. Next, they divide the record pairs into two partitions, consisting of likely positive examples in

one partition if *prediction* ≥ 0.50 and likely negative examples in the other partition if *prediction* ≤ 0.50. In this way, the query sampling strategy then selects record pairs with the lowest entropy from both partitions, so-called high-confidence positives (HPC) and negatives (HCN), as a treatment to prevent overfitting in the network [Wang et al., 2017]. In addition, the query strategy selects the record pairs with the highest entropy, denoted as low confident positive (LCP) and negative (LCF) record pairs, as a treatment to improve pair completeness and pair quality [Qian et al., 2017].

According to their described algorithm, an Oracle labels LCP and LCN pairs while HCP and HCN pairs are labeled unsupervised using the model's predictions to define their corresponding labels. The purpose of the strategy is to leverage semi-supervised learning to feed twice as many record pairs into the model. An Oracle labels half of the record pairs, and the remaining half of the record pairs are labeled by the model's predictions. Their idea is to achieve enough labeled training data to train data-hungry DL models properly for the specific-task. They argue that the advantage of better performance gained from an increased training size is more significant than the disadvantage from potentially incorrect labeling of record pairs caused by the model's predictions.

A limitation of these active learning informative query strategies is their use of model predictions to select examples for the model in the matching step [Kasai et al., 2019]. Previously proposed blocking methods are not prepared for making a probability distribution over predicted output classes based on the record pairs in the candidate set. Lastly, we want to point out that their experimental results are evaluating the matching step in EM.

3.6 Earlier Work Summarized

Li et al. [2020] applied TPLM for their blocking approach in a high-resource setting, assuming a sufficient amount of labeled data was available. Unfortunately, data acquisition and annotation consumes a lot of manpower, and as a result, high level of domain expertise is required. In EM, deep learning models can be unfeasible to apply for blocking if the labeling effort cannot be reduced to a low-resource setting. We argue that Ditto has several weaknesses in that they 1) do not consider that a realistic blocking scenario with a limited amount of labeled data. Ditto needs substantial labeling effort upfront before the learning of the TPLM models can start to reach target performance. 2) When Softmax is used in Ditto's model architecture, they do not separate between dissimilar record pairs and record pairs with some degree of similarity. 3) Previously proposed blocking approaches [Ebraheem et al., 2019, Zhang et al., 2020, Li et al., 2020] are not prepared to integrate novel informative active learning strategies based on model predictions for TPLM-based blocking approaches.

In a low-resource setting, our proposed method addresses all three challenges jointly by: 1) Introducing active learning (AL) combined with TPLM to reduce the labeling cost while maintaining the blocking performance at a similar level as a high-resource setting. The availability of labeled data needs to be sufficient to fine-tune the TPLMs properly for the task-objective. 2) Considering record pairs represented as doubt cases in the embedding

space more accurately to treat the negatively effects caused by Softmax. 3) Unlocking the potential for leveraging novel informative active learning query sampling strategies to select examples more efficiently with respect to train TPLMs. In our approach, we apply the partition sampling strategies [Kasai et al., 2019] as described in Section 3.5, and uncertainty sampling [Lewis, 1995]. Chapter 5 introduces our design choices to bridge this gap.

Chapter 4

Data

The following chapter covers the structure and content of the public EM benchmark datasets used to evaluate the experiments.

4.1 Public Datasets

The datasets used to validate our approach are five of the most used public available datasets for benchmarking entity matching [Li et al., 2020], presented in Table 4.1. These widely applied datasets[1] are utilized among others in various types of EM experiments performed by Jain et al. [2021], Li et al. [2020], Mudgal et al. [2018] and Kasai et al. [2019].

These public datasets are pre-blocked, the number of labeled record pairs is less than the Cartesian product, as the datasets are prepared to evaluate matching processes as a task. Consequently, we have to reproduce the Cartesian product to prepare these datasets for our blocking step. It will, unfortunately, result in unlabeled and labeled record pairs in the candidate set, which needs to be handled as our active learning strategy can only take advantage of labeled examples.

These datasets exist in several versions in published papers depending on pre-processing and the split factor used for train, test, and validation sets. In this version of datasets, there are fixed splits for train, validation, and test set with a ratio of 3:1:1. Mudgal et al. [2018] investigated the advantages and limitations of DL models when applied to various EM tasks and categorized the datasets into structured, textual, and dirty data to better separate the DL performance over the datasets. In our case, we only evaluate textual and structured datasets according to Jain et al. [2021], Li et al. [2020]. Structured and textural record pair examples can be found in Table 4.3.

[1] `https://github.com/anhaidgroup/DeepMatcher/blob/master/Datasets.md`

4.1.1 Dataset Overview

The datasets consist of five tables, records A (A), records B (B), train, test and validation. A and B represent entities from two different sources. To exemplify, using the dataset Walmart-Amazon, A contains products from Walmart and B contains products from Amazon. All of these data are originally created from existing websites containing data of things like books, electronics, and products.

Table 4.1 shows dataset statistics. Amazon-Google contains software data from Amazon and Google. As observed from Table 4.3, the dataset is structured and not very textual with short strings. Amazon dataset has less missing values with approximately 4.87 % missing values than Google dataset with 29.65 %. DPLP-ACM has bibliographic data from DBLP and ACM, and it has few missing values. Compared to Amazon-Google, DPLP-ACM has more wordy attribute values. DBLP-GoogleScholar contains bibliographic data from DBLP and Google Scholar, where it contains less wordy strings than DPLP-ACM. Note that Google Scholar has significant amounts of missing values. Walmark-Amazon contains product data from Walmark and Amazon. Walmart-Amazon has some missing values, and the word count is on average more or less comparable to DBLP-GoogleScholar and DPLP-ACM. Abt-Buy is defined as a textual dataset, containing product data from abt.com and buy.com, respectively. The datasets Abt and Buy have a high concentration of missing values, and Abt has significant longer strings than Buy.

Table 4.1: Statistics of the five real-world datasets in the experiments. Column 'type' denotes data types categorized into structured and textual. Domain shows data content are split into four domains. |Size| indicates the number of labeled pairs in the dataset. |Matches| represents the number of positive pairs, i.e. marked as a match in the dataset. |Attribute| represents the number of attributes in the dataset being matched. Missing values, NA(A) and NA(B), refer to the fraction of missing values in the Dataset A and B, respectively. Word counting, WC(A) and WC(B), denote the average number of words in Dataset A and Dataset B. In this case, attribute values for each row are concatenated, and numerical values are ignored. Additionally, the datasets are categorized as $easy^+$ and $hard^*$.

Type	Dataset	Domain	Size	|Attribute|	|matches|	NA_A [%]	NA_B [%]	WC_A	WC_B
Structured	Amazon-Google*	software	11460	1167	3	4.87	29.65	8.10	8.20
	DBLP-ACM$^+$	citation	12363	222	4	0.00	0.15	18.30	21.80
	DBLP-Scholar$^+$	citation	28707	5347	4	4.10	19.37	16.20	18.80
	Walmart-Amazon*	electronics	10242	962	5	2.33	9.41	14.20	15.2
Textual	Abt-Buy*	product	9575	1028	3	20.44	28.79	47.10	17.10

We chose to partition these datasets into two categories, easy and hard, which also showed in Table 4.1. The former consists of mostly structured datasets and has less noise in terms of missing information or typos. Therefore, it can be expected to achieve higher performance on datasets categorized as easy. The challenging datasets, denoted as hard in Table 4.1, have typically unstructured attributes and corresponding noisy attribute values. This characteristic is often seen in datasets consisting of product descriptions and natural

language. On these "hard" datasets, it is expected that our approach will struggle much more to achieve high performance. As our experimental results will demonstrate later, our approach can get better results on DPLP-ACM and DBLP-GoogleScholar. According to traditional blocking, the easy datasets are also the data that requires less extensive effort from a domain expert to do data cleansing, feature engineering, and so on.

Table 4.3 shows matching and non-matching record pairs examples from the five tested datasets. Both matching and non-matching record pairs are showed with various degrees of similarity between Table A and B for each dataset.

4.1.2 Training, Validation, and Test Set

Each labeled datasets are split into the subsets Train, Validation, and Test by using a split ratio 3:1:1. Further, Train, Test and Validation each contain three attributes: *ltable_id*, *rtable_id* and *label*. *ltable_id* and *rtable_id* referred to records in Dataset A and Dataset B, respectively. *Label* is defined to be 0 for non-matching and 1 for matching record pairs between Dataset A and Dataset B.

Table 4.2 shows positive rate for Train, Test and Validation for every dataset according to the split 3:1:1, respectively. The positive rate across the datasets varies at most with approximately 9.24 %. While Walmart-Amazon has the fewest matches with approximately 9.38 %, DBLP-GoogleScholar has the highest positive rate with approximately 18.62 % matches. The positive rate is an essential indicator to track because the query sampling strategies can be influenced by running out of matching pairs in the AL iterations. Additionally, the amount of record pairs varies significantly across the datasets, Abt-Buy has 9575 record pairs while DBLP-GoogleScholar has almost three times more with 28707 record pairs.

The label budget rate is derived by labeling budget of 1000 examples over the total number of record pairs in the respective datasets. As seen from Table 4.2, our labeling budget represents between 5.81 % and 17.41 % of the total size of the dataset, a significant reduction in labeling cost from a high resource setting to a low resource setting.

Table 4.2: The respective labeled datasets are split into three subsets of labeled pairs; train, validation, and test by using a split ratio 3:1:1. The positive rate among the train, validation, and test within each data sets are approximately constant. Additionally, the number of positive record pairs over total number of record pairs are showed regarding train, validation, and test sets. Last, the labeling budget rate are computed corresponding to significant reduction in labeling cost.

Dataset	Pos. Rate [%Pos]	Train [Pos/Tot]	Test [Pos/Tot]	Validation [Pos/Tot]	Total [Pos/Tot]	Label Budget Rate [%Budget/Total]
Amazon-Google	10.17	699/6874	234/2293	234/2293	1167/11460	14.55
DBLP-ACM	17.96	1332/7417	444/2473	444/2473	2220/12363	13.48
DBLP-GoogleScholar	18.62	3207/17223	1070/5742	1070/5742	5347/28707	5.81
Walmart-Amazon	9.38	576/6144	193/2049	193/2049	962/10242	16.28
Abt-Buy	10.73	616/5743	206/1916	206/1916	1028/9575	17.41

Table 4.3: Examples of matching and non-matching record pair from the five tested datasets.

(a) Abt-Buy

Match	Table	Name	Description	Price
True	A	monster icarplay wireless 250 fm transmitter with autoscan for ipod and iphone aipfmch250	monster icarplay wireless 250 fm transmitter with autoscan (...)	100.0
	B	monster a ip fm-ch 250 icarplay wireless 250 fm transmitter for ipod & iphone aipfm-ch250	NA	NaN
False	A	escort passport radar and laser detector black finish 8500	escort passport x50 radar and laser detector 8500 x-band (..)	313.95
	B	escort passport 9500ix radar/laser detector	NA	439.99

(b) DBLP-ACM

Match	Table	Title	Authors	Venue	Year
True	A	application servers : born-again tp monitors for the web ? (panel abstract)	c. mohan	sigmod conference	2001
	B	application servers (panel session) : born-again tp monitors for the web	c. mohan , larry cable , matthieu devin , scott dietzen , pat helland , dan wolfson	international conference on management of data	2001
False	A	the sift information dissemination system	tak w. yan , hector garcia-molina	acm trans . database syst .	1999
	B	duplicate removal in information system dissemination	tak w. yan , hector garcia-molina	very large data bases	1995

(c) Amazon-Google

Match	Table	Title	Manufacturer	Price
True	A	instant immersion italian v2 .0 (large box)	topics entertainment	29.99
	B	topics entertainment instant immersion italian 2.0	NA	17.55
False	A	printmaster 17 platinum by encore software	encore software	39.99
	B	printmaster (r) platinum 16	NA	39.97

(d) DBLP-GoogleScholar

Match	Table	Title	Authors	Venue	Year
True	A	automated resolution of semantic heterogeneity in multidatabases	m bright , a hurson , s pakzad	acm trans . database syst .	1994
	B	automated resolution of semantic heterogeneity in multi-databases	ar huroson , s pakzad	acm transactions on database systems ,	NA
False	A	building and customizing data-intensive web sites using weave	k yagoub , d florescu , v issarny , p valduriez	vldb	2000
	B	caching strategies for data-intensive web sites	k yagoub , d florescu , v issarny , p valduriez	vldb ,	2000

(e) Walmart-Amazon

Match	Table	Title	ModelNo	Category	Brand	Price
True	A	home portable stereo alarm clock with ipod dock gunmetal	electronics - general	ihome	ih16g	49.73
	B	home ih16 portable speaker system for ipod gray	speaker systems	ihome	ih16gxc	40.99
False	A	griffin ipod touch 4g screen care kit 3 pack	mp3 accessories	arkon	gb01913	13.3
	B	griffin gb01909 screen care kit for ipod nano 6g 3 pack	griffin technology	gb01909	12.11	

Chapter 5

Method

In this chapter we describes our approach TopKDAL, Top-K based Deep Active Learning solution for blocking as an entity matching task. First, the blocking strategy is introduced and then active learning strategy is explained. Lastly, the experimental setup is presented.

5.1 Blocking Strategy

5.1.1 Design Choices

We identified several challenges and limitations based on previous work in Chapter 3 and our empirical testing of various approaches for identifying a subset of the Cartesian Product, a so-called candidate set, that contains likely matching record pairs. Consequently, TopKDAL introduced three important design choices based on leveraging transformer pre-trained language models for blocking. The model is learned in an active learning loop in a low-resource setting. As follows, we outline each of these design choices.

- **Design choice 1) Top-k vs. Threshold Selection**
  Top-k search retrieves the k most similar pairs between all records using a specific record as a reference. The candidate set will consist of the pairs retrieved from the top-k search. The alternative is to set a parameter threshold. If this parameter is identified, then the candidates will be all pairs of records with similarities above this particular threshold. We chose top-k search to separate matching record pairs from non-matches, and as a result, avoid the burden to find a suitable threshold to achieve the same objective. In addition, top-k search seemed to be the most applied approach in the previous works.

  A challenge is to evaluate blocking performance on benchmark datasets when top-k search retrieves a candidate set consisting of labeled and unlabeled examples. As mentioned in Chapter 4, we have to recall that our benchmark datasets for training, validation, and test datasets are labeled after a pre-blocking schedule.

Per definition, those datasets are originally intended for evaluating matching steps rather than blocking steps. The burden of exposing the domain expert for even more time-consuming labeling of records was not a cost of man-hours we could defend in a production setting for the companies[1]. In advance of a top-k search, it is also impossible to predict which pairs of records will end up in the candidate set as output from the blocker. Hence, it is also challenging to forecast which pairs of records should be labeled beforehand. Consequently, we chose to filter out these unlabeled examples from the candidate set as a treatment for missing labels. The reason is that our query sampling strategies are prepared to select only labeled examples. This treatment was reasonable because the selected instances for each active learning iteration have to be labeled somehow by an Oracle.

- **Design choice 2) Choice of Loss Function vs. Query Sampling Strategy**
  Loss function choice is a critical component to fine-tune the model. It also determines how our embedding model works for the specific downstream task. To fine-tune the transformer network, we wanted somehow to affect our network to have similar record pairs closer in the embedding vector space while dissimilar record pairs should be further apart in the embedding space. That property indicated the characteristics of choosing a loss function such as TripletLoss. On the other hand, our system constraints of using informative query sampling strategies in the active learning loop indicated a transformation was needed to map the distribution of similarities in the embedding space to a probability distribution ranging between 0 and 1. Consequently, we decided to apply Softmax as our loss function to provide such a scale.

- **Design choice 3) Mimic Record Similarities in Pre-blocked Benchmark Datasets**
  The datasets are initially pre-blocked, as a result, it should be expected some similarity between the non-matching pairs. Forcing these pairs to converge towards a target of zero would be a false representation of the similarities. Hence, soft labels were a crucial component to set lower target for non-matches at a value above 0 to mimic some degree of pair similarity present between the records in the datasets.

5.1.2 Our Solution

According to our design choices, the main idea was to develop a deep learning methodology based on transformer-based pre-trained models (TPLMs) to leverage their strength of highly contextualized embeddings to provide better language understanding and semantics in words. Compared to conventional word embeddings methods (e.g. GloVE [Pennington et al., 2014], word2vec[Mikolov et al., 2013], and FastText[Bojanowski et al., 2017]), our approach TopKDAL is constructed to take benefit of these pre-trained language models by adding task-specific layers on top of the TPLM and fine-tuned for our task-specific objective to greatly improve the performance. Sadly, there is no single training strategy that works for all use-cases in entity matching. Instead, which training strategy to use

[1]For blocking, the optimal benchmark datasets have labeled all pairs of records. Currently, it is seldom datasets are completely labeled making evaluating of blocking approaches more difficult compared to matching approaches.

greatly depends on our available data and on our target task. We train on labeled data for updating the weights in the transformer network to produce meaningful embeddings for the records. This training involves a memory-intensive process of fine-tuning millions of parameters for the task based on a sequence of tokens as input to the transformer. The matching objective is to precisely classify matching and non-matching pairs of records while blocking attempts to separate the similarities between matching and non-matching record pairs to maximize the number of matching pairs in the candidate set C. An entity matching system has both blocking and matching capacities such as DIAL[Jain et al., 2021] and DITTO[Li et al., 2020], this could affect our blocking approach regarding which TPLM to use for the experiments since it might be simpler to use a single TPLM for a complete entity matching system. However, TopKDAL has not taken these entity matching system considerations into account in the design. Our approach has only been evaluated as an isolated unit to maximize blocking performance.

Encoding Records to Embeddings for Suiting Transformers

In blocking, the paired mode[2] commonly used in a matching task on every record pair in the Cartesian Product is not recommended [Jain et al., 2021, Li et al., 2020]. Instead, we separately encode each record according to a single mode[Jain et al., 2021, Reimers and Gurevych, 2019]. For each record pair, we then passed separately the records in dataset A and the records from dataset B through the transformer network, which yields the embedding $E(x)$ for record x in dataset A and the embedding $E(x)$ for record x in dataset B. The similarity of these embeddings was computed using cosine similarity, and the result was evaluated by comparing to the gold standard.

To obtain a embeddings $E(x_1), ..., E(x_n)$ for record x in Dataset A or Dataset B, we fed record x into the pre-trained transformer network such as RoBERTa, and DistilBERT according to

$$[CLS], x_1...x_n, [SEP] \tag{5.1}$$

where $x_1...x_n$ denotes the input tokens of the record x, CLS is a special start token, and SEP is a special separator token. According to Li et al. [2020], the each record $x = (attr_i, val_i)_{1 \leq i \leq k}$ can be serialized as

$$s(x) = [COL]attr_1[VAL]val_1...[COL]attr_k[VAL]val_k \tag{5.2}$$

where [COL] and [VAL] are special tokens to denote the start of attribute name and attribute value, respectively. The serialized record s(x) is inserted in Equation 5.3 as

$$[CLS], s(x), [SEP] \tag{5.3}$$

[2]In paired mode, the transformer feeds a token concatenation of two records for all record pairs in the dataset.

Note that each record $s(x)$ is serialized individually in contrast to a matching setting serializing a candidate pair $s(x, x')$ instead. Table 5.1 illustrates the challenge of converting a record x to a representation $s(x)$ that our chosen individual language model layer, such as RoBERTa and DistilBERT, can interpret to preserve the information given in the attribute names and values derived by the dataset Abt-Buy.

Table 5.1: Encoding of two individual records from dataset Abt-Buy. These records are also presented in Table 4.2.

Table	Encoded Record
Abt	[COL] name [VAL] onster icarplay wireless 250 fm transmitter with autoscan for ipod and iphone aipfmch250 [COL] description [VAL] emonster icarplay wireless 250 fm transmitter with autoscan (...) [COL] price [VAL] 100.0
Buy	[COL] name [VAL] monster a ip fm-ch 250 icarplay wireless 250 fm transmitter for ipod [COL] description [VAL] NULL [COL] price [VAL] NULL

The language model produces contextualized word embeddings for all input tokens, $x_1...x_n$, of the record x. Concerning various sequence lengths of input tokens, we limit the layer by defining that records with greater than 512 input tokens were truncated. However, this was not observed as an issue for our datasets. We used a pooling layer to get a fixed-sized output representation of vector $u(x)$. This fixed-sized, $d = 512$, dimensional contextual embeddings $E(x_1), ..., E(x_n)$ was obtained from the multiple layers of self-attention in the transformer network, and the embedding of the record x, E(x), was computing by averaging all contextualized word embeddings the language model produced. We used the default pooling strategy called MEAN as Reimers and Gurevych [2019], it is derived as

$$E(x) = \frac{1}{n} \sum_{i=1}^{n} E(x_i) \tag{5.4}$$

to achieve a fixed 512 output vector E(x) independent of the lengths of the record.

Further, as we separately computes E(x) for every record in Dataset A and B to map variables length input tokens to a fixed-sized dimensional contextual embeddings, we then needed to evaluate whether a pair of records were a match or non-match. For a pair of records (a, b) between Dataset A and B, we took the concatenation of the embedding E(a) of record a, E(b) of record b and the absolute difference between the two embeddings $|E(a) - E(b)|$ [Reimers and Gurevych, 2019]. Each concatenation of a pair of records was forwarded to a linear layer.

Based on inspiration from the work of Reimers and Gurevych [2019], a linear layer was added after concatenating of the embeddings. The simplified linear layer was an uninitialized layer. Hence, it was dependent on available labeled training data.

Training Objective of Classification Layer

The output from the linear layer was forwarded to a classification layer with Softmax. The classification layer fulfilled two objectives. 1) The first objective was to use Softmax to transform the embedding space to normalize the network's output to provide a probability distribution over predicted output classes in the range [0, 1]. It was essential to unlocking the potential of active learning as a strategy for blocking. We must recall that query sampling strategies need model prediction probabilities to select examples from the unlabeled pool. 2) The second objective was to introduce Softmax to have the ability to evaluate the corresponding loss in the model during training on the labeled datasets. Negative log-likelihood was applied here.

Doing so, Softmax pushed the highest identified values to converge towards the target value 1, typically matching record pairs. Correspondingly, Softmax pushed non-matches to converge towards the target value 0. By default, the lower target value of Softmax is 0. As described in design choice 1), we could observe that the non-matching record pairs in the pre-blocked benchmark datasets had some similarities. Hence, we used soft labels to mimic this fact into the model architecture by changing the lower target value from being 0 to be minimum 0.1. This seemed reasonable as the remaining record pairs in the pre-blocked benchmark dataset were meant for evaluating matching approaches.

In TopKDAL, we chose the Binary Cross-Entropy Loss with Logits[3] because the model achieved better numerical stability and model performance compared to using the Cross-Entropy Loss alone. In this way, we took advantage of the log-sum-exptrick for numerical stability in Binary Cross-Entropy Loss with Logits (BCEWithLogits) by combining Softmax layer with a BCELoss in one single class.

5.2 Active Learning Strategy

Active learning strategy was the second component required to develop our approach TopKDAL. The overall active learning algorithm used in the query sampling strategies is explained here. The evaluation metrics used to report the results are previously described in Section 2.6.

5.2.1 Active Learning Algorithm

In the experiments, the query sampling strategies were tested one-by-one by using the following active learning algorithm described stepwise. Figure 5.1 highlights the main operations performed by TopKDAL during active learning iterations, and clearly describes the data flow.

[3]A detailed derivation of BCEWithLogits can be found here: `https://pytorch.org/docs/master/generated/torch.nn.BCEWithLogitsLoss.html#torch.nn.BCEWithLogitsLoss`

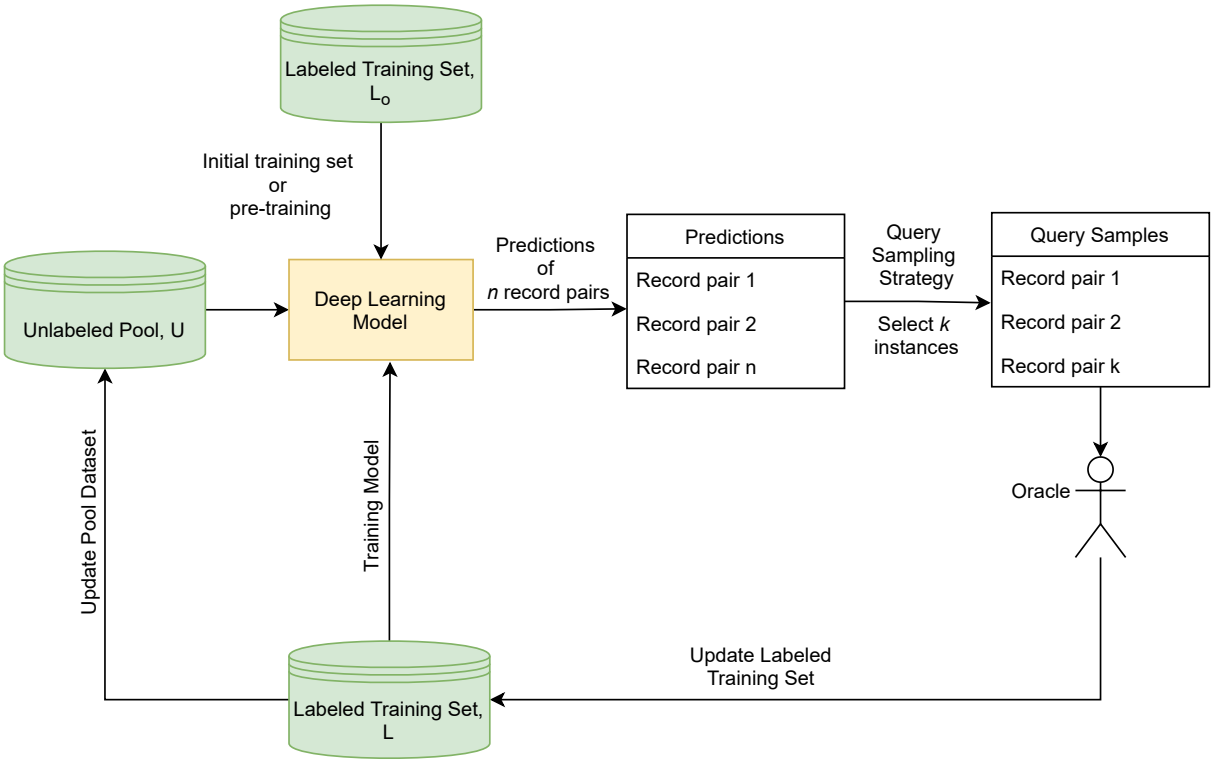


Figure 5.1: A simplified deep active learning process for our proposed Top-KDAL. Four concepts of sets are applied for the ongoing bookkeeping of record pairs in the active learning loop. The original training set, $\mathcal{T}$, is not considered in the illustration. Before active learning loop started, we selected examples from $\mathcal{T}$ to the initial training set $\mathcal{L}_0$. During the active learning loop, the newly selected examples are added to the updated training set $\mathcal{L}$. Unlabeled data in the pool, $\mathcal{U} = \mathcal{T} - \mathcal{L} - \mathcal{L}_0$, consists of the remaining record pairs unseen for the model during the active learning process. The unlabeled data in $\mathcal{U}$ achieved prediction scores from the model after each AL iteration, which the query sampling strategy used as a decision basis to select K new instances to label.

1. **Initialization of training set.** All models were initialized with an initial training set, $\mathcal{L}_0$, of 200 labeled record pairs from the original training set, $\mathcal{T}$, before the active learning loop started. The model was trained on $\mathcal{L}_0$ to generate prediction scores on the record pairs in unlabeled data pool $\mathcal{U}$.

   Two starting strategies were tested, either a balanced and an imbalanced initial training set. *Imbalanced initial training set* was conducted of 200 randomly selected record pairs in $\mathcal{T}$, in which the positive rate is assumed to be approximately as described in Table 4.1. *Balanced initial training set* was represented by selecting 100 positives and 100 negatives record pairs randomly from $\mathcal{T}$, summed up to 200 selected record pairs.

2. **Active learning procedure.** After initialization of training set $\mathcal{L}_0$, each active learning (AL) query strategy started to train the transformer pre-trained language model on $\mathcal{L}_0$. The query strategy could then select new examples from $\mathcal{U}$ based on

the model's prediction scores. The unlabeled pool, $\mathcal{U}$, is a function of the difference between $\mathcal{T}$ and $\mathcal{L}$ in each AL iteration. Before each iteration, the model was restarted and initialized with the origin weights from the TPLM. Then, it started the training of model on the updated training set $\mathcal{L}$. Next, the query strategy could select new examples again with respect to model's new prediction scores. In every iteration, the newly selected examples from the query strategy were removed from $\mathcal{U}$ and added to $\mathcal{L}$. All examples in $\mathcal{L}$ were labeled by the Oracle (see annotator below). The active learning iterations continued until the stop criteria was reached.

Important note: Top-k search generated pairs of all records in the training set $\mathcal{L}$, and as result, we implemented a filter such that the query sampling strategies could only retrieve labeled record pairs located in $\mathcal{L} \cap \mathcal{T}_{train}$. This was required to fulfill the requirement of assigning a label to all pairs of records selected by the query strategies.

3. **Model evaluation.** After each AL iteration, the model was evaluated on the test set, D_{test}, with respect to evaluation metrics described in Section 2.6. We kept the D_{test} unchanged for each iteration, and the results from each iteration are what is reported in Chapter 6. The evaluation step on the D_{test} used a top-K search to retrieve the record pairs with highest degree of similarity to the candidate set. It is important to keep in mind that the evaluation step after each AL iteration was independent of the prediction step performed inside the active learning loop.

4. **Stop criteria.** The AL iterations continued according to this procedure until the stop criteria was met. In our experiment, the stop criteria was set to a labeling budget of 1000 record pairs, $S = 1000$, for achieving the experimental run time within a reasonable time usage. The query strategies were dependent on the model's prediction scores to select K examples in each iteration. It was no guarantee to reach 1000 labeled examples. The number of AL iteration, I, was derived by

$$\frac{S - L_0}{K} = I \tag{5.5}$$

In our experiments, this amounted to 20 iterations because we set the sampling size to $K = 40$ examples in each iteration. This value of K was set by the experiences observed in Kasai et al. [2019], and our empirical testing on computational run time by varying the sampling size K and hyperparameters in the experiments. In fact, Equation 5.5 indicates increasing computation total run time when decreasing K due to an increasing number of iterations are required to reach $S = 1000$ labeled examples. Every experiment used an initial training set of 200 examples as described previously.

5. **Annotator.** As annotator, also known as the Oracle, we used the respective labeled datasets as a gold standard. In the training set, the Oracle manually labeled the examples, or more precisely, we used the gold standard as labels for the examples to mimic an Oracle doing labeling. In the prediction step, the labels for the training set was not available for the model.

5.3 Evaluated Query Sampling Strategies

Our approach TopKDAL is evaluated on 3 non-active learning-based experiments and 7 active learning-based experiments. Those seven active learning methods include one supervised baseline, three existing query sampling strategies, and our three new query sampling strategies.

5.3.1 Supervised Baseline

Currently, there are no previously published works that leverage deep active learning (DAL) approaches for blocking, and as a result, there is no natural point of comparison other than using two types of supervised baselines to evaluate our proposed TopKDAL and active learning query sampling strategies. These two supervised baselines are natural baselines to apply for evaluating a DAL approach for the first time within blocking.

Supervised Baseline with Active Learning. This active-learning based supervised Baseline, hereafter denoted as Baseline, used a query sampling strategy of naively sampling record pairs randomly for every active learning iteration, a standard often used in supervised learning strategies. The purpose was to establish a Baseline as benchmark target to evaluate the different active learning strategies compared to the performance of random query sampling strategy. In published research, it has been less attention of combining active learning (AL) with TPLM for blocking in entity matching. Consequently, to the best of our knowledge, there are no benchmarks to compare TopKDAL with either it comes to fine-tuning of hyperparameters, choice of language models or performance. Therefore, this optimization should be investigated in further work, as explained in Chapter Section 8.2.

This Baseline was implemented based on the active learning procedure as explained in Section 5.2.1, in which initially trained the model on $\mathcal{L}_{init}$ before selecting $K = 40$ random record pairs labeled by the Oracle in each AL iteration. For every K new received record pairs added to training set $\mathcal{L}$, the model was restarted and retrained on the updated training set $\mathcal{L}$ in each iteration. This AL loop continues until the stop criterion was reached, determined in Section 5.2.1.

Supervised Baseline with Non-Active Learning. We choose also to compare our proposed method against non-active learning-based Baseline to identify sufficient training set sizes for the model with respect to PC score, particular important to identify for the most difficult datasets. These baselines were run by 1/1, 1/2, and 1/4 of the training set size by applying random choice without replacement[4]. The non-active learning-based Baseline trained on the entire training set is denoted Baseline-Max, Baseline-1/2 trained on 50% of the training set, and the last Baseline-1/4 trained on 25% of the training set. In the two last Baselines, the record pairs consisting the training set are selected randomly with an approximate true positive rate estimated in Table 4.2.

[4]Documentation of Numpy random choice can be found at `https://numpy.org/doc/stable/reference/random/generated/numpy.random.choice.html`.

5.3.2 Existing Query Sampling Strategies

We chose to evaluate three well-known existing query sampling strategies on TopKDAL. These methods are used in previous active learning based work in entity matching [Kasai et al., 2019, Jain et al., 2021].

Margin-Based Sampling (Uncertainty sampling)

Margin-based sampling is a well-known query sampling strategy in active learning (AL) [Meduri et al., 2020], and uncertainty seemed to be essential as a strategy in our experiments. This query strategy aims to select the k most uncertain, also known as low confident, record pairs based on the model predictions on the unlabeled set $\mathcal{U}$. In each AL iteration, k record pairs closest to 0.50 in model's predictions are selected. There are no requirements to achieve a balanced set of low confident positives (LCP) and low confident negatives (LCN), which can result in LCNs are only selected in cases where model's predictions are less than 0.5. It can be noted that the terms low-confident record pairs and uncertain record pairs are used interchangeably.

The objective of testing margin-based sampling as a query sampling strategy was to evaluate the impact of low confident (LC) record pair compared to other strategies, such as Partition-2 and Partition-4, involving both LC and high confident (HC) record pairs or only HC record pairs.

Partition Sampling

Partition sampling is a tested query sampling strategy used in combination with deep learning and active learning (DAL) for entity matching (EM) by Kasai et al. [2019]. Their observation claimed that the model would be more resistant to overfitting by selecting high confident (HC) and low confident (LC) examples. This guided approach of sampling was also observed to improve precision and recall in a low resource setting. Instead, we chose to test this query sampling strategy with respect to our TopKDAL for blocking in a low resource setting. We implemented two partition strategies inspired by Kasai et al. [2019].

Partition-2 According to Kasai et al. [2019], Partition-2 is a semi-supervised learning method leveraging pseudo labels based on a labeled data model to add confident predicted unlabeled data to the training set. The purpose is to increase the size of the training set by inserting initially unlabeled data to improve the model accuracy. The semi-supervised strategy aims to select a balanced set of record pairs consisting of k high confident negatives (HCN) in an unsupervised manner, and k low-confidence positives (LCP), and k low-confident negatives (LCN). In our experiments, $k = 20$ to ensure 40 manually labeled examples across every query sampling strategy in the experiments.

We implemented Partition-2 by partitioning the record pairs in the unlabeled pool $\mathcal{U}$ into

two subsets. One subset consisted of positive model prediction while the other subset consisted of negative model prediction scores. By sorting these subsets based on the prediction scores, we could efficiently select k low-confident positives from the first subset and low-confident negatives from the second subset. These k LCPs and k LCNs were the record pairs the model was most uncertain about based on its predictions, and it was labeled manually by the Oracle. In addition, among these originally unlabeled record pairs in $\mathcal{U}$, it was selected k HCP and k HCN based on the most certain model predictions. In contrast to LCP and LCN, these HCPs and k HCNs were automatically labeled unsupervised in terms of their corresponding model predictions. This resulted in 40 manually labeled examples, 40 unsupervised labeled examples, and in total 80 labeled examples returned from each iteration for Partition-2.

Partition-4 We implemented Partition-4 to have a partition sampling method considering labeling only based on the Oracle. Partition-4 aims to select a balanced set of record pairs composed of k high confident positives (HCP), k high confident negatives (HCN), k low-confidence positives (LCP), and k low-confident negatives (LCN). In our experiments, k is set to 10 to let each of the four partitions in Partition-4 contain 10 record pairs.

In practice, Partition-4 works by partitioning the record pairs in the unlabeled pool $\mathcal{U}$ into two subsets, where the positive model predictions are placed in one subset while the negative model predictions are placed in the other subset. These two subsets are sorted based on their prediction scores. Then, we find the k and k lowest prediction scores in the subset of positives corresponding to HCP and LCP, respectively. The similar procedure was followed to find LCN and LCN in the subset of negatives. In contrast to Partition-2, the Oracle manually labels HCP, HCN, LCP and LCN in Partition-4. Thus, the drawback of Partition-4 is incorrectly labeling of record pairs by the Oracle. We used the gold standard to mimic the labeling by the Oracle, hence, that was not an issue in our experiments. A illustration describing the nuances in the query sampling Partition-4 and Partition-2 compared to Margin-based can be seen in Figure 5.2.

5.3.3 Our Introduced Query Sampling Strategies

We developed three new query sampling strategies inspired by the existing query sampling strategies. The purpose was to cover the missing counterparts to the existing query sampling methods to gain better understand about how these methods are affecting our TopKDAL over the AL iterations.

High Confident Positives/Negatives

High Confident Positives/Negatives, hereafter denoted as HC-P/N, tries to select a balanced set consisting of k high confident positives (HCP) and k high confident negatives (HCN), in which $k = 20$ in our experiments. The objective of testing this query sampling strategy as a counterpart to margin-based sampling is to understand better the impact of feeding the model with high confident (HC) record pairs instead of low confident (LC) record pairs.

Random Positives/Negatives

Random Positives/Negatives, abbreviated as Random-P/N, aims to evaluate how query sampling informativeness impacts the model in terms of using guided learning based on HC and LC model's predictions. In Random-P/N experiments, the method selects a balanced set of $k = 20$ positive and $k = 20$ negative examples randomly from the unlabeled pool, $\mathcal{U}$, with respect to whether their predictions have a $value \geq 0.50$ or $value < 0.50$.

Partition-4 Based Positive Boost

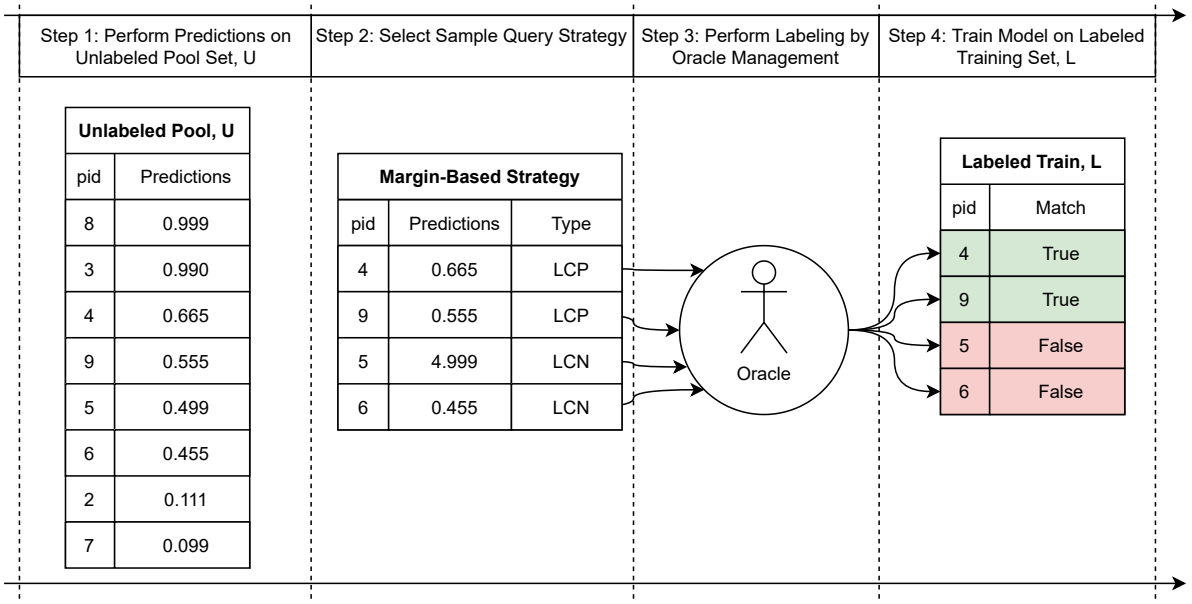
Partition-4 based Positive Boost, shortly named P-4-3xPB, is a modified Partition-4 approach designed to allow three times more HCP than HCN record pairs, twice as many HCP record pairs compared to Partition-4. This configured class distribution of positives and negatives is seldom present in the datasets. In our experiments, each iteration selected $K = 40$ record pairs, and in the best case P-4-3xPB selects $HCP = 15$, $LCP = 10$, $LCN = 10$, and $HCN = 5$ given sufficient model predictions covering the whole probability distribution in the interval $[0, 1]$. This query sampling strategy aims to evaluate if increasing the number of HCPs while decreasing the number of HCNs is beneficial to improve model performance in achieving higher PC scores.

5.3.4 Query Sampling Strategies Compared

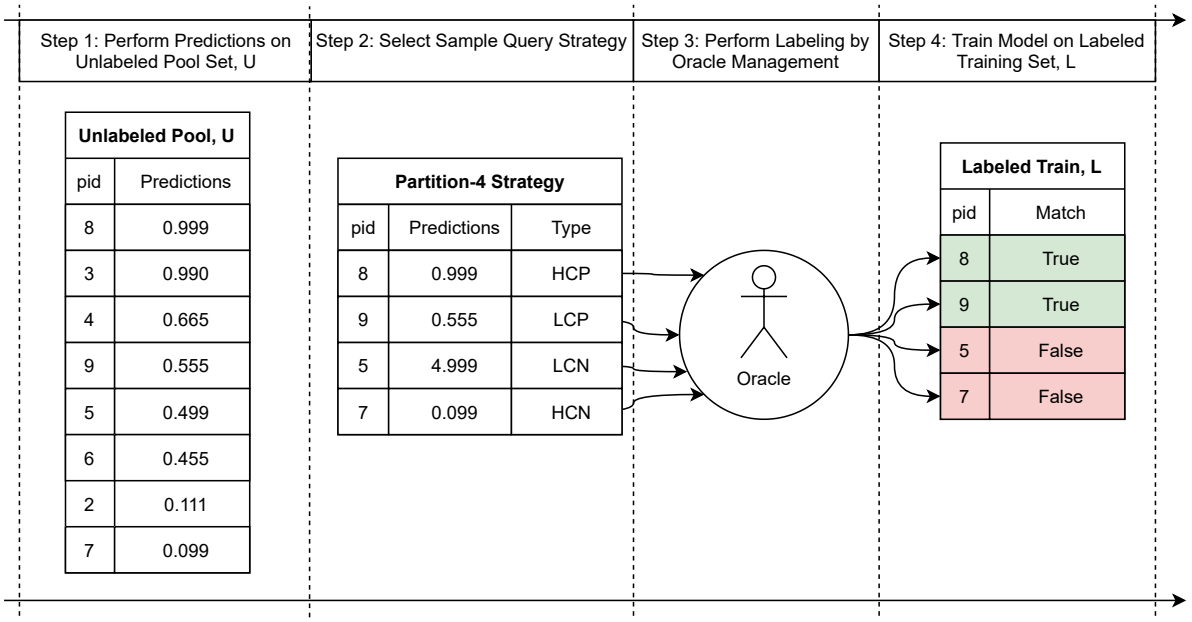
Table 5.2 shows how the passive and active query sampling strategies select different types of knowledge for the model. As illustrated in Figure 5.2, Partition-2 achieves twice as many labeled record-pairs compared to the other query sampling methods, which gives at most 800 additional labeled instances after 20 iterations, or 80% more labeled instances in training set, compared to the other query sampling strategies. However, half of the record-pairs during the AL loop might also be incorrectly labeled due to the pseudo labeling of HCP and HCN given the assumption that the Oracle always labels LCPs and LCNs correctly. We chose to include both Partition-2 and Partition-4 in our experiments to identify the effect of semi-supervised learning. Last, it can be mentioned that Random-P/N is the only query sampling strategy selecting a balanced set by sampling positive and negative examples randomly.

Table 5.2: Comparison between active learning query sampling strategies.

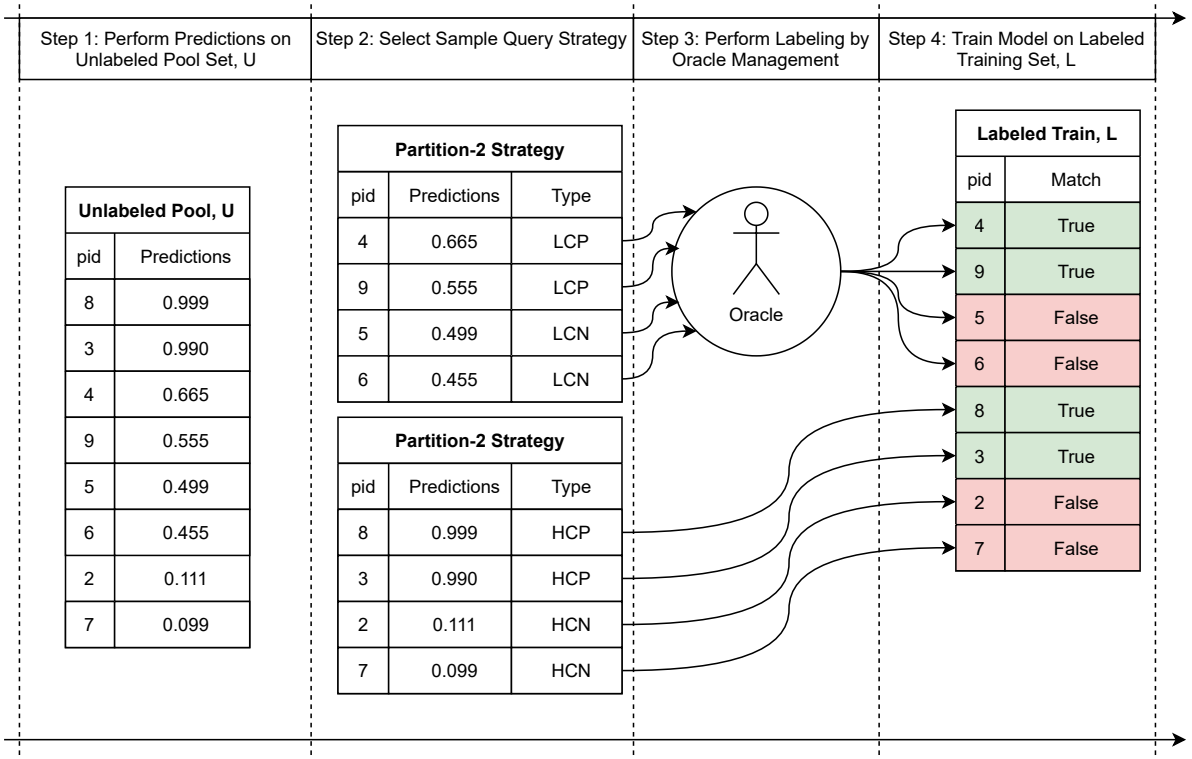
			Passive QSS	Active Query Sampling Strategies (QSS)					
Class ratio	Class Type	Sampling Strategy	Baseline	Uncertainty	HC-P/N	Random-P/N	P-4	P-2	P-4-3xPB
50/50	Positives	High Confident	-	-	X	-	X	-	X
		Uncertainty	-	X	-	-	X	X	X
		Random	-	-	-	X	-	-	-
	Negatives	Random	-	-	-	X	-	-	-
		Uncertainty	-	X	-	-	X	X	X
		High Confident	-	-	X	-	X	-	X
Random	Pos/Neg	Random	X	-	-	-	-	-	-



(a) Margin-based query sampling strategy. This method selects only low confident positives (LCP) and low confident negatives (LCN) instances during the query sampling. Last, the Oracle labels the instances selected by the query.



(b) Partition-4 query sampling strategy. The method selects high confident positives (HCP), high confident negatives (HCN), low confident positives (LCP), and low confident negatives (LCN). As illustrated, all the selected instances are labeled by the Oracle.



(c) Partition-2 query sampling strategy. The method selects high confident positives (HCP), high confident negatives (HCN), low confident positives (LCP), and low confident negatives (LCN). In contrast to Partition-4, only the LCP and LCN selected instances are labeled by the Oracle while HCP and HCN instances got their labels according to their assigned predictions.

Figure 5.2: Comparison of the query sampling strategies (a) Margin-based, (b) Partition-4, and (c) Partition-2. In the illustration, each query sampling strategy has an Oracle labeling four record pair, given the assumption $K = 4$. With respect to this assumption, Partition-2 achieves twice as much labeled record-pairs. However, in the worst case, half of the record-pairs might also be incorrectly labeled due to wrong prediction-based labeling. It is assumed the Oracle always labels correctly.

5.4 Experiments Setup

5.4.1 Datasets

Our proposed approach is validated on five real-world, publicly available and widely used datasets desribed in Chapter 4. These five datasets are already pre-blocked making the datasets mainly suitable and prepared for evaluating an isolated matching step. In our case, these blocked datasets define methodology constraints in our design choices and blocking approach with respect to being able to evaluate our TopKDAL on these datasets with their corresponding test sets D_{test}.

These five datasets are categorized into two domains, Walmart-Amazon, Amazon-Google, and Abt-Buy are product datasets, whereas DBLP-ACM and DBLP-Scholar are citation datasets. Abt-Buy is a textual dataset, whereas the other four are structured datasets. Chapter 4 provides more information regarding the datasets. As our motivation in Section 1.1, there may be scenarios where the record pairs between the datasets are incomparable, making rule based blocking methods infeasible. Our proposed TopKDAL with TPLM can also manage such scenarios.

5.4.2 Experiment Implementation Details

There are several hyperparameters and system constraints to consider in our experiments. In many cases, it is not a matter of course that the TPLMs give correct class predictions out-of-the-box. As experienced in our experiments, it was needed to fine-tune the pretrained language model (TPLM) to task-specific data to achieve suitable model predictions. Active learning was used to iteratively select and ingest relevant record pairs into the transformer model architecture.

Hardware/Computing. We performed the evaluation of our approach TopKDAL on the NTNU IDUN computing cluster [Själander et al., 2020]. The cluster has more than 70 nodes and 90 GPGPUs. Each node contains two Intel Xeon cores, at least 128 GB of main memory, and is connected to an Infiniband network. Half of the nodes are equipped with two or more Nvidia Tesla P100 or V100 GPGPUs. Storage is provided by two storage arrays and a Lustre parallel distributed file system.

All of our experiments used a hardware environment consisting of Dell PE740 with Intel Xeon Gold 6132 CPU with 28 cores. Experiments were run with exclusively privileges on this hardware environment with a single NVIDIA Tesla V100 16 GB GPU and Python/3.8.6-GCCcore-10.2.0 and CUDA/11.1.1-GCC-10.2.0. Thus, every experiment on the nodes did not share any resources on the hardware environment with others during our experimental runs.

Libraries/Tools. All experiments used PyTorch 1.7.1, deep learning framework, [Paszke et al., 2019], and sentence transformers library 0.4.1 by Huggingface [Wolf et al., 2020] to train and test TPLM combined with active learning for blocking.

Experimental reproducibility. All experiments have been run three times and averaged over their three results to validate the experimental results with respect to standard deviation and variance. In Section 6.2, the geometric average / mean and standard deviation from the performed experiments are showed. A requirement was to perform reproducible experimental results, thus, each experiment used a predefined random seed list to handle the randomization equally across the experiments. Our way of solving this was to set the experimental run 1, 2, and 3 to apply 1, 2, and 3 as a random seed, respectively.

Transformer Pre-trained Language Model (TPLM) and Hyperparemters. We used both the pre-trained language model RoBERTa [Liu et al., 2019] and DistilBERT [Sanh et al., 2020] in our experiments with a careful selection of hyperparameters.

The hyperparameter choices were based on our specified hardware environment. In the empirical testing of hyperparameters, it was set as a system constraint to have 200 sampled examples in the initial training set, and select 40 examples for every iteration until 1000 examples were reached in total. We chose to hold the hyperparameters unchanged across the experiments and the datasets. It was observed by decreasing the epochs below our choice, the model trained on 200 initial labeled examples struggled to achieve a PC score above zero. Unfortunately, this resulted in unusable predictions for our query strategies in the beginning of active learning iterations and locked the potential of using active learning as a strategy. Thus, we chose to extend the empirical testing related to find a suitable number of epochs because it was a trade-off in achieving better initial PC scores by increasing the number of epochs at the expense of higher iteration time. In addition, these observations were affected by sizes and the architecture of the language models.

Kasai et al. [2019] used 20 epochs in their experiments and Li et al. [2020] used 10, 15, or 40 epochs dependent on the size of the datasets the model was tested on. Instead, after empirical testing, we trained TPLM based on RoBERTa on 20 epochs with a training and evaluation batch size of 32, and TPLM with DistilBERT was trained on 12 epochs with a training and evaluation batch size of 32. The larger language model RoBERTa required more training to be numerical stable in the experiments.

Additionally, our model used mainly default configuration settings based on Reimers and Gurevych [2019]. For the classification layer, we have chosen AdamW, Adam with Weight Decay, optimizer to a learning rate of 1e-3 and the weight decay to 0.01 [Loshchilov and Hutter, 2019].

Our proposed approach is independent of the choice of pretrained LM, and TopKDAL can potentially perform even better by using other pre-trained language models. More in-depth hyperparameter evaluations could have been performed with respect to using TPLM for blocking in a low resource setting, however, these investigations are considered as further work.

Active Learning. All experiments conducted 20 iterations of active learning, with a labeling budget of $\mathcal{S} = 1000$. We start the experiments with an initial labeled training set containing either (1) imbalanced set of 200 record pairs sampled randomly or (2) balanced set of $|\mathcal{T}_p| = 100$ positive and $|\mathcal{T}_n| = 100$ negative record pairs sampled randomly. In these initial training sets, the random sampling was performed by selecting pairs from

the training set $\mathcal{T}$ for the respective datasets. As explained previously, all results are averaged over their three experimental runs.

Top-k Search. We used top-k search with $k = 10$ to retrieve ten pairs of records with the highest cosine similarity score for each record in Table A and Table B. In our case, we implemented a top-k search to take $O(n^2)$ time as only small datasets were considered in the experiments. However, a top-k search can be implemented more time efficient than $O(n^2)$[Yang and Kitsuregawa, 2011].

Choice of Candidate Set Size. After empirical testing we achieved high PC score with a candidate set size based on a top-k search with $k = 10$. The size of candidate is an important factor that influences the overall PC of the system. A small candidate set can lead to low PC and a large candidate set can inadvertently result in low precision [Jain et al., 2021, Li et al., 2020].

Performance Validation. We validate the blocking performance with respect to pair completeness (PC) score and reduction rate as described in Section 2.6. PC score is computed based on matching record pairs retrieved by our proposed approach and the gold standard consisting of all matching record pairs in D_{test}.

Chapter 6

Results

This chapter presents the experimental results. First, the performance of our proposed blocking approach TopKDAL is showed for six informative query sampling strategies over active learning iterations. In addition, the model performance stability and time consumption are reported.

We have chosen to only describe the experimental results on the smaller, cheaper and faster TPLM DistilBERT. Results from RoBERTa experiments are added to show the challenge to train a larger TPLM to achieve a stable model with high performance. Our results and findings will be discussed in Chapter 7, and complete sets of the experimental results are attached in Appendix 8.2.3.

6.1 Performance of Query Sampling Strategies

All experiments have been validated over three experimental runs on the datasets according to Section 5.4.2, and the average result over these three runs are what is reported as results. Furthermore, the query sampling strategies have been deterministic seeded, which indicates the experiments initially are trained on identical randomly sampled examples. The objective was to aim for identical experimental setups and environmental run conditions for every experiment, resulting in training the experiments on the same randomly sampled examples in the initial training set at the same hardware. Consequently, we can observe from Figure 6.1 that all experiments with imbalanced training set have equal PC scores at 200 labeled instances in the plots. The corresponding starting points related to PC scores are seen for the experiments fed with the balanced initial training sets. Given the system constraint on 200 labeled instances in the initial training set size[1], the initial PC scores were only dependent on the ratio of positive and negative examples in the training sets and the TPLM preference, either DistilBERT and RoBERTa.

At the point of 200 labeled instances, the six tested query sampling strategies select records

[1]Our empirical analysis indicated $PCscore > 0$ for all experiments when 200 labeled instances were used as initial training set size.

to label. As a function of the respective sampling objectives for the query strategies as explained in Section 5.2, it is observed that their PC scores start to differ from baseline over AL iterations.

6.1.1 Comparison of Initial Training Set Class Distributions

At 200 labeled instances, we observe from Table 6.1 that using DistilBERT as TPLM trained on a pre-selected balanced initial training set performed best achieving the highest PC score in 5 out of 5 datasets. For DistilBERT as TPLM, the balanced initial training set consisting of 50/50 positive and negative labeled instances sampled randomly has on average 0.144% higher initial PC score across the datasets than the imbalanced initial training set. For our TopKDAL, the initial PC scores indicate satisfactory performance when using DistilBERT as TPLM, with PC scores varying between 0.325-0.975.

Table 6.1 showcases the initial PC scores gained from the two starting strategies before the active learning experiments start looping. In this low-resource setting, a balanced training set seems beneficial for kick-starting performance yielding a higher PC score and reducing the likelihood for cold start problems. This observation also shows how TPLMs should be trained initially in a low-resource setting, assuming the situation before active learning algorithms start working. An important question that arises from these observations is how those 50/50 initial training set distributions can be pre-selected with less data-hungry ML models combined with active learning. In that case, the performance gained from these imbalanced training sets of 200 instances is the benchmark to target.

6.1.2 Comparison of Query Sampling Strategies over 1000 Labeled Instances

First, several baselines are represented in the results. The active learning-based Baseline is colored blue in the plots. Baseline-Max indicates the PC score when training TPLM on all available training data, Baseline-1/2 denotes training on half of the available training data, and Baseline-1/4 uses 25 % of the training data. In all plots, these three non-active learning approaches are marked with red dashed lines. Baseline-Max results in the maximum PC score the model can achieve. Hence, it is also our benchmark target to reach or surpass during the active learning iterations.

Table 6.2 reports the performance after 20 active learning iterations based on evaluating our model on the test set D_{test} for each of the five datasets. For both starting strategies, Table 6.2 shows TopKDAL is robust enough to be competitive with Baseline. However, the performance for the query sampling strategies varies depending on initial training set distributions. TopKDAL outperforms Baseline over the active learning iterations when consuming as little as 1.3-14.6% of the total size of training datasets. This observation is further presented in Table 6.3. As expected, our AL methods improve their performance with an increasing number of examples over the active learning iterations. The final PC score for all the query sampling strategies is reported in Table 6.2. Two reasons cause the

Table 6.1: Snapshot of initial PC score at 200 labeled instances when trained on balanced or imbalanced initial training set using either DistilBERT or RoBERTa as TPLM. From the results, the balanced initial training set trained on DistilBERT outperforms the imbalanced set with on average 0.144 higher PC score. The highest scores with respect to TPLM are highlighted.

			Passive Query Sampling Strategies (QSS)
TPLM	Initial Training Set	Dataset	Random
DistilBERT	Imbalanced	Amazon-Google	0.822
		DBLP-ACM	0.950
		DBLP-Scholar	0.822
		Walmart-Amazon	0.832
		Abt-Buy	0.325
		Mean	0.750
	Balanced	Amazon-Google	**0.933**
		DBLP-ACM	**0.975**
		DBLP-Scholar	**0.872**
		Walmart-Amazon	**0.933**
		Abt-Buy	**0.759**
		Mean	**0.894**
RoBERTa	Imbalanced	Amazon-Google	0.410
		DBLP-ACM	0.966
		DBLP-Scholar	0.784
		Walmart-Amazon	0.484
		Abt-Buy	0.202
		Mean	0.5692
	Balanced	Amazon-Google	**0.739**
		DBLP-ACM	**0.985**
		DBLP-Scholar	**0.917**
		Walmart-Amazon	**0.611**
		Abt-Buy	**0.430**
		Mean	**0.7364**

improvements: (1) the ability of TopKDAL to leverage the learned semantic relationships in the input data, and (2) a model architecture customized for maximizing PC score.

Two best performing query sampling strategies. We find that Imbalanced-Partition-2 and Balanced-Uncertainty provide gains over native random query sampling as a baseline strategy and beat all other strategies by a significant margin to show their effectiveness as query sampling strategies. Imbalanced-Partition-2 yields on average 0.027 higher PC-score than Baseline after 1000 labeled instances when the model is trained on an imbalanced training set. Given a balanced initial training set, Balanced-Uncertainty is the best performing query sampling strategy achieving on average 0.018 higher PC score than Baseline.

Imbalance vs. balance starting strategy at 200 labeled instances. We wanted to investigate how the varying class ratio of training data impacts TopKDAL performance. We used a positive to negative ratio of 1:1 in the initial training set as a counterpart to the imbalanced training set for this understanding. From the results at 200 labeled instances, the balanced set yielded on average with a 0.144 higher PC score than the imbalanced training set. Therefore, it tends to be beneficial to aim for an initial training set consisting of a 50/50 ratio between positive and negative examples, at least in situations where the model struggles initially to achieve a PC score > 0 corresponding to cold start problems. Balanced starting strategy trained the model initially with a true positive rate of 50%, and the idea was to maintain TPR over AL iterations. Opposite, the imbalanced starting

strategy trained the model with TPR randomness, and idea was to increase the TPR over AL iterations. TPR progress related to query sampling strategies are further discussed in Section 7.2.3.

Imbalanced vs. balanced starting strategy at 1000 labeled instances. At the beginning of the active learning process, we observe from Figure 6.1 that query sampling strategies with imbalanced starting strategy are not performing at a similar level as balanced starting strategy. As the number of examples increases, we observe that these strategies catch up with the approaches with the balanced started strategy and yield decent performance as showed in Table 6.2. This phenomenon can be attributed to the fact that randomly sampling an imbalanced training sets may not be the most informative ones, and as a result, the learned decision boundary tends to be inaccurate. However, as the number of examples in the training set increases the balanced starting strategy tends to lose its edge, the initial difference in PC score of 0.144 between the balanced and the imbalanced training set reduces to 0.012 after the AL iterations were finished. Over the active learning iterations, Figure 6.2 shows that the imbalanced starting strategy yields converging model stability while a balanced starting strategy continue to fluctuate. At 1000 labeled instances, 3 out of 6 query strategies yield a higher PC on average with pre-selecting an imbalanced instead of an initial balanced training set.

Imbalanced-Partition-2 can be pointed out to improve its PC score more rapidly and converges faster given an imbalanced initial training set, as showed in Section 8.2.3. At the same time, it can be seen that the other query strategies had a slower performance improvement in PC score. With a balanced starting strategy as showed in Figure 6.1, we cannot see the same progress related to Partition-2. One subtle problem we encountered is how to sample an initial training set with 50/50 distribution in practice that is not detrimental to learning good embeddings for blocking.

Models trained on a balanced distributed training set prefer uncertain examples to improve their performance rather than high confident examples. In comparison, a model trained on an imbalanced distributed training set gains more reward and better performance using partition sampling as a query sampling strategy. In both cases, high confident positive and negative examples seem to be the query sampling strategy having the most negligible advantageous impact on the model compared to the other AL strategies. Further experiments showed that P-4-3xPB does improve the model performance of TopKDAL compared to Baseline, when boosting the selected number of positive examples. However, P-4-3xPB achieves still insignificant impact on the model performance when it is compared to Imbalanced-Uncertainty and Balanced-P-2.

Random-P/N - A counterpart against the other query strategies. As a counterpart against the other query strategies, Random P/N strategy was chosen to wipe away the informativeness gained from selecting positive and negative least confident and high confident examples. As observed, Random-P/N based on the imbalanced set achieves higher PC score than Baseline, but it achieves lower PC score than Baseline on the balanced set. However, this unexpected contraction was marginal with 0.003 lower PC score than Baseline. We believe the success of Imbalanced-Random-P/N is due to the principle of choosing unlabeled examples that are both informative and representative, and as a result, the learned decision boundary tends to be accurate.

Outstanding balanced kick-start resulted in decreasing PC score from 200 examples in Walmart-Amazon. In dataset Walmart Amazon, Figure 6.1e, we can surprisingly observe a decreasing PC score between 200-1000 examples after being kick-started with a balanced starting strategy ever after running these experiments three times. The model do not manage to maintain the initial excellent performance over the active learning strategies, indicating that the model initially has been trained too confident in retrieving doubt cases into the candidate set instead of filtering out false negatives. However, it is important to remember that the model might be affected by a class distribution mismatch between an initial balanced training set and imbalanced test set. This result is a reminder that training the model with an increasing number of labeled examples does not necessarily yield a higher performance reward.

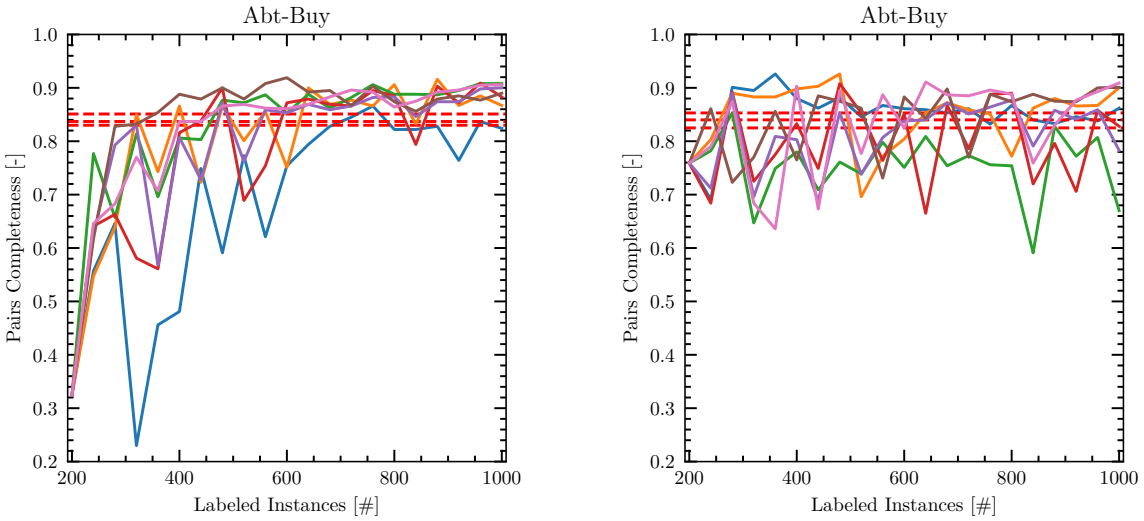
High reduction rate achieved. Reduction rate varies approximately between 0.90-1.00 across the datasets, which significantly reduces the size of the candidate set. Combined with the resulting high PC scores in the outputted candidate sets for every dataset, a decent collection of potential matching record pairs is prepared before starting the matching step.

No query sampling strategy achieved 100% PC score on hard datasets. It might be caused by several factors such as incorrect labeled data in test sets and/or the challenge to identify the most difficult true positive examples for the model. Here, it is important to investigate why these remaining positive examples are difficult to learn for the model in further work.

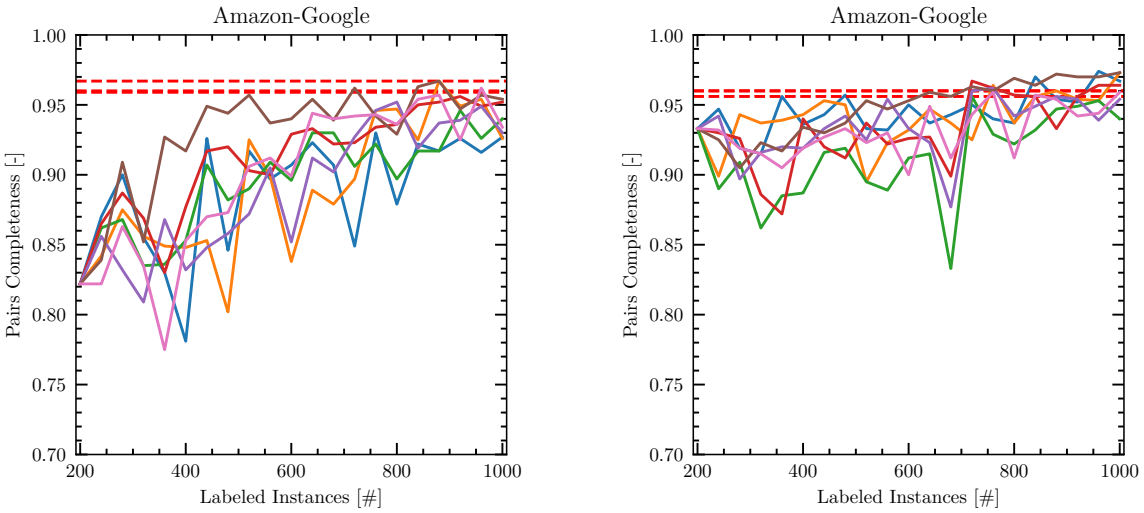
Blocking performance on hard datasets. TopKDAL seems to need several iterations to reach Baseline-Max for the hard datasets, which seems reasonable because the model needs to perform a sufficient number of gradient steps as the added linear layer on top of transformer architecture is uninitialized.

Table 6.2: Snapshot of final PC score after 1000 labeled instances tested with DistilBERT or RoBERTa as TPLM for balanced or unbalanced initial training set. The table shows a comparison of the tested query sampling strategies after 20 iterations of active learning, in which the strategies with the highest PC score is highlighted. The following abbreviations are used: HC-P/N = High Confident Positives and Negatives, Random-P/N = Random Positives and Negatives, P-4 = Partition-4, P-2 = Partition-2, and P-4-3xPB = Partition-4 3xPosBoost. Section 5.3 describes each query sampling strategy in detail.

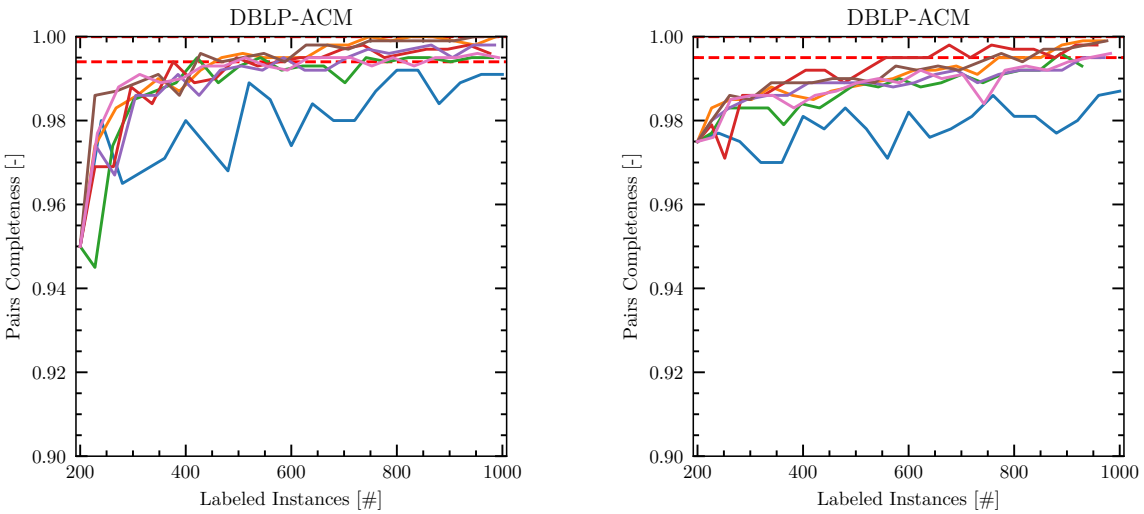
			Passive QSS	Active Query Sampling Strategies (QSS)					
TPLM	Initial Training Set	Dataset	Baseline	Uncertainty	HC-P/N	Random-P/N	P-4	P-2	P-4-3xPB
DistilBERT	Imbalanced	Amazon-Google	0.927	0.926	0.940	0.952	0.930	**0.954**	0.934
		DBLP-ACM	0.991	**1.00**	0.995	0.996	0.998	**1.00**	0.995
		DBLP-Scholar	0.962	0.989	0.885	0.989	0.979	**0.993**	0.988
		Walmart-Amazon	0.862	0.877	0.869	0.872	**0.891**	0.865	0.865
		Abt-Buy	0.824	0.867	**0.908**	0.883	0.900	0.890	0.906
		Mean	0.913	0.932	0.919	0.938	0.939	**0.940**	0.938
	Balanced	Amazon-Google	0.967	**0.973**	0.940	0.964	0.954	**0.973**	0.959
		DBLP-ACM	0.987	**0.999**	0.993	0.998	0.995	**0.999**	0.996
		DBLP-Scholar	0.971	**0.992**	0.924	0.983	0.976	0.988	0.979
		Walmart-Amazon	0.883	0.895	0.869	0.883	**0.908**	0.876	0.884
		Abt-Buy	0.862	0.900	0.670	0.828	0.783	0.901	**0.909**
		Mean	0.934	**0.952**	0.879	0.931	0.923	0.947	0.945
RoBERTa	Imbalanced	Amazon-Google	0.855	0.603	0.88	0.882	0.85	**0.905**	0.875
		DBLP-ACM	0.985	0.992	0.989	0.996	0.994	**0.998**	0.997
		DBLP-Scholar	0.596	0.665	0.845	0.986	0.979	**0.991**	0.983
		Walmart-Amazon	0.732	**0.853**	0.546	0.838	0.475	0.288	0.846
		Abt-Buy	0.091	0.309	0.675	0.505	**0.914**	0.595	0.608
		Mean	0.652	0.684	0.787	0.841	0.842	0.755	**0.862**
	Balanced	Amazon-Google	0.793	0.895	0.858	0.798	0.876	**0.916**	0.601
		DBLP-ACM	0.980	0.996	0.988	0.995	0.993	**0.997**	0.994
		DBLP-Scholar	0.916	0.661	0.907	0.965	**0.987**	0.985	0.973
		Walmart-Amazon	0.831	0.223	0.865	0.570	0.582	**0.870**	0.463
		Abt-Buy	0.589	0.505	0.828	0.90	0.424	**0.921**	0.788
		Mean	0.822	0.656	0.889	0.846	0.772	**0.938**	0.764



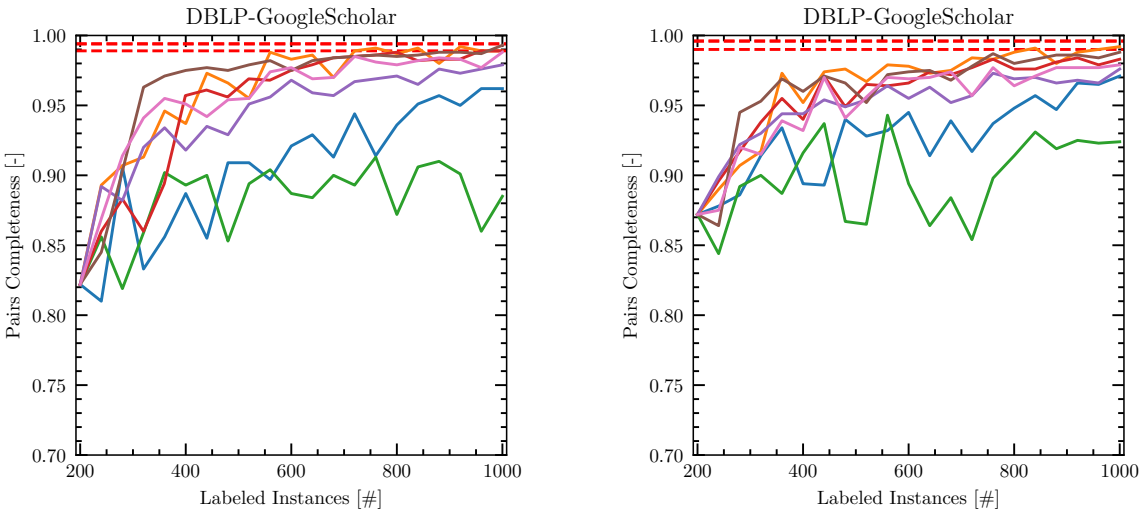
(a) Imbalanced(left) and balanced(right) initial training set with DistilBERT: PC score with respect to labeled instances for Abt-Buy.



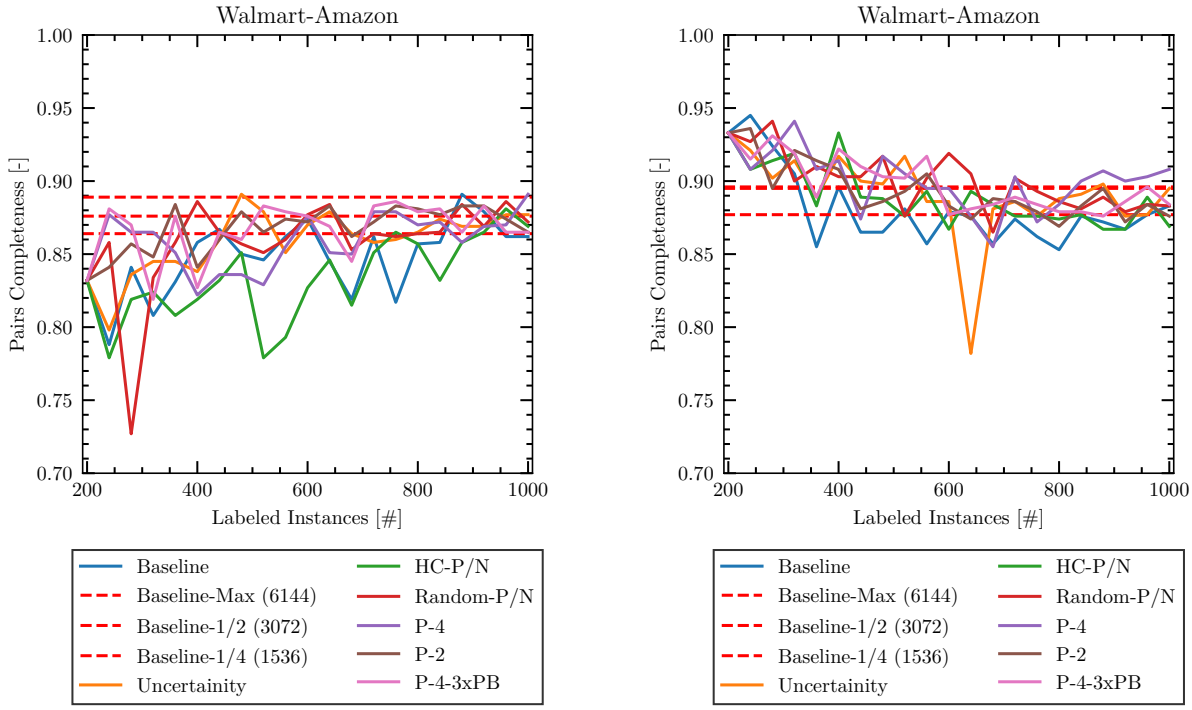
(b) Imbalanced(left) and balanced(right) initial training set with DistilBERT: PC score with respect to labeled instances for dataset Amazon-Google.



(c) Imbalanced(left) and balanced(right) initial training set with DistilBERT: PC score with respect to labeled instances for dataset DBLP-ACM.



(d) Imbalanced(left) and balanced(right) initial training set with DistilBERT: PC score with respect to labeled instances for dataset DBLP-GoogleScholar.



(e) Imbalanced(left) and balanced(right) initial training set with DistilBERT: PC score with respect to labeled instances for dataset Walmart-Amazon.

Figure 6.1: PC score after each active learning iteration based on TopKDAL performance with the different query sampling strategies. By evaluating DistilBERT as TPLMs on balanced and imbalanced initial training sets, it seems TopKDAL with DistilBERT as TPLM trained on balanced initial training set outperforms the other combinations after 1000 labeled instances. Initially, all experiments started with 200 sampled instances. Then, the query strategies aimed to label 40 examples in each active learning iteration as explained in Section 5.2.1. The red dashed lines show Baselines with non-active learning applied with 25%, 50%, and 100% of the training set size. The total number of examples in the training set is showed inside the parenthesis.

Query Sampling Strategies Compared to Baseline-Max

Table 6.3 shows that several query sampling strategies reached or surpassed the Baseline-Max after a significantly lower labeling budget than the size of the entire training set. As seen from Abt-Buy, Partition-2 reached Baseline-Max after 360 labeled examples for the imbalanced training set and after 260 labeled examples for the balanced training set, which corresponds to only 2.1 % and 1.5% of the total dataset size for Abt-Buy, respectively. These results indicate the advantage of starting with an initial balanced training set distribution as a starting strategy in a low resource setting.

From Table 6.3, Partition-2 and Uncertainty seem to be the query sampling strategies with the highest probability of outperforming Baseline-Max, where Balanced-Partition-2 often needs fewer labeled instances than Imbalanced-Uncertainty. Our TopKDAL demonstrates its effectiveness in reaching PC scores of Baseline-Max, especially on the hard datasets Amazon-Google, Walmart-Amazon, and Abt-Buy. These intersection points between the query sampling strategies and Baseline-Max are showed in Figure 6.1 and Section 8.2.3, where Baseline-Max is denoted with dashed red lines. On the easy datasets, DBLP-ACM and DBLP-Scholar, the query sampling strategies often require more than 1000 examples to outperform Baseline-Max performance. Hence, active learning as a strategy seems to have a higher likelihood to reach Baseline-Max performance on hard datasets within the labeling budget.

Table 6.3: Labeling effort analysis based on active learning and DistilBERT as TPLM: DistilBERT and active learning query sampling strategies are compared with Baseline-Max to evaluate the number of labeled instances required to reach Baseline-Max PC-score. The query strategy with the highest reduction in the number of labeled instances required to reach the performance of Baseline-Max is highlighted. The percent of required training set size over the total training set size is denoted in the parenthesis.

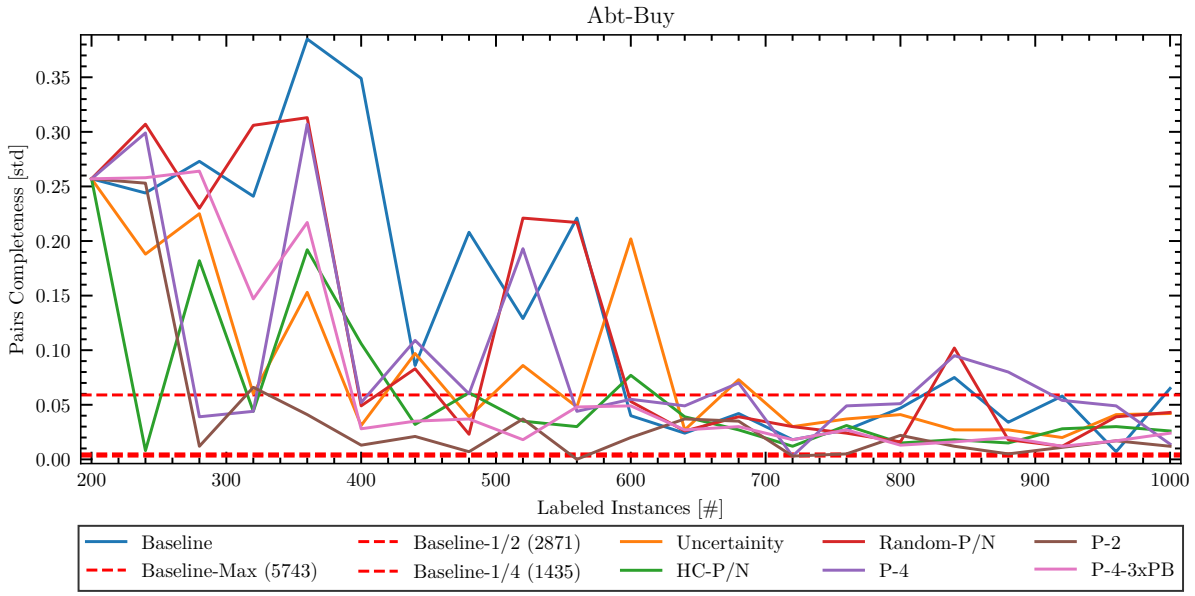
			Passive QSS	Active Query Sampling Strategies (QSS)					
Initial Training Set	Dataset	Baseline-Max	Random	Uncertainty	HC-P/N	Random-P/N	P-4	P-2	P-4-3xPB
Unbalanced	Amazon-Google	6874	-	880 (12.8%)	-	-	-	**840** (12.2%)	-
	DBLP-ACM	5743	-	840 (14.6%)	-	-	-	**420** (7.3%)	-
	DBLP-Scholar	6144	-	-	-	-	-	-	-
	Walmart-Amazon	7417	880 (11.7 %)	**480**(6.5%)	-	-	-	-	-
	Abt-Buy	17223	740 (4.3 %)	400 (2.3%)	480 (2.7%)	445 (2.6%)	480 (2.8%)	**360** (2.1%)	460 (2.7%)
Balanced	Amazon-Google	6874	825 (12.0%)	760 (11.1%)	-	720 (10.5%)	720 (10.5%)	**700** (10.2%)	760 (11.1%)
	DBLP-ACM	5743	-	-	-	-	-	-	-
	DBLP-Scholar	6144	-	-	-	-	-	-	-
	Walmart-Amazon	7417	**200** (2.7%)	**200** (2.7%)	**200** (2.7%)	**200** (2.7%)	**200** (2.7%)	**200** (2.7%)	**200** (2.7%)
	Abt-Buy	17223	270 (1.57%)	270 (1.57%)	295 (1.7%)	275 (1.3%)	280 (1.6%)	**260** (1.5%)	280 (1.6%)

6.2 Model Performance Stability

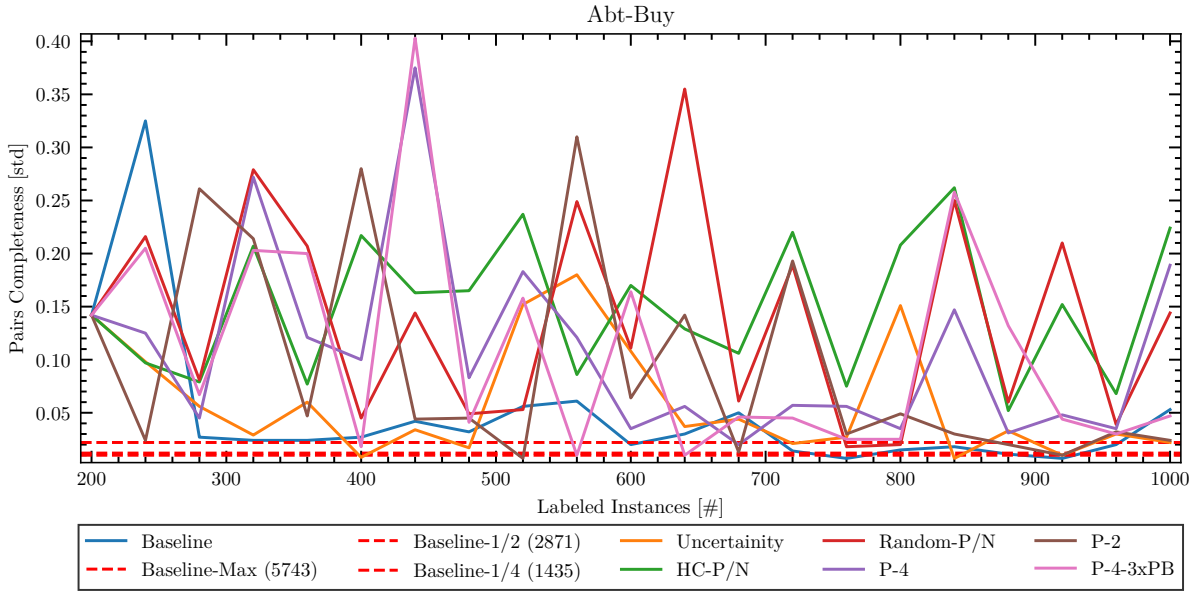
Figure 6.2 shows the standard deviation of PC score for Abt-Buy. We have performed three experimental runs to compute standard deviation, where the random seed was modified for each run according to the procedure explained in Section 5.4.2. As expected, we can observe that Baseline and Random-P/N have the highest standard deviation after 2-3 AL iterations for both starting strategies. The PC-score standard deviations achieved

for the non-active learning Baseline-Max, Baseline 1/2, and Baseline-1/4 are lower and more stable than the active learning strategies' models.

At 200 labeled instances, the balanced initial training set has a lower standard deviation of PC score for 5 out of 5 tested datasets than an imbalanced initial training set. After 200 labeled instances, the model stability converges over the AL iterations. Surprisingly, the model stability seems to be significantly more stable for query sampling strategies with imbalanced seed, and the query sampling strategies themselves are more dependent on the initial class distribution than expected regarding model stability. This observation assumes that the query sampling strategies obtain to select 50/50 distributions over every AL iterations. If not, the observed effects might be caused by decreasing true positive rate (TPR) for the balanced training set and increasing TPR for the imbalanced training set over the AL iterations. This effect is discussed in Chapter 7. The Abt-Buy dataset shows TopKDAL converges more rapidly to a PC score standard deviation below 0.05 for the imbalanced initial training set. Query sampling strategies started with a balanced initial training set violate more in amplitude, even at 1000 labeled instances for HC-P/N, P-4, and Random-P/N. We can observe that the query sampling strategies using the imbalanced initial training set as starting strategy reached the same stability as Baseline-1/4 at approximately 700 labeled instances. Additional results of the PC score standard deviation can be found in Section 8.2.3.



(a) Unbalanced random initial training set with LM DistilBERT: Standard Deviation of Pair completeness with respect to labeled instances for the dataset Abt-Buy.



(b) Balanced 50/50 initial training set with LM DistilBERT: Standard deviation of Pair completeness with respect to labeled instances for the dataset Abt-Buy.

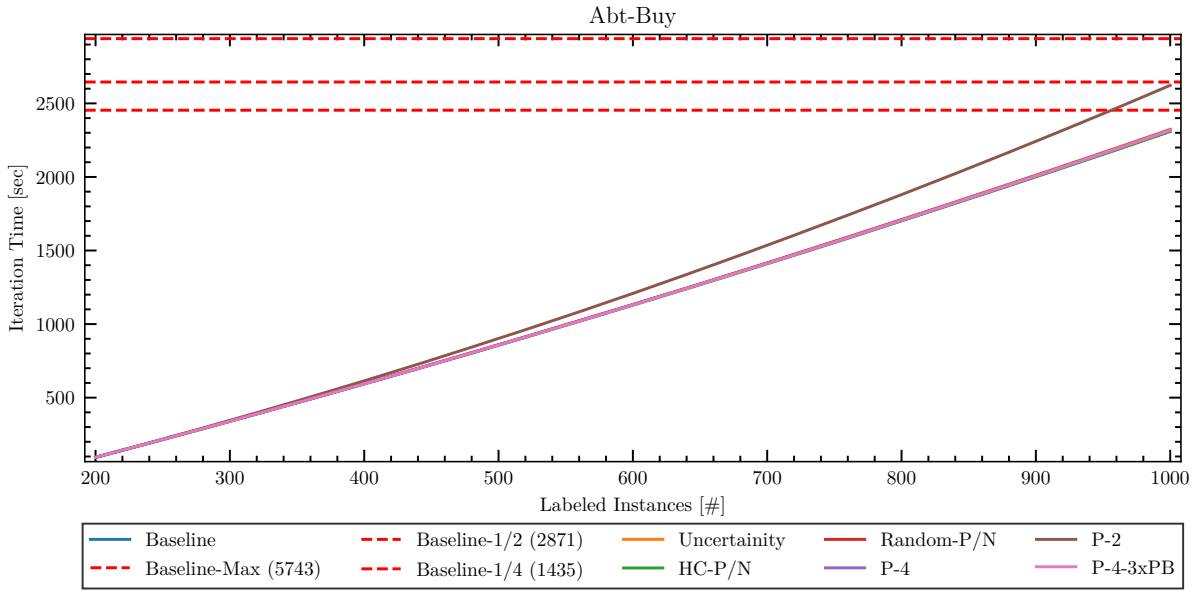
Figure 6.2: PC score standard deviation over an increasing number of labeled instances. Query sampling strategies use DistilBERT as TPLM. At 200 labeled instances, the balanced initial training set achieves a lower PC score standard deviation over the active learning iterations in 5 out of 5 datasets. After 700-1000 labeled instances for Abt-Buy, the imbalanced initial training set seems to stabilize below 0.05, while the balanced initial training set violates more in amplitude over the AL iterations. The standard deviation has been computed from 3 experimental runs. PC score related to different query sampling strategies are explained in Section 6.1.2, and the overall performance is reported in Table 6.2.

6.3 Iteration Time Consumption

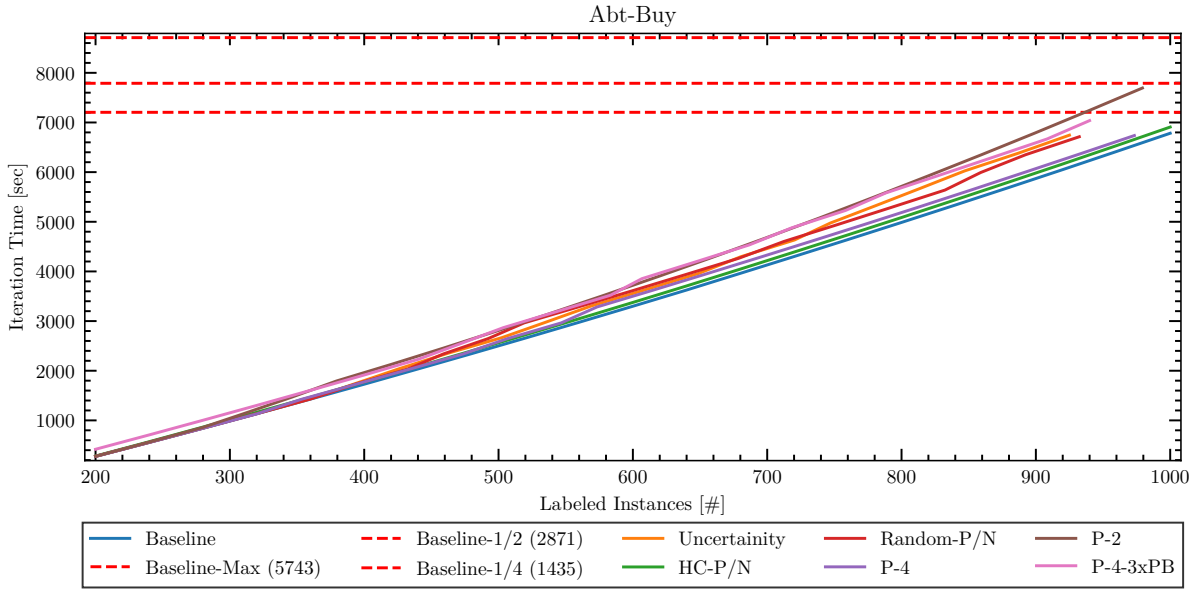
Our proposed TopKDAL demonstrates computational efficiency by reducing the time complexity from $\Theta(n^2)$ to a linear running time $O(n)$ for all datasets tested, independent of which query sampling strategy was applied. From the experimental results, we observe that both DistilBERT and RoBERTa behave quite similar over AL iterations. P-2 strategy has often more rapidly increase in the iteration time compared to the other query strategies. All experiments achieve linear running time over AL process although the top-k search computes similarity values for all record pairs in the training set. It indicates that the quadratic complexity caused by top-k search is insignificant compared to the training time itself.

As described in Chapter 5, we emphasize that the iteration time consumption reported after each iteration in the plots represent the cumulative training time. In the plots, we have excluded the time usage to setup the initial training sets to only show time usage over the active learning iterations. For all datasets, it was not found any significant deviation in time usage between the two starting strategies sampling examples into the initial training set. Each iteration time represents the time to train the model on all examples at the particular active learning iteration instead of training the model on only the new record pairs added to the updated training set. The learning model was consequently restarted and retrained after each active learning iteration. For the Oracle, iteration time indicates the waiting time before start query the Oracle to label the new selected examples.

As expected, RoBERTa was slower than DistilBERT caused by a larger language model requiring a higher number of epochs and more training data to achieve a stable model with high accuracy. Since the size of language model DistilBERT is pre-trained with approximately 10 times less training data than RoBERTa, it also affects the iteration time significantly. For dataset Abt-Buy, the iteration time varies between 2-3 minutes for DistilBERT and between 5-7 minutes for RoBERTa. It indicates around 1.5x-3.5x higher iteration time for RoBERTa compared to a lighter and faster LM as DistilBERT. The time usage related to our experiment runs seems reasonable as we also chose exclusive privileges to avoid shared resource loads in the test environment with others. Despite all experiments being run on the test environment by using exclusive privileges, as explained previously, the iteration time consumption could have been affected by minor load disturbances in the test environment. Hence, these results should be interpreted keeping this in mind. A more detailed description of our experimental setup can be found in Section 5.4.2. In many datasets, RoBERTa as TPLM struggled to reach 1000 examples due to predicted positive examples in the unlabeled pool was exhausted. For RoBERTa, this effect seems to be a bigger issue for the imbalanced starting strategy than the balanced starting strategy, especially for the datasets Abt-Buy and Walmart-Amazon. It might indicate that RoBERTa requires an even higher number of epochs to improve the model's predictions. Section 8.2.3 shows the iteration time when the imbalanced starting strategy is applied to DistilBERT and RoBERTa for the respective datasets.



(a) LM DistilBERT with the balanced starting strategy: Iteration time with respect to labeled instances for dataset Abt-Buy at epochs=12. Section 8.2.3 shows the iteration time when the starting strategy is replaced by random initial strategy.



(b) LM RoBERTa with the balanced starting strategy: Iteration time with respect to labeled instances for dataset Abt-Buy at epochs=20. Section 8.2.3 shows the iteration time with random starting strategy.

Figure 6.3: Iteration time comparison between DistilBERT and RoBERTa at balanced starting strategy. The iteration time in the graphs represent the cumulative training time indicating an iteration time between 2-3 minutes using DistilBERT and between 5-7 minutes for RoBERTa. The learning model was consequently restarted and retrained after each active learning iteration. All examples in the training set were used during retraining instead of training the model on only the new record pairs in the updated training set. The model demonstrates computational efficiency by reducing the time complexity from $\Theta(n^2)$ to a linear running time $O(n)$ for all datasets tested.

Chapter 7

Discussion

This chapter discusses our findings and observations related to the results obtained from our TopKDAL approach.

7.1 Application of Transformers in Blocking

The first research questions address how TPLM can be leveraged for enabling efficient hands-off blocking tasks in entity matching. This blocker strategy component is also essential to prepare for unlocking the potential of using active learning as a strategy, which up-to-now has not been applicable in combination with active learning strategies.

7.1.1 Model Architecture Trade-offs

Table 7.1 lists several interesting identified strengths and limitations related to our approach TopKDAL. We chose to fine-tuning the transformers built on pre-trained DistilBERT or RoBERTa for the task based on labeled data, as it updates all attention weights and layers in TPLM network, for achieving a versatile approach with high performance. As a result, it yields useful sentence embeddings and reduces significantly the training time in our blocking strategy, in contrast to if we have used BERT out-of-the-box[Reimers and Gurevych, 2019].

Our model architecture was constructed for semantic similarity search as well as providing probability distribution capacity for doing predictions. The purpose of using similarity search was to efficiently enable evaluation of similarity measures on all pairs of records corresponding to the Cartesian Product. In the networks, every record was encoded to derive semantically meaningful record embeddings[1]. While BERT uses attention to compare directly both sentences (records), e.g. word-by-word comparison [Reimers and Gurevych, 2019], TopKDAL mapped each record from an unseen topic to a vector space

[1]With semantically meaningful it means that semantically similar sentences are closer in vector space.

where similar record pairs were closer and dissimilar pairs were further apart. We always used cosine-similarity score as similarity measure to compare the similarity between two sentence embedding, according to Reimers and Gurevych [2019].

Among these record pairs, a top-k approximate nearest neighbor search was performed for every record. The k most similar record pairs with the highest cosine similarity score from each record were added to the candidate set. The drawback of this semantic similarity search related to informative active learning strategies is the lacking capability for making probability predictions, an essential component to apply informativeness-based active learning strategies. To solve this challenge, we integrated Softmax in TopKDAL to provide probability predictions for the record pairs retrieved for the respective records in the training set. However, we have to recall that our benchmark datasets were pre-blocked, in which means the benchmark datasets only consist of a labeled subset compared to size of all record pairs in the Cartesian Product. To mimic a realistic blocking setting for evaluating our TopKDAL, it was fundamental to generate a Cartesian Product between every record in the training set at each AL iteration. On the other hand, this introduced another drawback in terms of a training set consisting of both labeled and unlabeled record pairs caused by the approximate nearest neighbor top-k search. As previously described, the query sampling algorithms used in the active learning loop could only handle labeled record pairs to mimic an human-in-the-loop. Consequently, it was necessarily to design a solution that filtered out the unlabeled record pairs before we could applied a query sampling strategy to select a set of examples at each iteration. However, it was not investigated to obtain any advantages by using unlabeled record pairs.

Towards explainable blocking and the important aspect to build responsible technologies, TopKDAL has not to date the capability to explain the reason two records have obtained matching outcome in the candidate set, it applies to both true positives and false positives. For the application users, it might be challenging to interpret and explain why TopKDAL made a particular prediction. In datasets where the TopKDAL achieves low PC score, it is essential to understand why the model is predicting the outcomes incorrectly between the record pairs. Thirumuruganathan et al. [2019] investigated the existing gap between explaining classifier prediction methods and EM, and they stated that a direct application of those methods are not suited for EM. We believe that methods explaining classifying predictions made in EM should have the capability to describe a global explanation and local explanation. The global explanation should have the capability to explain how matching decisions in TopKDAL are performed in general, and a local individual explainability for outcomes in the candidate set such as why one individual is denoted as a potential candidate for mortgage[Lundberg et al., 2019]. A question to address is whether the blocking step and/or the matching step should present user-friendly textual explanations or identify relevant facts related to the outcomes. However, we can guess that an individual that is incorrectly filtered out caused by the blocking step and rejected to obtain the mortgage will ask for justification due to the refusal. Another question that can be raised is how TopKDAL can guarantee algorithmic fairness if unwanted biases are implicitly captured in deep learning models from the training data.

Lastly, we attribute the computational efficiency of TopKDAL to TPLM architecture, achieving linear running time $O(n)$ for all benchmark datasets despite the problem complexity is $\theta(n^2)$. One can reason that model training time dominates the total time usage

while the quadratic top-k search time (see comment in Section 5.4.2) is insignificant compared to model training time. In practice, a low model training time corresponds to a shorter waiting-time for Oracle and more frequently beneficial labeling of new examples by Oracle.

Table 7.1: Identified strengths and limitations in our approach TopKDAL.

Strengths	**Limitations**
Prepared for informative active-learning query sampling strategies such that labeled training data is not an impedance for the adoption of such a blocking approach. Model architecture provides probability predictions to identify informative examples in the unlabeled training dataset	Sub-optimal configured to handle doubt cases in the embedding space due to Softmax function affects the embeddings negatively.
Versatile blocking approach for different pre-trained LMs to be fine-tuned using labeled data. Pre-trained LMs derive semantically meaningful sentence embeddings, resulting in immensity of language understanding and ability to learn where to pay attention between the records. TopKDAL is easy to apply to different domains, whether it is about bibliographic or product data. Apart from changing base transformer to multilingual BERT base model and constructing labeled seed sets, the model architectures of TopKDAL might remain the same to perform blocking on multilingual datasets[2][Jain et al., 2021].	Not *responsible and explainable approach* yet. There are no capability to explain the reasons to matching or non-matching outcomes between two records[Stoyanovich et al., 2020, Monroe, 2018]. In practice, the blocking step can impact the lives of people as it can incorrectly eliminate acceptable individuals from receiving a mortgage or job seekers are not further considered as job seekers for a job. In blocking, any automation related to candidate set selection in mortgage or hiring processes should be explainable to obtain trust and fairness for the applicants.
Tunable blocking performance. Size of retrieved candidate set can be varied by increasing or decreasing approximate k nearest neighbors using top-k search. Smaller TPLM or faster hardware are preferred to reduce the iteration time.	Weak query sampling performance if modeled is not trained sufficient which can amplify the amount of non-matching predictions and further contribute to deterioration in the model performance. As a result, the blocking step can miss out on important regions in the product space leading to low PC score.
Demonstrates computational efficiency achieving linear running time $O(n)$ for all benchmark datasets tested, even though the problem complexity is $\theta(n^2)$. Model training dominates the time usage.	No transfer learning adaption from other datasets supported yet to kick-starting TopKDAL, i.e. no transferable representations from multiple source datasets, neither with or without active learning.
TopKDAL with TPLM and active learning demonstrates promising PC score with respect to labeling effort in a low-resource setting.	TopKDAL do not guarantee *algorithmic fairness* caused by unwanted biases can be implicitly captured in deep learning models from the training data.

7.1.2 Comparison between TopKDAL and Traditional Blocking Methods

Traditionally, performing blocking on a dataset requires the assistance of domain experts to identify a number of hand-crafted tuning and design choices. This can involve manual tasks such as selection of appropriate blocking functions, attribute selection, feature engineering, selection of appropriate similarity functions and thresholds, hyperparameter tuning for ML models, and so on. To exemplify its challenge, blocking rules can agree about the attribute values in 2-3 attributes, and the comparison between these attributes seems realistic. However, the attribute values can be very dissimilar in the other attributes. In addition, traditional blocking methods do not capture lexical patterns as obtained from the attention vectors in the shallow layers of transformer-based models, or syntactic and semantic meaning which is determined in the deeper layers. Last, but not least, prior blocking methods are hard to tune for maximizing pairs completeness score (i.e. recall). That said, the common denominator of the traditional blocking methods has been their obstacle to leverage any of the configurations obtained on one dataset and further apply these setups as-is to new datasets.

We can observe without much difficulty that TopKDAL for blocking obviates many of these issues. There are several beneficial properties compared to traditional blocking methods. First, we enable to reduce the involvement of domain experts from the tasks described above, and TopKDAL requires minimal input from the domain expert, except for labeling examples during AL loop. Secondly, TopKDAL has significant flexibility to encode the records as they are proposed in the actual schema of datasets. Transformer architecture does not require data preprocessing such that two records need to have the same schema or to perform schema matching before encoding the records. Instead, TopKDAL learns what is the important attributes for blocking automatically without manual feature engineering, and any inconsistencies in the datasets such as corrupted and missing values are also handled. Lastly, TopKDAL is a solution that handles incomparable record pairs between datasets due to its deeper language understanding, infeasible for rule-based blocking methods. Towards hands-off blocking, TopKDAL demonstrates a versatile blocking approach working across different datasets and domains as frictionless as possible. As the best of our knowledge, it is not published any work or experimental results directly comparable to our approach TopKDAL on the benchmark datasets described in Chapter 4.

7.1.3 Comparison between using Transformers in Blocking vs. Matching in Entity Matching

In entity matching, TopKDAL shares some similarities with the matching step. However, there are several differences as highlighted in Table 7.2.

In blocking, we ideally want to avoid doing binary classification with Softmax, and instead, rank similar record pairs to achieve a high pair completeness score. Softmax is a classification loss that wants, in this case, the cosine similarity to be as low as possible when record pairs do not match and as high as possible when the ones match. Thus, the

ideal is that embeddings to matching records are similar and that embeddings to those that do not match are as far apart as possible. Softmax tends to squeeze away nuances in how similar record pairs are. It is essential to preserve semantically similarity inputs close together in the embedding space to achieve a better decision basis.

As described in Section 5.1.1, our proposed TopKDAL shows the challenge to mimic the actual similarity between record pairs in the model architecture when combining TPLM with informative active learning query sampling strategies. When leveraging Softmax to normalize the output of a network to a probability distribution over the predicted output classes, it affects the doubt record pairs negatively.

As explained previously, triplet margin loss function determines the relative similarity existing between records, in which provides several enhancements related to handling doubt cases in the embedding space. On the other hand, the challenge related to triplet loss is difficulty to find the threshold separating matches from non-matches for each dataset. This results in difficulty to combine triplet margin loss function and informative active learning strategies. Hence, we need to design strategies that can identify the threshold separating matches from non-matches to unlock the potential of combining triplet loss with active learning strategies. With that in mind, we can attempt to design suitable active learning strategies for triplet margin loss.

Table 7.2: Comparison of the properties between Blocking vs. Matching.

Blocking	Matching
Fast approximate nearest neighbor top-k search implementation required to reduce candidate set size efficiently, co-embed matches that correspond to the same real-world entity and pruning away unlikely pairs from the Cartesian product.	Binary classification task-objective. Hence, no top-k search is required. All candidate pairs are classified to either matches or non-matches.
In a blocking setting, Softmax functionality is leveraged to normalize the output of a network to a probability distribution over the predicted output classes and unlock the potential using active learning as a strategy. However, Softmax affects the doubt record pairs negatively. Soft labels is often necessarily to set lower target above 0 for mimic some similarity present between all pairs in the real-world datasets.	Softmax functionality can be used as-is. No soft labels required to change the lower or upper target for non-matches or matches, respectively, because transformations of embedding do not effect negatively to the task objective to separate precisely non-matching from matching record pairs.
In blocking step, maximizing PC score is not always the situation. To exemplify, a crime investigation aims to match individuals to a large databases of people, a high PC would be desired to increase the likelihood that potential criminals are included for investigation. On the other hand, in public health studies aiming to find matches corresponding to a patient with certain medical conditions, a high PQ score required to only include patients that do have the medical condition under study [Christen, 2012a]. PC and PQ are negatively correlated in many applications.	In matching step, maximizing F1 score is often the task-objective. Therefore, finding the correct blocking algorithm is critical for many EM applications as the matching step is affected by the candidate set from the blocking step.
Individually input record representation obtain its encoding(*single mode*).	Concatenated input record representation of r and s to obtain candidate pair (r, s), a joint representation encoding(*paired mode*).

7.2 Active Learning as a Strategy in Blocking

Research question 2 is related to how active learning with transformers perform for blocking as a function of labeling effort, in which the labeling effort involves to investigate the number of labeled instances required.

We have highlighted how our proposed TopKDAL, combining a blocking strategy with an active learning strategy, improves the PC score compared to non-active learning based Baseline-Max and active learning based Baseline. TopKDAL demonstrate its capability of achieving PC scores at a similar level based on a limited labeling budget of maximum 1000 labeled instances, and simultaneous achieving a reduction in the total time consumption using DistilBERT instead of RoBERTa as TPLM.

7.2.1 Active Learning with Transformers in a Low-Resource Environment

As discussed previously in Chapter 5, we adopted uncertainty sampling and partition sampling as existing methods for our active learning strategies. In addition, we tested our three new active learning approaches. Here, we show some of the effects considering these methods for a balanced starting strategy observed to have more significant model instabilities than the imbalanced starting strategy. All query sampling strategies had the same initial balanced training set of positive and negative examples and accordingly have the same initial PC score.

Uncertainty and HC-P/N strategies impact on the model decision boundary. Figure 6.1 shows TopKDAL performance on least confident sampling (Uncertainty) and high confident sampling (HC-P/N) contribute to the volatile performance in PC score over the AL iterations, such as a sudden drop in PC score at 650 labeled examples for the hard dataset Walmart-Amazon and at 680 labeled examples for the hard dataset Amazon-Google. First, Uncertainty sampling might select a significant fraction of doubt cases, mostly uncertain non-matches. In some AL iterations, the sudden drop might be caused by the model do not make correct predictions, and select non-representative outliers that affect the transformer model negatively. We have to recall that transformer model is restarted and retrained on the updated training set after each AL iteration. Hence, such new examples must somehow have a significant impact during training of the transformer model as evaluation on the test set results in a decreasing PC score. Secondly, the bad performance of the query sampling strategy HC-P/N might indicate the transformer model has trained the decision boundary to be too confident, and as a result, it struggles to separate least confident and difficult non-matching record pairs from matching record pairs. Those effects highlight the advantages related to partition sampling a balanced set with the least confident and high confidence examples, in particular after starting with an imbalanced seed.

Unexpected Performance with Pseudo Labeling in Partition-2. Partition-2, Partition-4 and Uncertainty are quite similar query strategies, as described in Chapter 5, where the latter is a semi-supervised method. As Partition-2 could train on twice as many new examples in every iteration, we expected to achieve a significant improvement in PC score for this query strategy compared to Partition-4 and Uncertainty. Although Partition-2 yielded better performance on imbalanced seeds than balanced seeds, it is still an unanswered question why pseudo labeling of examples has a minor impact than expected on the model performance. One can reason that it can be caused by incorrectly labeling high confidence examples. However, those query sampling strategies had more

similar blocking performance than expected.

Faster Convergence and Higher Asymptote with Active Learning. We evaluated if the informative query sampling strategies improves the blocking performance based on our TPLM approach. Table 7.3 shows the best performing results. The results matched our intuition that for the challenging datasets, query sampling strategies helped to boost the PC score, whilst for the easy datasets, it had small effect or no effect at all related to surpass the PC score above Baseline-Max. Table 6.3 showed the labeling cost was dramatically reduced when active learning query strategies was applied compared to Baseline. Thus, we advice that in general, it is beneficial to use active query sampling strategies. From the experimental results, there are only minor variations between the different query sampling strategies. In some cases, TopKDAL showcases a more rapid increase in PC score when query sampling strategies were used compared to Baseline, in particular for the unbalanced starting strategy presented in Section 8.2.3. In this way, we argue that active learning query sampling strategies have a greater impact to improve the blocking performance when the transformer model starts with an imbalanced seed than a balanced seed. From our experimental results, TopKDAL achieves in some cases an even higher target performance than Baseline-Max, i.e. a higher PC score asymptote, in particular for the hard datasets Abt-Buy, Amazon-Google and Walmart-Amazon within a labeling budget of 1000 examples. In practice, a faster convergence in PC score results in less time spent labeling for the human-in-the-loop.

Reduced Successfully the Labeling Effort. As seen in Table 6.3, TopKDAL achieved competitive performance across five real-world datasets by using an order of magnitude less number of labeled examples than the entire training set. Our labeling budget represents 5.81-17.41% of the total size of the datasets. Doing so, less training data to label in blocking step corresponds to a shorter training time for TopKDAL and a shorter total time required for retrieving a candidate set to the matching step. Our approach reached or surpassed the PC score for Baseline-Max on fewer labeled examples with a balanced initial training set and Uncertainty as a query sampling method than with an imbalanced initial training set and Partition-2. On the former, TopKDAL reached Baseline-Max after 200-700 labeled examples for the hard datasets Abt-Buy, Amazon-Google, and Walmart-Amazon. In comparison, this corresponds to a training set size of 1.5-10.2% of the total training set size used by Baseline-Max.

Towards mitigating biases. We attribute Random-P/N strategy started with an imbalanced initial training set as our best query sampling strategy choice towards mitigating biases. Table 7.3 shows the imbalanced-Random-P/N strategy yielded competitive performance at the same level as the other query sampling strategies across the five real-world datasets. There are several reasons to why we believe this strategy might be beneficial. (1) Historically, random sampling has mitigated the likelihood of selecting biased examples in the training set to cause biased learning. However, unwanted biases can be implicitly captured in transformer models from the training data. (2) Random-P/N unlocks new blocking opportunities and simplifies how we can mimic the similarities of records in model architecture. Such a query sampling strategy only needs to identify the threshold separating similar examples from dissimilar examples.

7.2.2 Starting Strategy of the Initial Training Data

As seen from the experimental results in Chapter 6, seven query sampling strategies with a balanced starting strategy yielded a higher initial PC score than the seven experiments based on the imbalanced starting strategy, including random sampling as baseline. These two starting scenarios were inspired by Ein-Dor et al. [2020], which investigated active learning methods for BERT-based binary text classification on real-world scenarios with limited labeling budget and imbalanced datasets.

At 200 labeled examples, we can observe from Table 7.3 that the improved initial PC score might be gained by learning from an increase number of informative positive examples. One can reason that the transformer model has obtained a kick-start learning after a 50/50 distribution of 100 positive and 100 negative informative examples. In particular, the transformer model seems to find the decision boundary separating matching and non-matching record pairs with fewer labeled examples, i.e. less labeling cost, compared to the imbalanced starting strategy. Both starting strategies selected the examples randomly to be independent on uncertainty and diversity sampling, and no transformer models or classic machine learning models were involved in this initial sampling process. In this way, we determined the class distribution targets for the initial seed, essentially to avoid cold start problems and kick-start the active learning process for our TopKDAL at a higher initial PC score. As a result, the query sampling methods with a balanced starting strategy have a greater likelihood to achieve a higher PC score over the AL iterations.

Table 7.3: Comparison of PC score between best performing active learning strategies and Baseline with DistilBERT as TPLM. Overall, after 1000 labeled examples, the recommended combination from the results are a balanced training set as starting strategy combined with Uncertainty as query sampling strategy. The following abbreviation is used in the table: P-2 = Partition-2, Random-P/N = Random Positives and Negatives.

		Initial Starting Strategy	Best Performing Query Sampling Strategies (QSS)			
Initial Training Set	Dataset	Random	Baseline	Random-P/N	Uncertainty	P-2
Imbalanced	Amazon-Google	0.822	0.927	0.952 (+0.025)	0.926 (-0.001)	0.954 (+0.027)
	DBLP-ACM	0.950	0.991	0.996 (+0.005)	**1.00** (+0.009)	**1.00** (+0.009)
	DBLP-Scholar	0.822	0.962	0.989 (+0.027)	0.989 (+0.027)	**0.993** (+0.031)
	Walmart-Amazon	0.832	0.862	0.872 (+0.010)	0.877 (+0.015)	0.865 (+0.003)
	Abt-Buy	0.325	0.824	0.883 (+0.059)	0.867 (+0.043)	0.890 (+0.066)
	Mean	0.750	0.913	0.938	0.932	0.940
Balanced	Amazon-Google	**0.933**	0.967	0.964 (-0.003)	**0.973** (+0.006)	**0.973** (+0.006)
	DBLP-ACM	**0.975**	0.987	0.998(+0.011)	0.999 (+0.012)	0.999 (+0.012)
	DBLP-Scholar	**0.872**	0.971	0.983 (+0.012)	0.992 (+0.021)	0.988 (+0.017)
	Walmart-Amazon	**0.933**	0.883	0.883	**0.895**(+0.012)	0.876 (-0.007)
	Abt-Buy	**0.894**	0.862	0.828 (-0.034)	0.900 (+0.038)	**0.901**(+0.039)
	Mean	**0.894**	0.934	0.931	**0.952**	0.947

A downside to the balanced starting strategy is how an initial balanced training set can be sampled efficiently among all record pairs between two datasets, in many situations dominating the fraction of non-matching record pairs. In large-scale industrial datasets, the challenge of identifying matching record pairs to obtain a 50/50 class distribution in initial training set might be time consuming work. In general, it will be a trade-off between the cost of adding extra overhead to obtain a balanced starting strategy and the improvements gained as a higher initial PC score. As seen from Table 6.2, a transformer model starting with a balanced strategy was successful in the beginning of the

AL iterations. However, as a result of inaccurate predictions, a transformer model with imbalanced starting strategy seems to catch up with a transformer model. In many situations, a more straightforward imbalance starting strategy mindset might be the preferred solution to minimize extra overhead

7.2.3 Comparing True Positive Rate and Starting Strategy

We chose the dataset Abt-Buy to showcase how true positive rate acts for the two starting strategies. In theory, recall that we expected 50/50 sampling of positives and negatives via query sampling strategies over the active learning iterations. Figure 7.1 indicates models predict significant wrong in many cases, in particular for a balanced start point because TPR is not maintained at approximately 0.50 over the AL iterations and the query sampling strategies started with imbalanced start point have not reached approximately $(0.50 * 800 + 0.1073 * 200)/1000 = 0.42\%$ matching examples in the Abt-Buy[3] training set.

As expected, we can observe that Uncertainty strategy actually retrieves the fewest positive examples over the AL iterations for both an imbalanced and balanced seed. This sounds reasonable as the query strategy selects the least confident examples and it is the overweight of non-matching examples in the pool $\mathcal{U}$. According to our P-4-3xPB objective described in Section 5.3, partition sampling with three times as many HCP as HCN examples increase TPR most with imbalanced starting strategy, and opposite, it maintains better TPR for a balanced starting strategy. However, as showed in Table 6.2, this boost in number of positive examples cannot be used as a treatment to outperform Balanced-Uncertainty and Imbalanced-P-2.

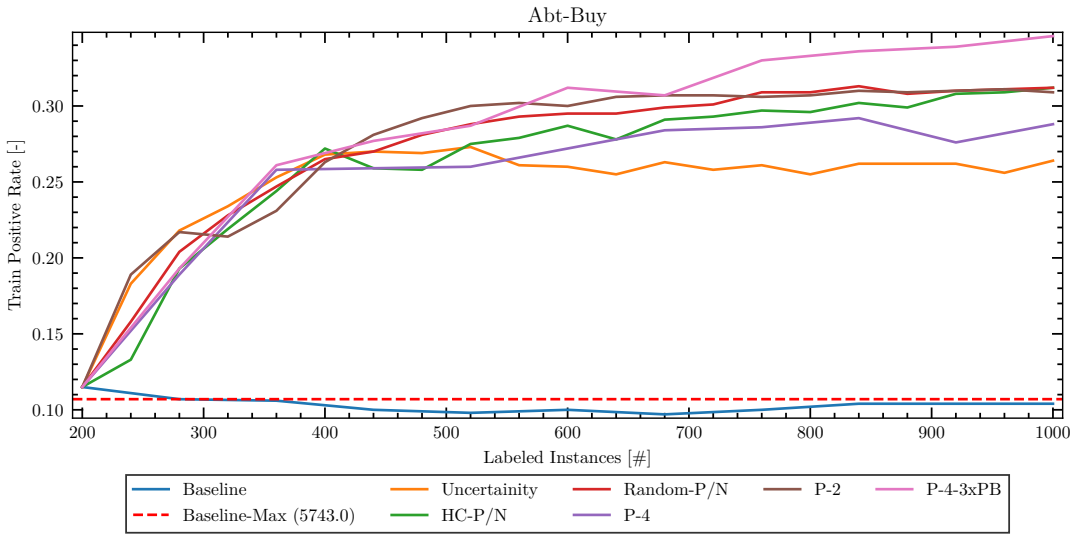
Balanced-Uncertainty demonstrated the best results. One can reason that the model gains the most advantage of training on least confident examples after being kick-started with a balanced seed. Figure 7.1 shows Balanced-Uncertainty achieved the lowest fraction of positive examples in the training set among the query sampling strategies after approximately 600-650 labeled examples while achieving the highest PC score. Even though the model attempts to select 20 positive and 20 negative least confident examples for each iteration during Balanced-Uncertainty, the TPR indicates the model selects even more uncertain non-matching than matching examples. As a result, the transformer model seems to improve blocking performance by better separating difficult positive and negative examples from each other during the evaluation of the test set. Such an observation might indicate that the model should be trained with a balanced starting strategy covering half of the positive space randomly and other half of the negative space randomly as a seed. Potentially, when starting with a balanced starting strategy, a positive side effect of decreasing TPR over the AL iterations might be minor class distribution deviation between training and test set.

In the opposite case, we observed that Imbalanced-P-2 strategy performed best with imbalanced starting strategy. When the model is trained with an imbalanced training set closer to the distribution in the source dataset, the model improves the blocking performance mostly by selecting a balanced set based on partition sampling with pseudo

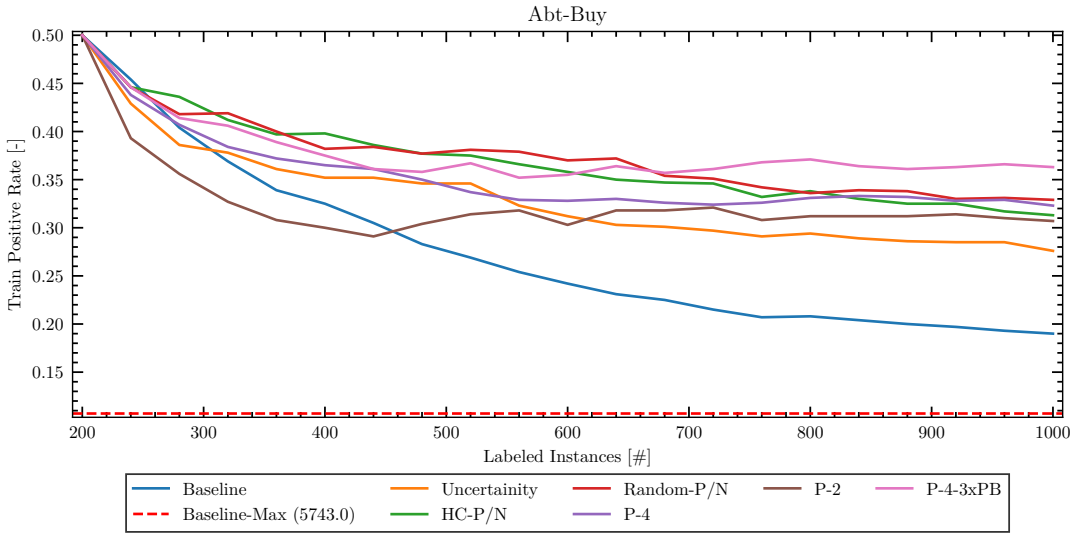
[3]All benchmark datasets consist of more than 500 matching record pairs according to Table 4.2

labeling of high confident examples and labeling of least confident examples performed by the Oracle. One can reason that the model seems to get reward from a twice as large training size because TPR for Partition-4 is higher than for Partition-2. Overall, we see that all query sampling strategies starting with an imbalanced initial set increase their TPR over the AL iterations. In this case, an unwanted side effect is an increasing deviation in the class distribution between the training and test set.

After 1000 labeled examples, it is interesting that TPR of almost every query sampling strategy converges to approximately TPR of 0.26-0.36. This observation addresses the problem with AL and TPLM if the model is not performing at a sufficient level. As the query sampling depends on the model's predictions, it can result in such a situation that the model predicts all examples as non-matching examples. This way, if the query strategy cannot select positive matches to label, it results in a decreased TPR in the training set as observed in Figure 7.1. At a labeling budget below 1000 labeled examples, Balanced-Uncertainty seems to be the preferred strategy for achieving a high TPR while reaching a high PC score. More TPR results for the other datasets with DistilBERT as TPLM can be found in Section 8.2.3.



(a) True positive rate over 1000 labeled examples given an imbalanced initial training set distribution. The number of positive examples in the training set increases over the active learning sampling iterations. The increase in TPR is caused by the query strategy can find a high amount of positive examples in the pool $\mathcal{U}$ based on model's predictions.



(b) True positive rate over 1000 labeled examples given a balanced initial training set. The number of positive examples in the training set decreases over the active learning sampling iterations. The reduction in TPR is caused by the query strategy cannot find enough positive examples in the pool $\mathcal{U}$ based on model's predictions to maintain the initial 50/50 distribution.

Figure 7.1: True positive rate (TPR) for the query sampling strategies on dataset Abt-Buy. The red dashed line denotes Baseline-Max trained on all training data.

7.3 Challenges

The third research question addresses the challenges of combining active learning and Transformer pre-trained models (TPLM) for blocking in entity matching. Some challenges we faced when developing our approach TopKDAL for blocking are also discussed.

7.3.1 Active Learning Needs Interactive Domain Expert

Active learning (AL) is not a suitable strategy to improve the blocking performance and reduce the labeling effort related to TPLM if an interactive domain expert, who can label the record pairs correctly, is not a part of the system. The critical component to unlock the potential of AL in blocking is 1) the involvement of an interactive domain expert and 2) a suitable user interface customized for the domain expert to perform labeling. While the public datasets were pre-labeled in our benchmark datasets, active learning as a strategy has a limited application domain in the real world if nobody can correctly and efficiently label the datasets. Hence, the design of a labeling application for the domain experts should have a straightforward interface. Next, the user interface needs to have the functionality to make the labeling work process more manageable. Finally, the new labeled examples should be passed to update the training set and injected into the model in each active learning iteration.

How to solve the user interactive work process in active learning is not well researched. Due to this complex obstacle about how active learning can be applied in a production setting, it can be argued that many companies would mainly use the TPLM on smaller labeled datasets. However, the research community should work with companies to comprehensively review how active learning can be used in an industrial setting such that our proposed TopKDAL can be applied on large-scale industrial datasets.

7.3.2 Informativeness vs. Representativeness AL Strategies

An important aspect is identifying any fundamental weaknesses of using our active learning strategies to perform blocking in entity matching. Our work focused on the most popular active learning approach to evaluate informativeness-based active learning strategies, which involves querying the most informative instances. We chose informativeness to represent the ability of a sample to reduce the generalization error of the adopted classification model. In this way, we attempted to reduce the uncertainty related to the classification model in the next AL iteration. Our tested approaches include Uncertainty, P-4, P-2, HC-P/N, P-4-3xPos. The main weakness of these approaches is that they cannot exploit the abundance of unlabeled data, and a few labeled examples solely determine the selection of instances, consequently making it prone to sampling biases.

In our work, representativeness is not considered since it usually involves applying a clustering method. It could exploit the underlying unlabeled data's cluster structure to select the most representative examples among the unlabeled data. The main weakness

of representativeness-based approaches is that their performance heavily depends on the quality of clustering results. In addition, it brings more complexity to our interpretations of results. However, the common problem of our informativeness active learning approaches is that the query samples merely rely on scarce labeled data instead of using the full advantage of the information of abundant unlabeled data. Therefore, evaluating TopKDAL with an active learning sampling strategy approach that effectively fuses informativeness and representativeness would be interesting as further work. The purpose would be to ensure that the query samples possess the representativeness of the unlabeled data and reveal the diversity of the labeled data.

7.3.3 Application in Industry

A question to address is how AL with TPLM can be applied to significantly reduce labeling costs while improving PC scores in an industry setting. Our experimental setup attempted to take into consideration real-world scenarios for blocking bounded time and resource limitations. The purpose was to limit the iteration time spent over 20 rounds of active learning. As a result, a shorter iteration time was achieved with a smaller TPLM to gain a shorter waiting time for the Oracle between each time it could start labeling new examples. For a company, the human domain expert involvement as an Oracle has to be used efficiently in a AL loop. It means to maximize the time spent on labeling by a human domain expert while reducing its waiting time because human domain experts are often an expensive asset and a limited resource for the company. Therefore, we aim to have a short time window between each labeling iteration, independent of how trivial the labeling task can be perceived by the human domain expert. The most important is to fill up the human domain experts' available time as they also have many other work tasks on their schedule.

There are several ways to involve the domain expert in that AL loop. Labeling record pairs individually by an Oracle is one approach. A more optimal approach is to query the Oracle regarding regular expressions to find matches in the dataset. A regular expression needs to identify something in common between the record pairs in the candidate set. To exemplify, it might be that all records with the attribute value "*7030 Trondheim*" in the attribute name '*post address*' have to match. In this way, the Oracle can label larger batches of examples per time unit by applying these regular expressions systematically on the unlabeled data. A challenge of using TPLM in the heavy asset industry is getting certification on a deep learning model, which TopKDAL depends on. Without a certification, TopKDAL is worth nothing because the company needs to understand the model's behavior. This understanding is essential when companies do not accept any false positives or false negatives in their systems. In this case, AL can be a beneficial method as it is a human-in-the-loop process where a domain expert can review and validate the label of the record pairs.

7.3.4 Pseudo Labeling and Semi-supervised Methods for Blocking

The second-best performing query strategy was imbalanced Partition-2. Although pseudo labeling of high confident examples and semi-supervised methods can be relevant for extending the number of labeled data to reduce the human interaction in each active learning iteration, it could also introduce incorrectly labeled examples and model performance problems. The reason is twofold. 1) It is essential not to alternate positive examples with negative ones during labeling when the data availability is low and highly imbalanced. 2) An entity matching system might not accept any degree of false labeled positives in its system. Particularly, if there are few positive examples available in total, these false positives could affect the model performance negatively during training.

7.3.5 Selection of Transformer Pre-trained Language Model

As we have seen, one needs to adjust the language model size and fine-tuning training time to reach a acceptable response time for human interaction in active learning. Thus, it is not straightforward to find a suitable TPLM and its corresponding hyperparameters appropriately fine-tuned on labeled data. As experienced with TopKDAL, there are a huge amount of different TPLMs with different pre-trained objectives [Qiu et al., 2020]. However, there seems to be less attention to pre-trained language models for blocking or matching in entity matching, except for Brunner and Stockinger [2020], who published some work to benchmark several TPLMs on matching as a task. As observed in Figure 6.3, the iteration time for dataset Abt-Buy varied between 2-3 minutes and 5-7 minutes for DistilBERT and RoBERTa, respectively. This result was achieved after empirical testing. Such empirical testing of TPLM and hyperparameters to define a comprehensive benchmark of different TPLMs and hyperparameters are needed for blocking. This way, it would be easier to identify which pre-trained language models works with a versatile blocking approach. As experienced in our results, it could also be helpful to include how TPLM is affected by different class distributions in the initial training set.

7.4 Alternative Blocking Approaches

Transfer active learning-based blocking with TPLM is an alternative solution to our active learning with TPLM approach. The idea is based on Kasai et al. [2019] as stated that transfer learning (TL) itself can achieve an additional increase in performance by incorporating active learning for doing matching. This approach can be developed by selecting a TPLM from available language models. It can be used as the starting point before the model is trained on labeled source datasets, which reducing the need for human interaction. Then, the model can be applied to the target dataset. Many research institutions release language models based on large and challenging datasets that may be applied as well. It might be several advantages of using transfer active learning with TPLM compared to our proposed TopKDAL. Some of the advantages could be 1) higher initial PC

score competitive with our two starting strategies, 2) higher and steeper improvement rate in performance than achieved here, 3) the converged performance of the trained model might gain a higher asymptote, and 4) there is a minor need for an Oracle if the source datasets are labeled. However, TL can be negatively affected by specific relationships in the source datasets, particular if model learns relationships in the source datasets that are negatively for the target dataset. To the best of our knowledge, a TAL-based blocking approach with TPLM has not obtained any published research yet.

Chapter 8

Conclusion

8.1 Conclusion

We have presented TopKDAL, our blocking approach combines active learning with Transformer pre-trained language models. TopKDAL is a hands-off blocking approach on tabular records, based on similarity-preserving representation learning and top-k nearest neighbor search. Model architecture is based on Transformers with respect to ignore non-matching record pairs in the datasets in question from the candidate set, and to enable active learning as a strategy to reduce the labeling effort among data-hungry transformer models. Through experiments conducted on five public entity matching datasets, we showed that the blocking performance and query sampling strategies were depended on the initial class distribution in the starting strategy. TopKDAL achieved competitive PC scores with significant reduction in required training set sizes, demonstrating active learning as a valid strategy to consider for TPLM-based blocking approaches.

RQ1 *How can Transformers be used to improve the blocking performance in entity matching??*

We have presented TopKDAL, a versatile blocking approach for different pre-trained transformer language models (TPLM) to be fine-tuned on labeled data from the target dataset. TPLMs derived semantically meaningful sentence embeddings, resulting in the immensity of language understanding and ability to learn where to pay attention between the records. Doing so, TPLMs unveiled similarities between entities. These similarities were used as input in a top-k search to retrieve the most similar record pairs for each record into a candidate set. Such a candidate set simplifies the binary classification performed by a single matcher.

A simple classification layer on top of the powerful TPLM was used for unlocking the potential of active learning (AL) as a strategy. This layer provided model predictions among the record pairs retrieved in the candidate set, and enabled the application of

informative query sampling strategies in an AL loop. In this way, we reduced the required labeling effort by incorporating active learning to select informative examples to fine-tune a TPLM. At the same time, we maintained the model accuracy and the blocking performance where involvement of domain experts were simplified to only label record pairs in the AL loop.

TopKDAL does not suffice to construct a unified entity matching system. Hence, the blocking performance might be improved even more if the blocking step and the matching step are jointly integrated and learned as a system. The performance improvement in one can benefit the other, independent of where and how these examples are selected.

RQ2 *How does active learning with transformers perform for blocking with respect to labeling effort (i.e. the number of labels)?*

In a low-resource setting with a labeling budget of 1000 examples, TopKDAL achieved competitive performance on five real-world datasets by only using an order of magnitude less number of labeled examples than training on an entire training set. Our labeling budget represents 5.81-17.41% of the total size of the datasets. In practice, fewer examples to label in the training set for the blocking model corresponds to a shorter total time required to obtain a candidate set for a matching step in entity matching. How an application of TopKDAL would perform in an industry setting is suggested as further work.

Active learning with TPLM contributed to a more rapid increase in PC score and faster convergence than Baseline. We found that the best-performing query sampling strategy is depending on the initial training set the model was trained on. On an imbalanced initial training set, the partition sampling strategy of pseudo labeling of high confident examples and manually labeling low confident examples yielded on average 0.027 higher PC score than Baseline after 1000 labeled instances. In this semi-supervised method, the model was trained on a balanced set of twice as many examples per AL iteration. Given a balanced initial training set, Uncertainty achieved the best performing query sampling strategy on average 0.018 higher PC score than Baseline. Overall, the balanced initial training set with trained on uncertain examples as the query sampling strategy achieved on average a PC score of 0.947 across the five datasets.

Our approach reached or surpassed the PC score for Baseline-Max with fewer labeled examples with Uncertainty as a query sampling method with a balanced starting strategy than Partition-2 with an imbalanced initial training set. On the former, TopKDAL reached Baseline-Max after 200-700 labeled examples for the hard datasets Abt-Buy, Amazon-Google, and Walmart-Amazon. In comparison, this corresponds to a training set size of 1.5-10.2 % of the total training set size used in Baseline-Max.

Given 200 labeled instances as an initial training set, our approach TopKDAL achieved on average 0.144% higher initial PC score across the datasets with a balanced initial training set than the imbalanced initial training set. This improvement indicates that a balanced starting strategy can be used to kick-start TPLMs, in particular if the labeling

budget is reduced to 200 examples in total. In addition, a balanced distribution seems to be beneficial 1) to reduce the likelihood of a cold start problem, and 2) to avoid the use of transfer learning from source datasets. However, our starting strategies are based on random sampling of positive and negative examples. Hence, an initial training set of 200 examples selected by combining the active learning query sampling strategy with traditional machine learning methods might improve the initial PC score even more.

We found that the initial PC score based on a balanced initial training set initially could not be caught up by TopKDAL with an imbalanced starting strategy. However, the initial difference in PC score of 0.114 on average was reduced to 0.012 after 1000 labeled examples. However, the model stability was significantly more stable for query sampling strategies trained on imbalanced initial training set.

The experiments showed that the iteration time was significantly reduced when a smaller TPLM was combined with a lower number of epochs compared to the larger language model RoBERTa. In active learning, a shorter iteration time is beneficial to gain a shorter waiting time between each labeling sequence.

RQ3 *What are the challenges in combining active learning and Transformer pre-trained models (TPLM) with respect to blocking in an entity matching system?*

TopKDAL addressed several challenges of using AL with TPLM for blocking. Firstly, which search algorithm should be used to retrieve matching record pairs from non-matching record pairs. In our approach, we used top-k search. Secondly, A loss function is interconnected with query sampling strategies. Hence, model architecture for blocking needs to be designed for providing probability predictions of the record pairs for unlocking the potential of active learning. Although active learning as a strategy introduces extra overhead to setup, it reduces the labeling cost significantly in low-resource setting. Lastly, record pairs categorized as doubt cases are difficult to mimic in the embedding space with Softmax. An option to consider is to combine Triplet Loss and Random-P/N based on threshold as a query sampling strategy. In this way, we can apply the TPLM's embeddings more directly when selecting examples to label. We believe further work in this direction could lead to improvements for AL with TPLMs for blocking.

Currently, no specific TPLM is pointed out for blocking in entity matching. The availability of different TPLMs is high. Unfortunately, it is published few previous works explaining which TPLMs and hyperparameters should be used. Our work has showed some aspects of how hyperparameters, TPLM, initial training set distribution, and query sampling strategies are interconnected with each other. When working with blocking on larger datasets, such as industrial datasets, an optimization might be even more critical for reducing the iteration time. The consequence of having a sub-optimal choice of parameter combinations might result in instability and unusable models.

8.1.1 Industrial Application

TopKDAL unlocks several opportunities for industry companies of any size to investigate relationships and similarities among record pairs across several data sources to gain value outtakes from the company's data by retrieving a candidate set consisting of likely matches. TopKDAL is a proposed approach to companies' call for improving blocking performance in a low-resource setting, and to handle datasets consisting of constantly increasing volumes of diverse, heterogeneous, inconsistent, and noisy information. Today, a rise in heterogeneity data has resulted in even more unstructured, unclean, incomplete data and diverse data types. Towards hands-off blocking TopKDAL demonstrates a versatile blocking approach working across different datasets and domains as frictionless as possible, in which the domain experts can apply their efforts to label the most informative data. TopKDAL, as a proposed solution, is a key component for companies to unlock entity matching as a task on large and very large datasets, such as industrial datasets.

However, there are still several practical challenges the companies are facing before TopKDAL can be used. 1) Establish a solution related to involving an interactive domain expert in an active learning loop. 2) There are no capability to explain the reasons to matching or non-matching outcomes between two records. Hence, TopKDAL can be perceived as a black-box when the companies do not understand the processes under the hood. 3) TopKDAL does not guarantee algorithmic fairness caused by unwanted biases can be implicitly captured in deep learning models from the training data. In this case, TopKDAL needs to be tested to not be subject to anti-discriminatory laws in many countries.

8.2 Further Work

There are several interesting avenues of research arising from the results of this thesis, and some of those are described as recommendations for further work.

8.2.1 Extension of Our Work

Evaluate how model parameters impact each other and blocking performance accuracy. As experienced during the blocking experiments, it is interconnected relationships between TPLMs, class distribution in training sets, hyperparameters and query sampling strategies, and top-k nearest neighbor search. Comparing the relationships between these factors could also be of great interest, as this could shed some light on how to target an optimal blocking model architecture with active learning query sampling strategies. Related to TopKDAL, it could be relevant to obtain better insight into trade-offs between iteration time, model performance accuracy, and tuning of k to apply for top-k search[1]. For instance, a too high k could result in a sub-optimal top-k search retrieving unnecessarily more non-matching examples in the candidate set C, impacting the size of C, PC score, and RR for TopKDAL negatively. A comprehensive benchmark, as a result of extensive testing such as Meduri et al. [2020], of TPLMs and AL query sampling strategies on benchmark datasets for blocking could 1) draw more robust conclusions on the TopKDAL performance, and 2) contribute to concrete guidelines as to what active learning combinations will work well for blocking in EM.

Combine Data Augmentation and Active Learning to reduce the Labeling Effort. Inspired by Li et al. [2020], it could be interesting to combine data augmentation for text with active learning to improve the TopKDAL blocking performance. Although data augmentation shares the common goal with active learning of improving label efficiency in TopKDAL, it is two different approaches. Active learning requires human interaction in each AL iteration, whereas data augmentation has not that requirement. For instance, data augmentation could expand the number of difficult examples in the training set to force TopKDAL to learn on difficult examples. Additionally, transfer learning could also be combined with the suggested approach. We believe further work in this direction, combining these three approaches, could results in state-of-the-art results.

8.2.2 Practical Application in an Industry Setting

In answer to RQ3 of this thesis, we have discussed the challenges faced in combining AL and TPLM for blocking. A natural next step is to review how TopKDAL can setup active learning with a human-in-the-loop in an industry setting, and then combine TopKDAL with a matching step to be applied as a unified EM system. Another important aspect to consider is how to take advantage of a rise in novel query sampling strategies more closely coupled with the properties of TPLMs. As presented in Chapter 5, we used informative

[1]Our experimental results used fixed $k = 10$ in the top-k approximate nearest neighbor search.

query sampling strategies depending on model predictions to select the examples to label. In turn, such novel query sampling strategies could potentially retrieve a better training set over AL iterations than our used model agnostic query sampling strategies. Today, it is difficult to directly evaluate the various applications of traditional blocking methods, non-active learning blocking methods, and AL blocking methods. In an industry setting, we believe a standardized test framework would simplify their burden of comparing applications of non-AL and AL-based blocking approaches, a challenge still remains today. A test framework would need to incorporate AL strategies and evaluation metrics such as iteration time and labeling time. Although TopKDAL is a promising approach for blocking, there are a series of hurdles that need to be overcome before a unified framework for EM system can be deployed. In this case, a comprehensive benchmark of blocking methods for benchmark datasets would be beneficial to reproduce the approaches published up-to-now and identify improvements among these blocking methods.

Towards leveraging TPLM-based approaches and investigating how seamlessly TopKDAL can be applied across other domains and datasets compared to traditional blocking methods, it would also be interesting to evaluate our approach on multilingual datasets such as Jain et al. [2021]. Towards industry setting settings, we also suggest to evaluate training set with labeling noise against training set with no-noise. Most of the published researched work assumes that the training data is perfectly labeled. However, TopKDAL needs to show competitive results with fixed fractions of incorrectly labeled instances in the training set to be defined as a robust approach. This scenario is coming from the increasing popularity of crowd-sourcing, which often can lead to incorrectly labeled matches and non-matches in the labeled training set.

8.2.3 New Approaches for Blocking

Utilizing Transfer Learning for Blocking. The application of domain-specific TPLM with AL should be evaluated to enhance the blocking performance while reducing the labeling effort and iteration time usage. Every TPLM is of interest if they can reduce the active learning iteration time, which means the time used to train the model and predict the record pairs. As observed in our experimental results, a pre-trained smaller general-purpose language model, such as DistilBERT, decreases the iteration time significantly compared to a more prominent language representation model like RoBERTa. We expect that pre-training TopKDAL on domain-specific or EM-specific source datasets upfront might improve the blocking performance on the target datasets by increasing PC score and potentially decreasing the training time. Another open question is whether it can be beneficial to build language models meant for blocking and matching as tasks.

Combine Triplet Loss and a modified Random-P/N. Our experiments' query sampling strategies are mainly based on informative query sampling strategies based on model predictions to select examples to label. Suppose a solution to find where the threshold separating matches from non-matches could be achieved efficiently. In that case, we suggest using cosine similarities between the record pairs in the embedding space combined with sampling balanced positives and negatives randomly as a query strategy. Hopefully, sampling a balanced class distribution over the active learning iteration would lead to

1) enhanced model performance accuracy and reaching a higher PC score, and/or 2) decreased labeling effort. In this way, we can apply the TPLM's embeddings more directly when selecting examples to label [Jain et al., 2021]. We believe further work in this direction could lead to improvements for AL with TPLMs for blocking.

Combine Triplet Loss and a modified Random-P/N. In this way, we can apply the TPLM's embeddings more directly when selecting examples to label [Jain et al., 2021]. We believe further work in this direction could lead to improvements for AL with TPLMs for blocking

Bibliography

A., A., Nordin, A., Alzeber, M., and Zaid, A. (2017). A Survey of Schema Matching Research using Database Schemas and Instances. *International Journal of Advanced Computer Science and Applications*, 8(10).

Abuzaid, F., Sethi, G., Bailis, P., and Zaharia, M. (2019). To Index or Not to Index: Optimizing Exact Maximum Inner Product Search. *arXiv:1706.01449 [cs]*. arXiv: 1706.01449.

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. (2020). Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. *arXiv:1906.03671 [cs, stat]*. arXiv: 1906.03671.

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*. arXiv: 1409.0473.

Barlaug, N. and Gulla, J. A. (2020). Neural Networks for Entity Matching. *arXiv:2010.11075 [cs]*. arXiv: 2010.11075.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *arXiv:1607.04606 [cs]*. arXiv: 1607.04606.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*. arXiv: 2005.14165.

Brunner, U. and Stockinger, K. (2020). Entity Matching with Transformer Architectures - A Step Forward in Data Integration. Version Number: 1 type: dataset.

Cai, W., Zhang, Y., Zhang, Y., Zhou, S., Wang, W., Chen, Z., and Ding, C. (2017). Active Learning for Classification with Maximum Model Change. *ACM Transactions on Information Systems*, 36(2):15:1–15:28.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]*. arXiv: 1406.1078.

Christen, P. (2012a). *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer-Verlag, Berlin Heidelberg.

Christen, P. (2012b). A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *IEEE Transactions on Knowledge and Data Engineering*, 24(9):1537–1555.

Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., and Stefanidis, K. (2020). End-to-End Entity Resolution for Big Data: A Survey. *arXiv:1905.06397 [cs]*. arXiv: 1905.06397.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *arXiv:1901.02860 [cs, stat]*. arXiv: 1901.02860.

Deng, Y., Chen, K., Shen, Y., and Jin, H. (2018). Adversarial Active Learning for Sequences Labeling and Generation. pages 4012–4018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. arXiv: 1810.04805.

Dunn, H. L. (1946). Record Linkage. *American Journal of Public Health and the Nation's Health*, 36(12):1412–1416.

Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., and Tang, N. (2019). DeepER – Deep Entity Resolution. *arXiv:1710.00597 [cs]*. arXiv: 1710.00597.

Efthymiou, V., Stefanidis, K., and Christophides, V. (2020). Benchmarking Blocking Algorithms for Web Entities. *IEEE Transactions on Big Data*, 6(2):382–395.

Ein-Dor, L., Halfon, A., Gera, A., Shnarch, E., Dankin, L., Choshen, L., Danilevsky, M., Aharonov, R., Katz, Y., and Slonim, N. (2020). Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online. Association for Computational Linguistics.

Ertekin, S., Huang, J., Bottou, L., and Giles, L. (2007). Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 127–136, Lisbon, Portugal. Association for Computing Machinery.

Fisher, J., Christen, P., Wang, Q., and Rahm, E. (2015). A Clustering-Based Framework to Control Block Sizes for Entity Resolution. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 279–288, Sydney, NSW, Australia. Association for Computing Machinery.

Hernández, A. and Amigó, J. M. (2021). Attention Mechanisms and Their Applications to Complex Systems. *Entropy*, 23(3):283.

Hernández, M. A. and Stolfo, S. J. (1995). The merge/purge problem for large databases. *ACM SIGMOD Record*, 24(2):127–138.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hsu, W.-N. and Lin, H.-T. (2015). Active learning by learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2659–2665, Austin, Texas. AAAI Press.

Jain, A., Sarawagi, S., and Sen, P. (2021). Deep Indexed Active Learning for Matching Heterogeneous Entity Representations. *arXiv:2104.03986 [cs, stat]*. arXiv: 2104.03986.

Johnson, J., Douze, M., and Jegou, H. (2021). Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Kasai, J., Qian, K., Gurajada, S., Li, Y., and Popa, L. (2019). Low-resource Deep Entity Resolution with Transfer and Active Learning. *arXiv:1906.08042 [cs]*. arXiv: 1906.08042.

Konda, P., Naughton, J., Prasad, S., Krishnan, G., Deep, R., Raghavendra, V., Das, S., Suganthan G. C., P., Doan, A., Ardalan, A., Ballard, J. R., Li, H., Panahi, F., and Zhang, H. (2016). Magellan: toward building entity matching management systems. *Proceedings of the VLDB Endowment*, 9(12):1197–1208.

Konyushkova, K., Sznitman, R., and Fua, P. (2017). Learning Active Learning from Data. *arXiv:1703.03365 [cs]*. arXiv: 1703.03365.

Köpcke, H. and Rahm, E. (2010). Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197–210.

Lewis, D. D. (1995). A sequential algorithm for training text classifiers: corrigendum and additional data. *ACM SIGIR Forum*, 29(2):13–19.

Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. (2020). Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14(1):50–60.

Liang, H., Wang, Y., Christen, P., and Gayler, R. (2014). Noise-Tolerant Approximate Blocking for Dynamic Real-Time Entity Resolution. In Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Kobsa, A., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Terzopoulos, D., Tygar, D., Weikum, G., Tseng, V. S., Ho, T. B., Zhou, Z.-H., Chen, A. L. P., and Kao, H.-Y., editors, *Advances in Knowledge Discovery and Data Mining*, volume 8444, pages 449–460. Springer International Publishing, Cham.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*. arXiv: 1907.11692.

Loshchilov, I. and Hutter, F. (2019). Decoupled Weight Decay Regularization. *arXiv:1711.05101 [cs, math]*. arXiv: 1711.05101.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2019). Explainable AI for Trees: From Local Explanations to Global Understanding. *arXiv:1905.04610 [cs, stat]*. arXiv: 1905.04610.

Maystre, L. and Grossglauser, M. (2017). Just Sort It! A Simple and Effective Approach to Active Preference Learning. *arXiv:1502.05556 [cs, stat]*. arXiv: 1502.05556.

McCallum, A., Nigam, K., and Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 169–178, Boston, Massachusetts, USA. Association for Computing Machinery.

Meduri, V. V., Popa, L., Sen, P., and Sarwat, M. (2020). A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, pages 1133–1147, Portland, OR, USA. Association for Computing Machinery.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*. arXiv: 1301.3781.

Monroe, D. (2018). AI, explain yourself. *Communications of the ACM*, 61(11):11–13.

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., and Raghavendra, V. (2018). Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34, Houston TX USA. ACM.

Papadakis, G., Alexiou, G., Papastefanatos, G., and Koutrika, G. (2015). Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data. *Proceedings of the VLDB Endowment*, 9(4):312–323.

Papadakis, G., Koutrika, G., Palpanas, T., and Nejdl, W. (2014a). Meta-Blocking: Taking Entity Resolutionto the Next Level. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1946–1960.

Papadakis, G., Papastefanatos, G., and Koutrika, G. (2014b). Supervised meta-blocking. *Proceedings of the VLDB Endowment*, 7(14):1929–1940.

Papadakis, G., Papastefanatos, G., Palpanas, T., and Koubarakis, M. (2016a). Boosting the Efficiency of Large-Scale Entity Resolution with Enhanced Meta-Blocking. *Big Data Research*, 6:43–63.

Papadakis, G., Skoutas, D., Thanos, E., and Palpanas, T. (2019). Blocking and Filtering Techniques for Entity Resolution: A Survey.

Papadakis, G., Svirsky, J., Gal, A., and Palpanas, T. (2016b). Comparative analysis of approximate blocking techniques for entity resolution. *Proceedings of the VLDB Endowment*, 9(9):684–695.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv:1912.01703 [cs, stat]*. arXiv: 1912.01703.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Qian, K., Popa, L., and Sen, P. (2017). Active Learning for Large-Scale Entity Resolution. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1379–1388, Singapore Singapore. ACM.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350.

Reimers, N. and Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv:1908.10084 [cs]*. arXiv: 1908.10084.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., and Wang, X. (2020). A Survey of Deep Active Learning. *arXiv:2009.00236 [cs, stat]*. arXiv: 2009.00236.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2020). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]*. arXiv: 1910.01108.

Settles, B. (2009). Active Learning Literature Survey. Technical Report, University of Wisconsin-Madison Department of Computer Sciences.

Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 287–294, Pittsburgh, Pennsylvania, USA. Association for Computing Machinery.

Shao, J., Wang, Q., and Lin, Y. (2018). Skyblocking for Entity Resolution. *arXiv:1805.12319 [cs]*. arXiv: 1805.12319.

Shao, J., Wang, Q., and Liu, F. (2019). Learning to Sample: an Active Learning Framework. *arXiv:1909.03585 [cs, stat]*. arXiv: 1909.03585.

Shui, C., Zhou, F., Gagné, C., and Wang, B. (2020). Deep Active Learning: Unified and Principled Method for Query and Training. *arXiv:1911.09162 [cs, stat]*. arXiv: 1911.09162.

Simonini, G., Bergamaschi, S., and Jagadish, H. V. (2016). BLAST: a loosely schema-aware meta-blocking approach for entity resolution. *Proceedings of the VLDB Endowment*, 9(12):1173–1184.

Själander, M., Jahre, M., Tufte, G., and Reissmann, N. (2020). EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. *arXiv:1912.05848 [cs]*. arXiv: 1912.05848.

Stoyanovich, J., Howe, B., and Jagadish, H. V. (2020). Responsible data management. *Proceedings of the VLDB Endowment*, 13(12):3474–3488.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [cs]*. arXiv: 1409.3215.

Thirumuruganathan, S., Ouzzani, M., and Tang, N. (2019). Explaining Entity Resolution Predictions: Where are we and What needs to be done? In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, HILDA'19, pages 1–6, Amsterdam, Netherlands. Association for Computing Machinery.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *arXiv:1706.03762 [cs]*. arXiv: 1706.03762.

Wang, K., Zhang, D., Li, Y., Zhang, R., and Lin, L. (2017). Cost-Effective Active Learning for Deep Image Classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600.

Wang, Q., Cui, M., and Liang, H. (2016). Semantic-aware blocking for entity resolution. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1468–1469, Helsinki, Finland. IEEE.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771 [cs]*. arXiv: 1910.03771.

Xu, Z., Akella, R., and Zhang, Y. (2007). Incorporating Diversity and Density in Active Learning for Relevance Feedback. In Amati, G., Carpineto, C., and Romano, G., editors, *Advances in Information Retrieval*, volume 4425, pages 246–257. Springer Berlin Heidelberg, Berlin, Heidelberg.

Yang, Y., Ma, Z., Nie, F., Chang, X., and Hauptmann, A. G. (2015). Multi-Class Active Learning by Uncertainty Sampling with Diversity Maximization. *International Journal of Computer Vision*, 113(2):113–127.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2020). XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv:1906.08237 [cs]*. arXiv: 1906.08237.

Yang, Z. and Kitsuregawa, M. (2011). Efficient searching top-k semantic similar words. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2373–2378, Barcelona, Catalonia, Spain. AAAI Press.

Yin, C., Qian, B., Cao, S., Li, X., Wei, J., Zheng, Q., and Davidson, I. (2017). Deep Similarity-Based Batch Mode Active Learning with Exploration-Exploitation. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 575–584, New Orleans, LA. IEEE.

Zhang, W., Wei, H., Sisman, B., Dong, X. L., Faloutsos, C., and Page, D. (2020). AutoBlock: A Hands-off Blocking Framework for Entity Matching. *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 744–752. arXiv: 1912.03417.

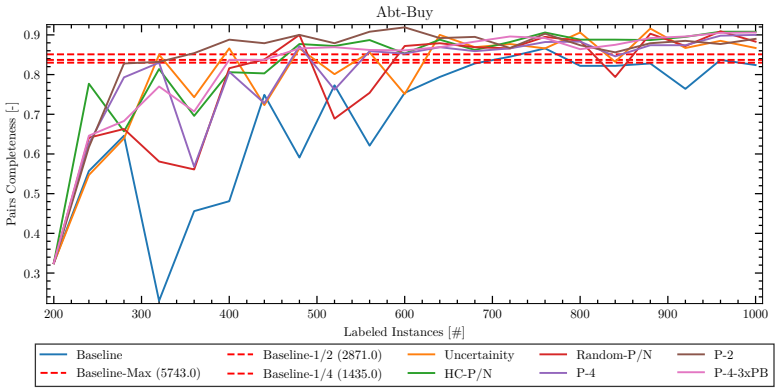
Zhao, C. and He, Y. (2019). Auto-EM: End-to-end Fuzzy Entity-Matching using Pre-trained Deep Models and Transfer Learning. In *The World Wide Web Conference on - WWW '19*, pages 2413–2424, San Francisco, CA, USA. ACM Press.

Zhdanov, F. (2019). Diverse mini-batch Active Learning. *arXiv:1901.05954 [cs, stat]*. arXiv: 1901.05954.

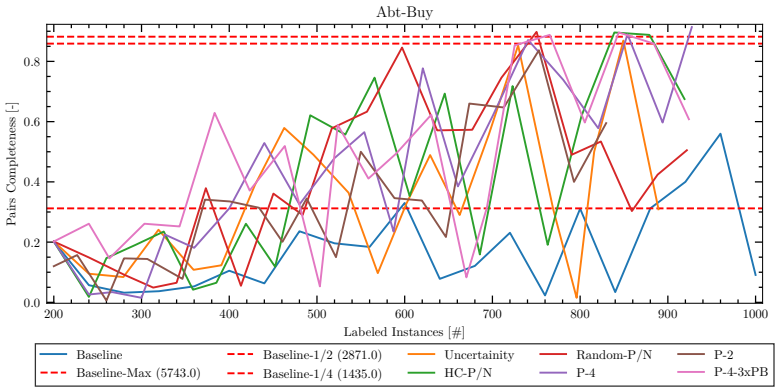
Zhu, H.-J., Zhu, Z.-W., Jiang, T.-H., Cheng, L., Shi, W.-L., Zhou, X., Zhao, F., and Ma, B. (2018). A Type-Based Blocking Technique for Efficient Entity Resolution over Large-Scale Data. *Journal of Sensors*, 2018:e2094696.

Appendices

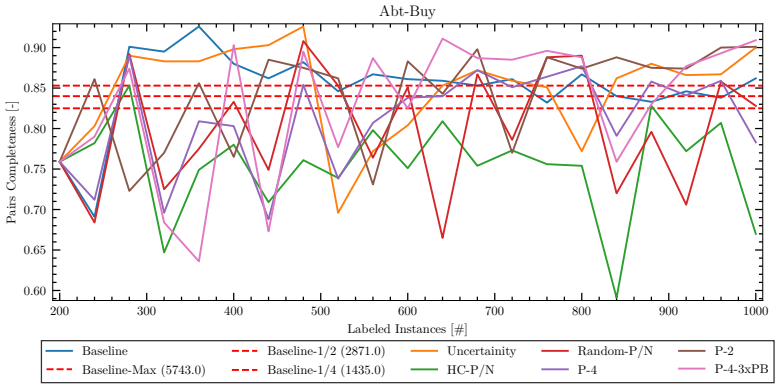
Appendix A Pairs Completeness Results



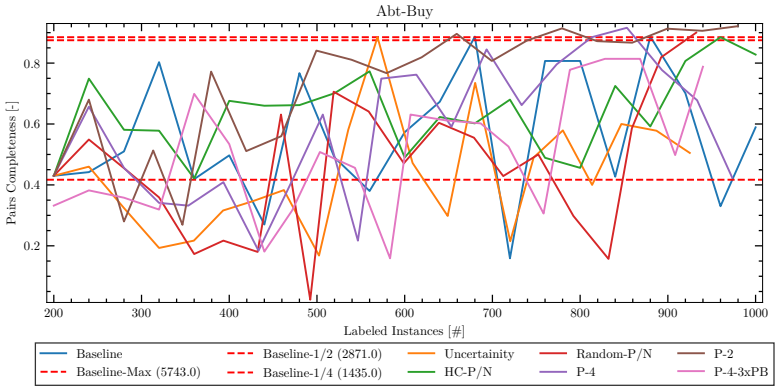
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

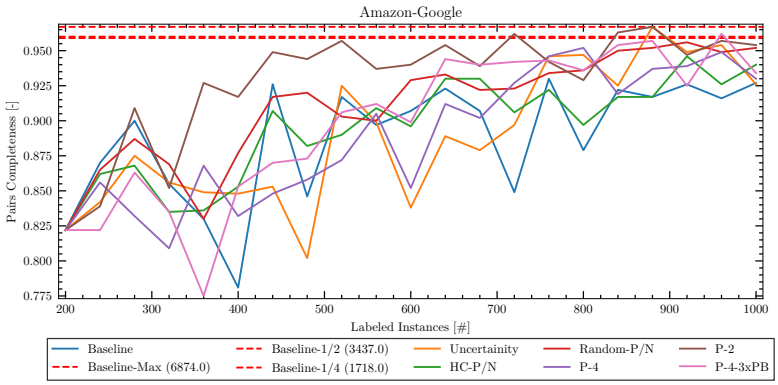


(c) 50/50 initial training set with LM DistilBERT.

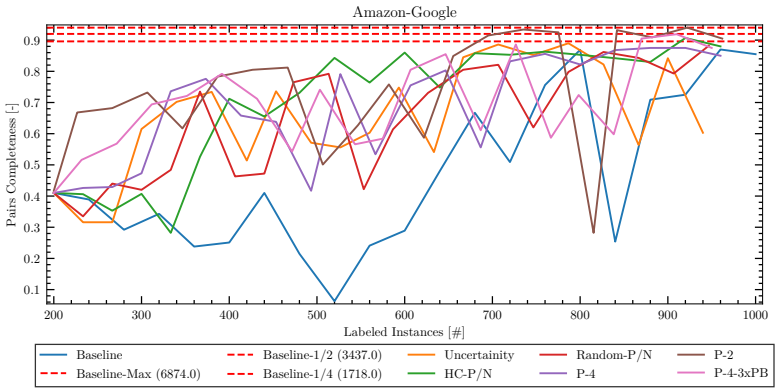


(d) 50/50 initial training set with LM RoBERTa.

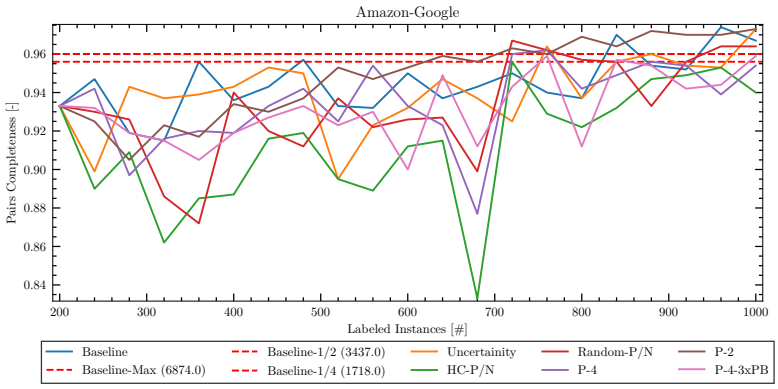
Figure A.1: Experimental results of pair completeness with respect to labeled instances for dataset Abt-Buy.



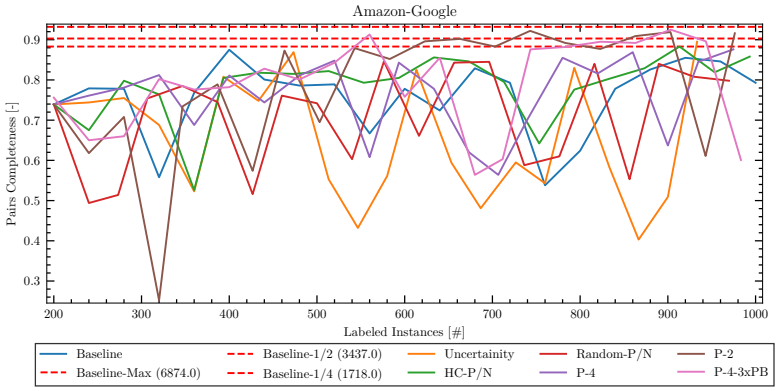
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

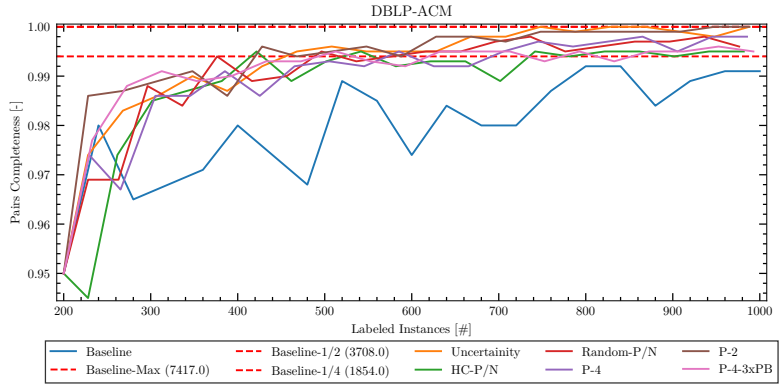


(c) 50/50 initial training set with LM DistilBERT.

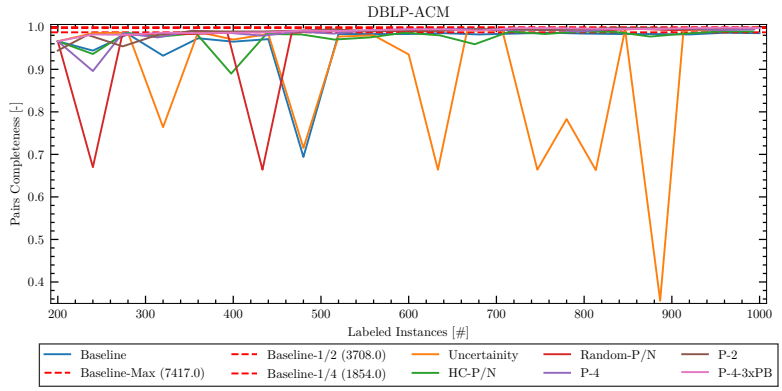


(d) 50/50 initial training set with LM RoBERTa.

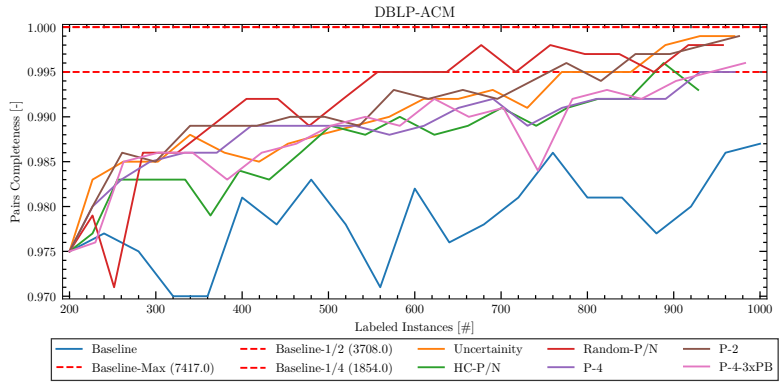
Figure A.2: Experimental results of pair completeness with respect to labeled instances for dataset Amazon-Google.



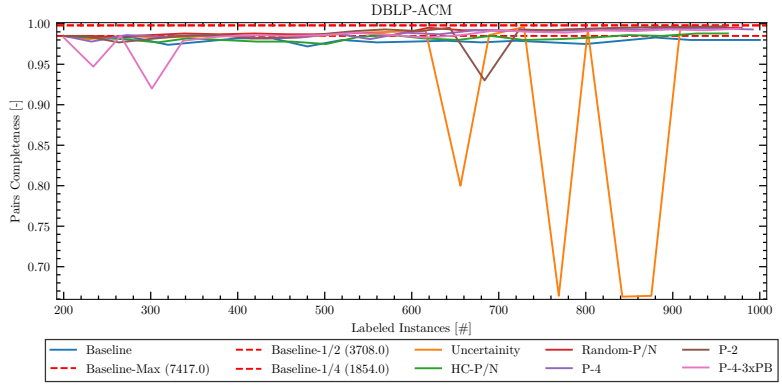
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

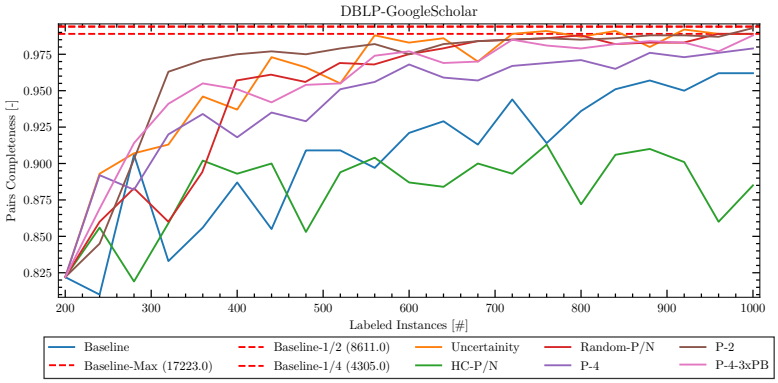


(c) 50/50 initial training set with LM DistilBERT.

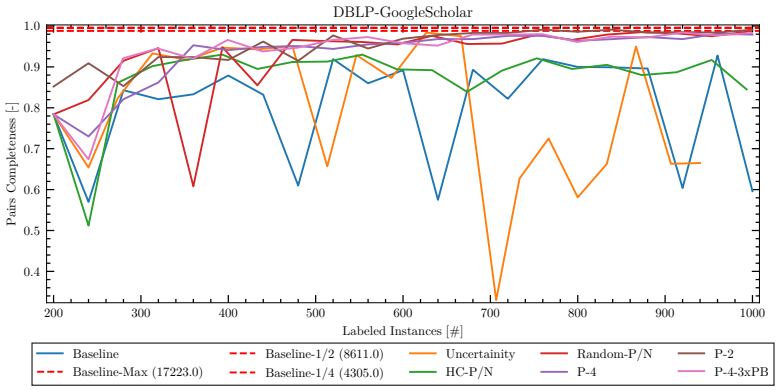


(d) 50/50 initial training set with LM RoBERTa.

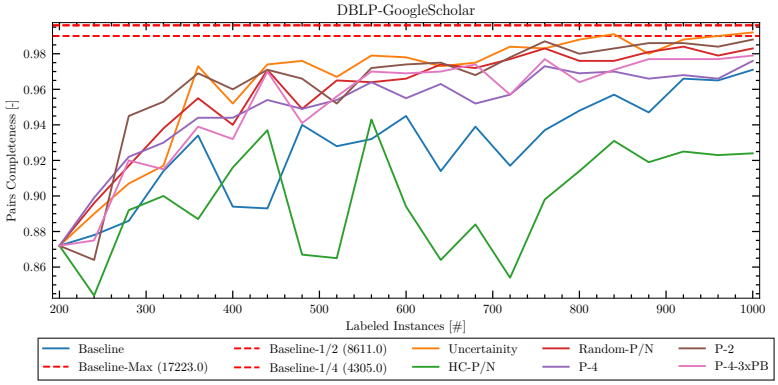
Figure A.3: Experimental results of pair completeness with respect to labeled instances for dataset DBLP-ACM.



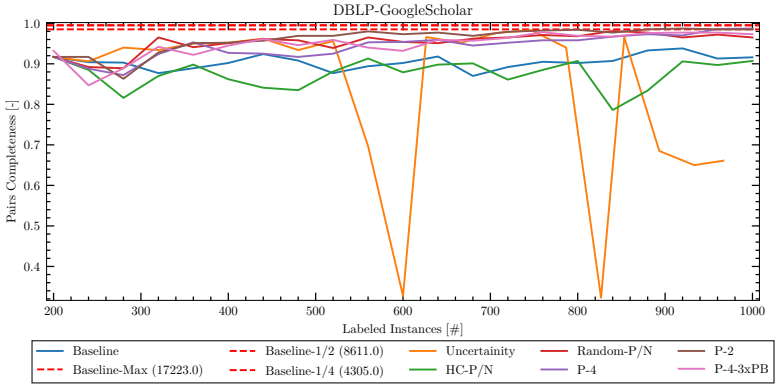
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

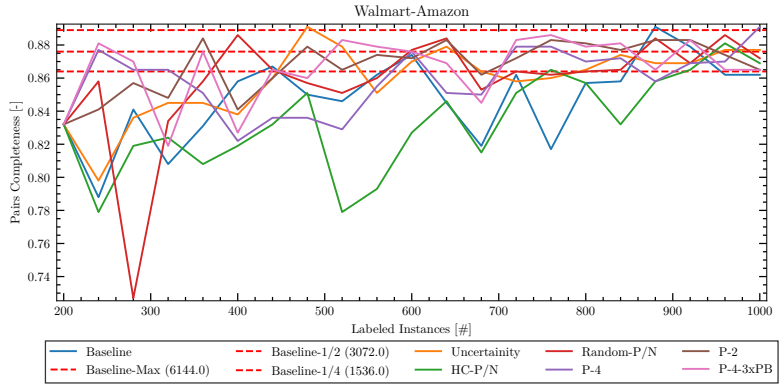


(c) 50/50 initial training set with LM DistilBERT.

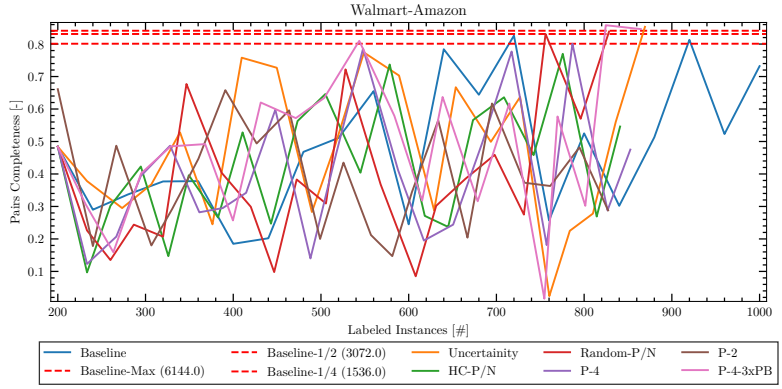


(d) 50/50 initial training set with LM RoBERTa.

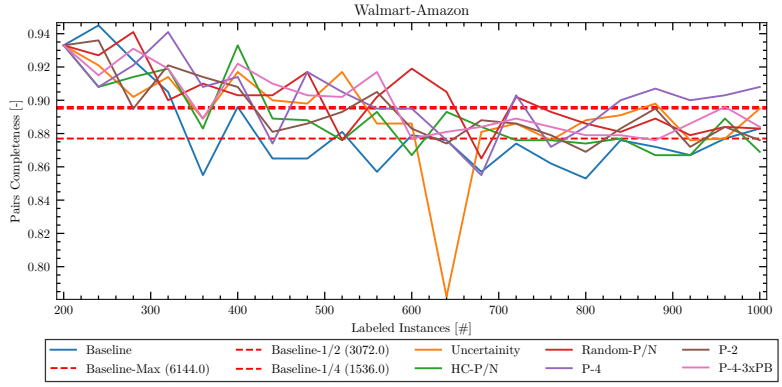
Figure A.4: Experimental results of pair completeness with respect to labeled instances for dataset DBLP-GoogleScholar.



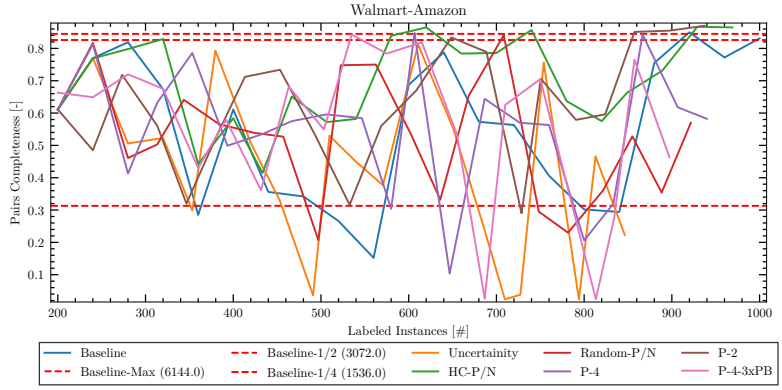
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.



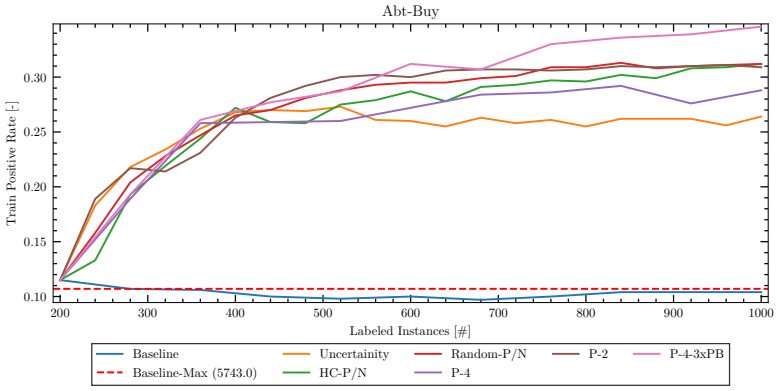
(c) 50/50 initial training set with LM DistilBERT.



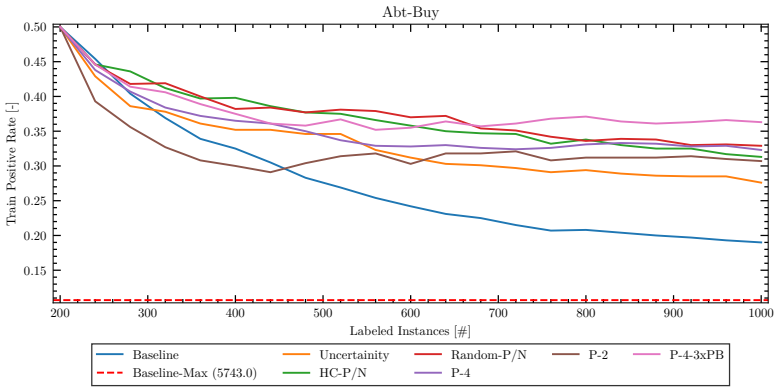
(d) 50/50 initial training set with LM RoBERTa.

Figure A.5: Experimental results of pairs completeness with respect to labeled instances for dataset Walmart-Amazon.

Appendix B True Positive Rate Results

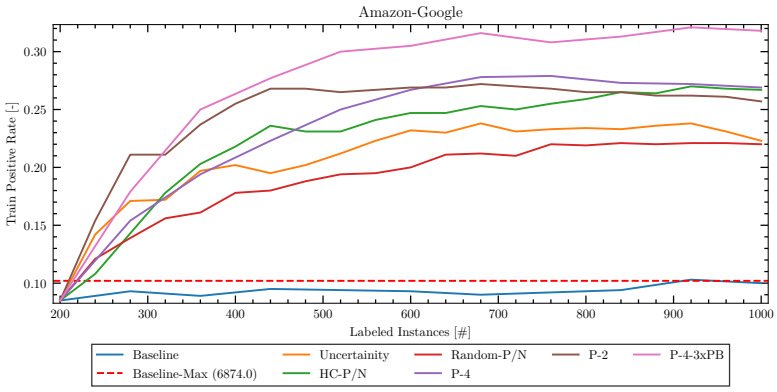


(a) Random initial training set with LM DistilBERT.

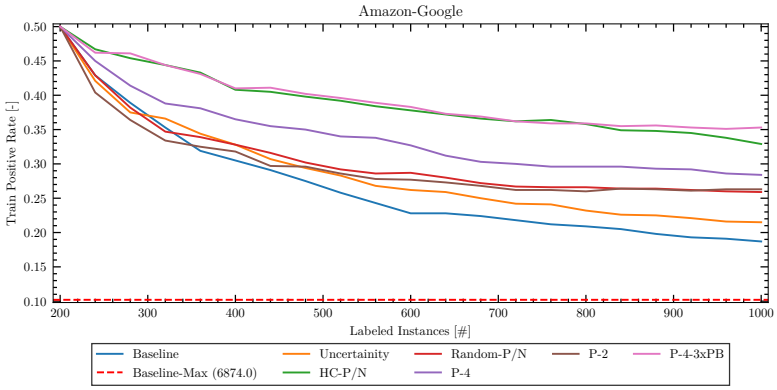


(b) 50/50 initial training set with LM DistilBERT.

Figure B.1: Experimental results of train positive rate with respect to labeled instances for dataset Abt-Buy.

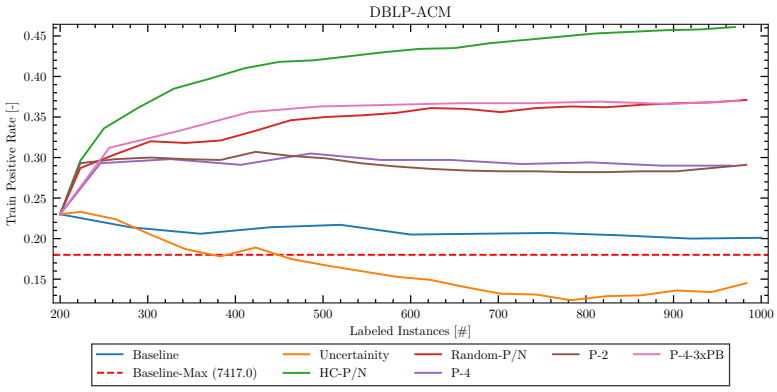


(a) Random initial training set with LM DistilBERT.

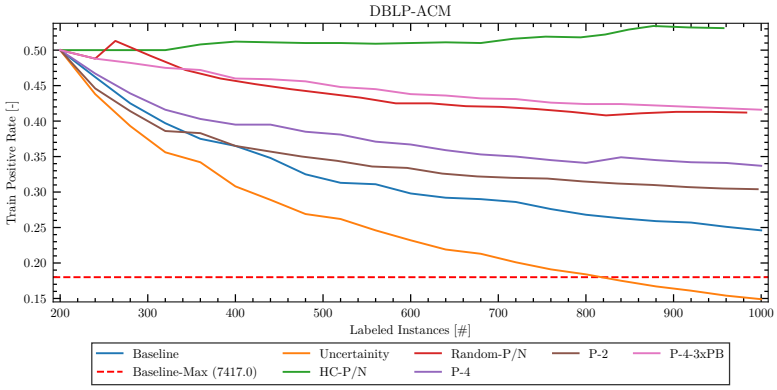


(b) 50/50 initial training set with LM DistilBERT.

Figure B.2: Experimental results of train positive rate with respect to labeled instances for dataset Amazon-Google.

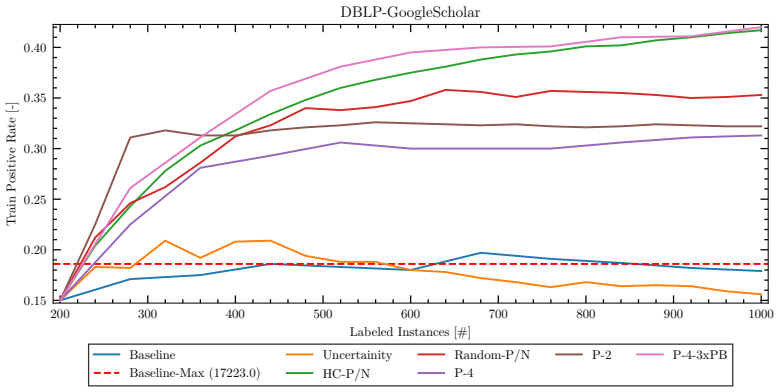


(a) Random initial training set with LM DistilBERT.

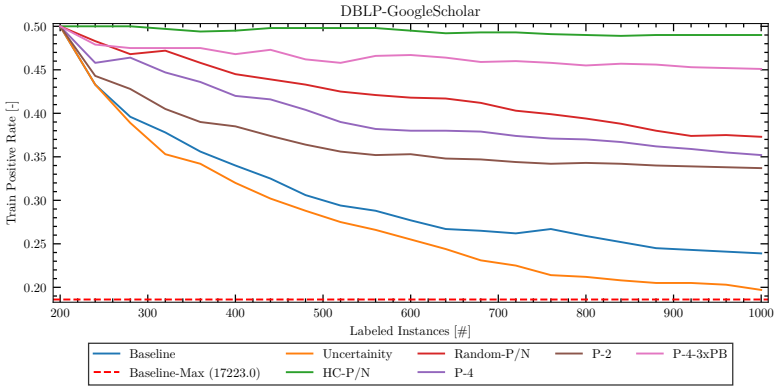


(b) 50/50 initial training set with LM DistilBERT.

Figure B.3: Experimental results of train positive rate with respect to labeled instances for dataset DBLP-ACM.

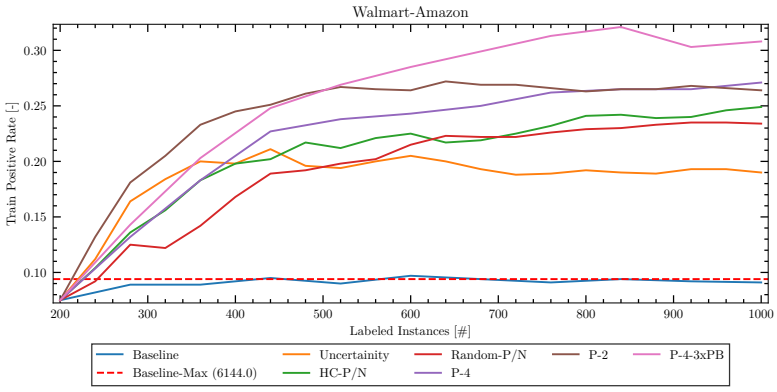


(a) Random initial training set with LM DistilBERT.

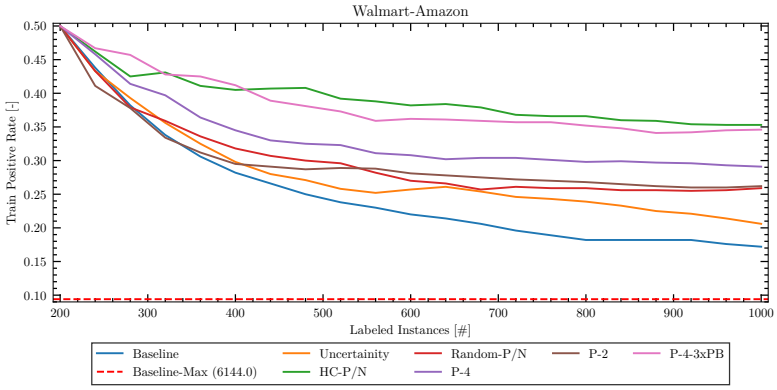


(b) 50/50 initial training set with LM DistilBERT.

Figure B.4: Experimental results of train positive rate with respect to labeled instances for dataset DBLP-GoogleScholar.



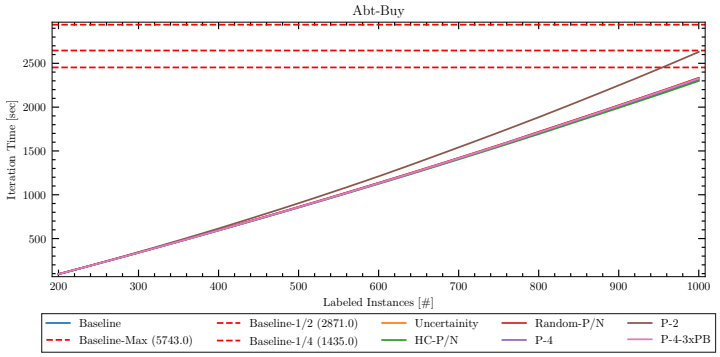
(a) Random initial training set with LM DistilBERT.



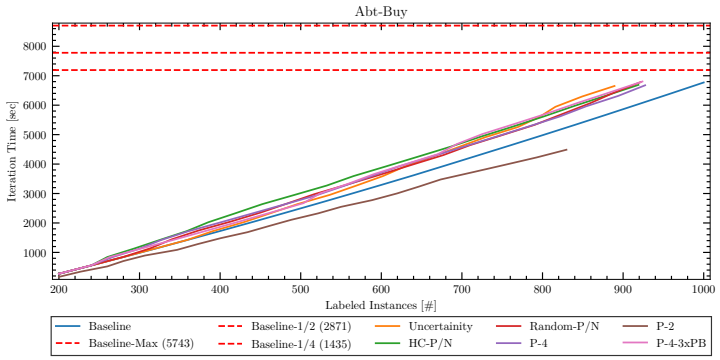
(b) 50/50 initial training set with LM DistilBERT.

Figure B.5: Experimental results of train positive rate with respect to labeled instances for dataset Walmart-Amazon.

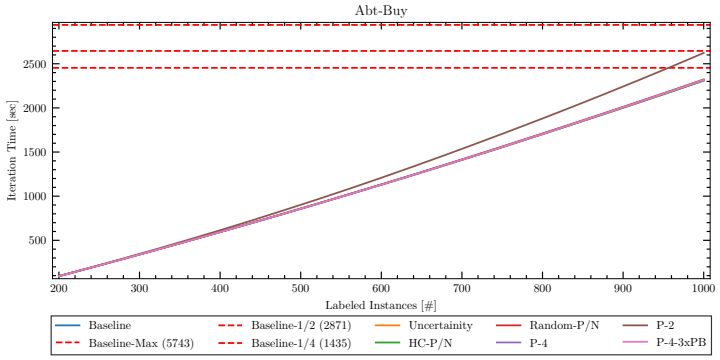
Appendix C Iteration Time Results



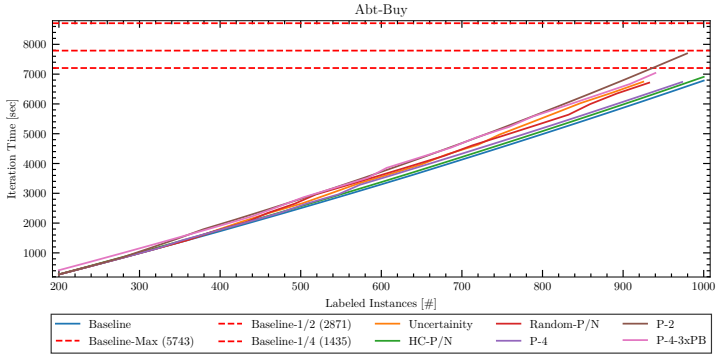
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

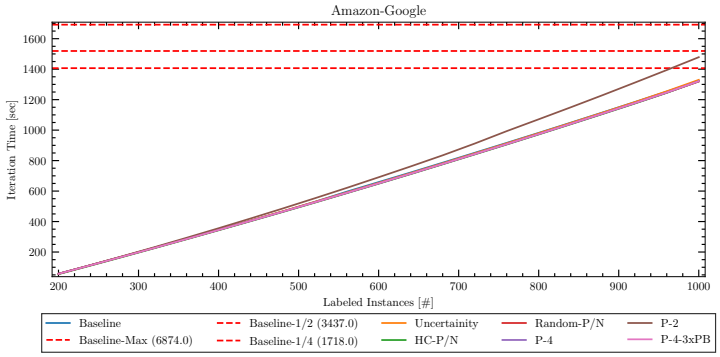


(c) 50/50 initial training set with LM DistilBERT.

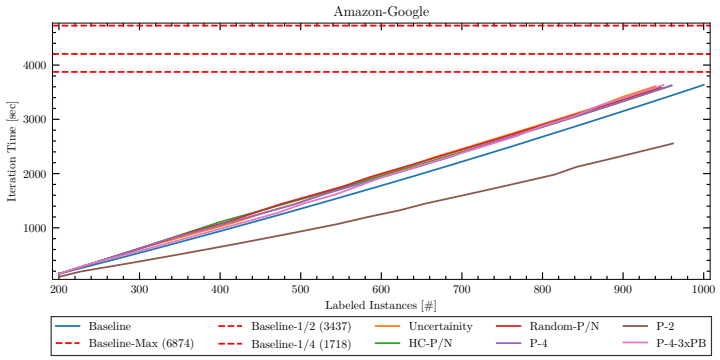


(d) 50/50 initial training set with LM RoBERTa.

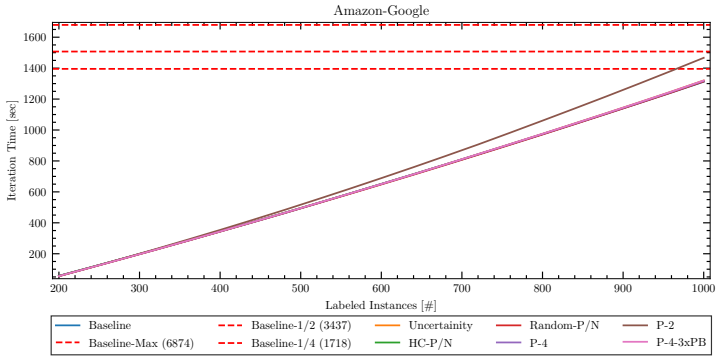
Figure C.1: Experimental results of iteration time with respect to labeled instances for dataset Abt-Buy.



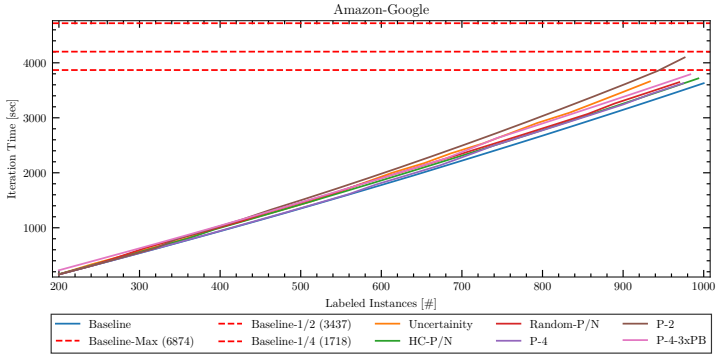
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

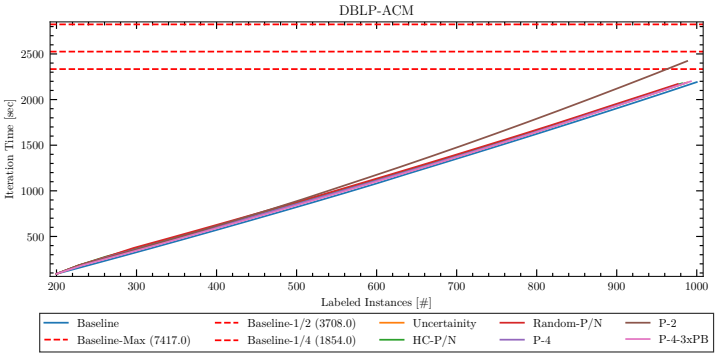


(c) 50/50 initial training set with LM DistilBERT.

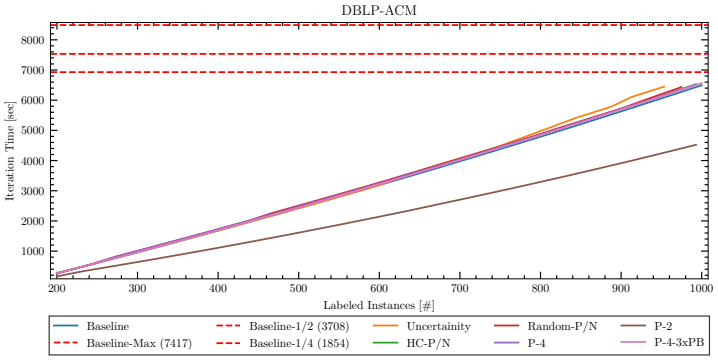


(d) 50/50 initial training set with LM RoBERTa.

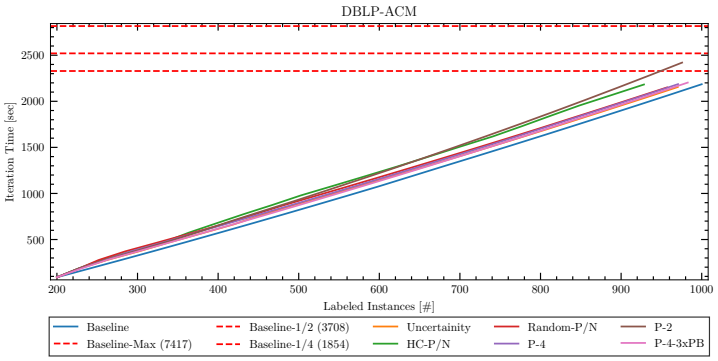
Figure C.2: Experimental results of iteration time with respect to labeled instances for dataset Amazon-Google.



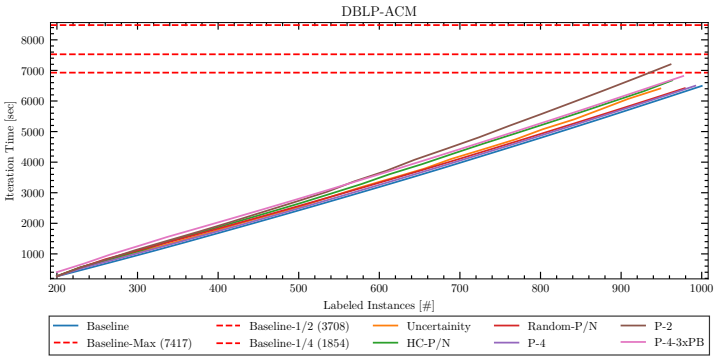
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

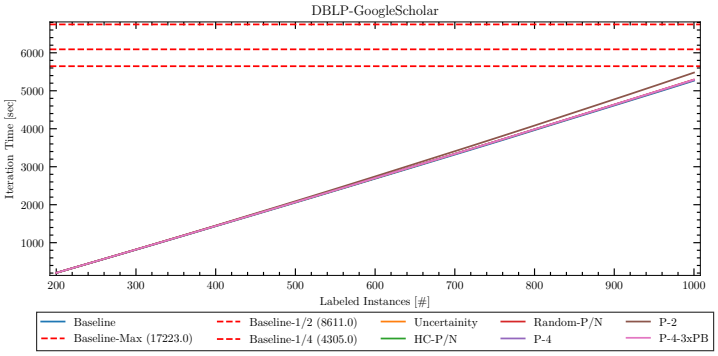


(c) 50/50 initial training set with LM DistilBERT.

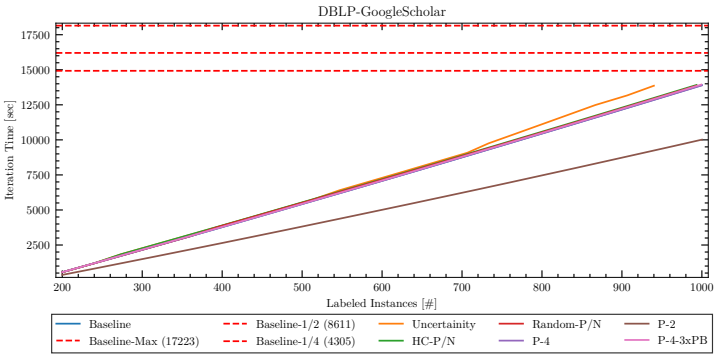


(d) 50/50 initial training set with LM RoBERTa.

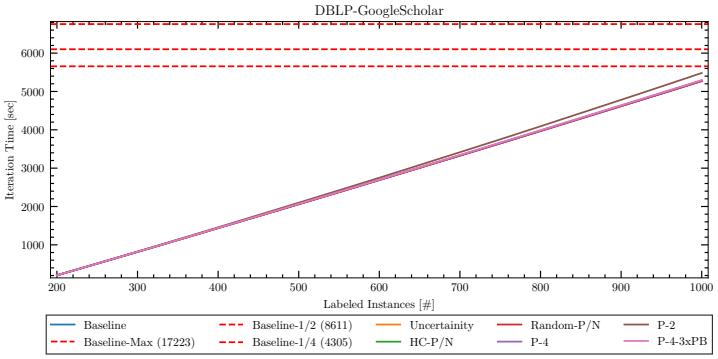
Figure C.3: Experimental results of iteration time with respect to labeled instances for dataset DBLP-ACM.



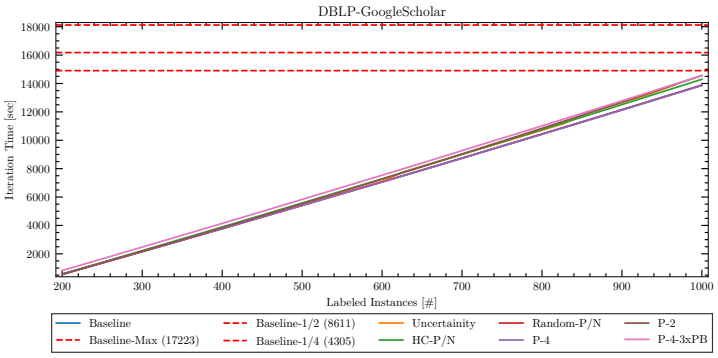
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

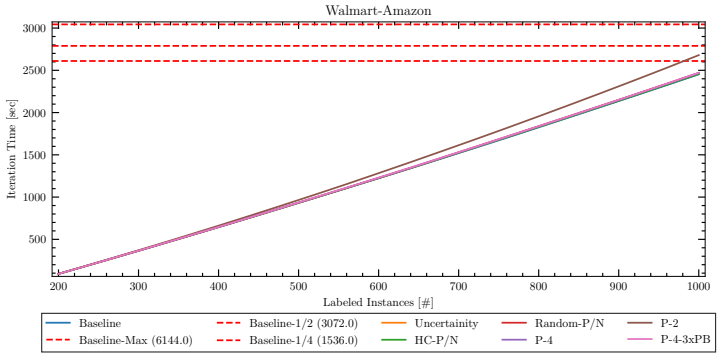


(c) 50/50 initial training set with LM DistilBERT.

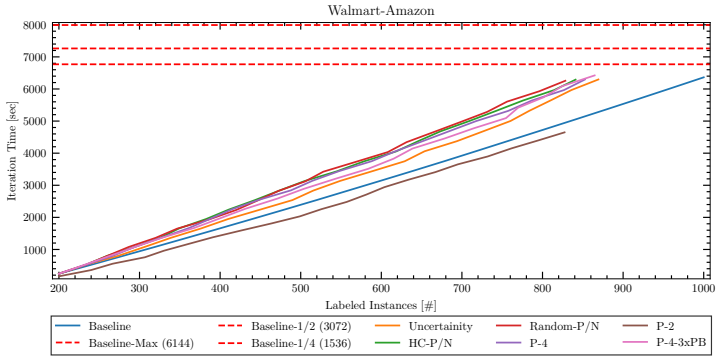


(d) 50/50 initial training set with LM RoBERTa.

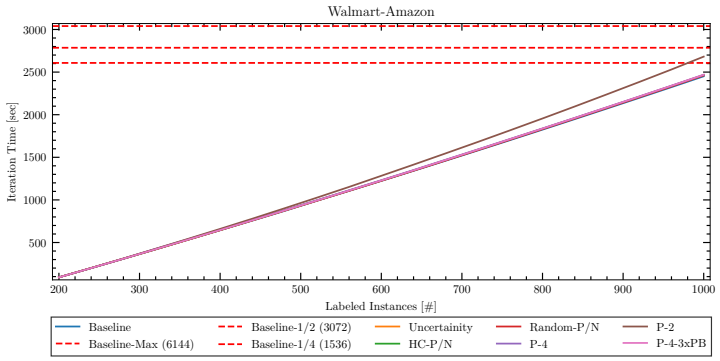
Figure C.4: Experimental results of iteration time with respect to labeled instances for dataset DBLP-GoogleScholar.



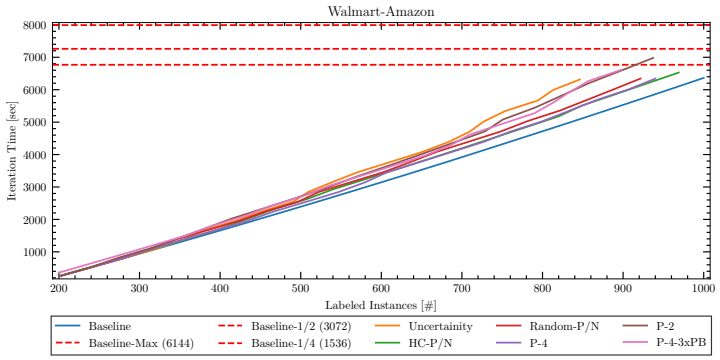
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.



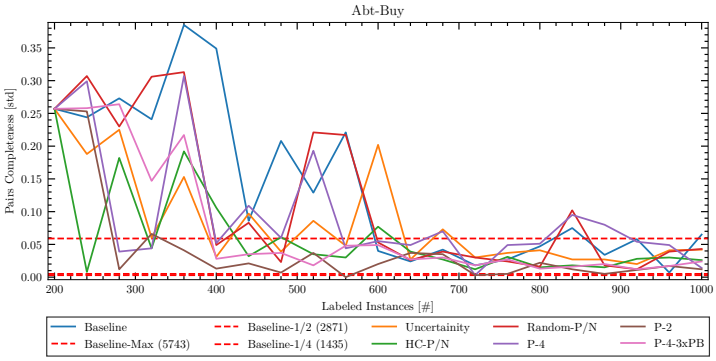
(c) 50/50 initial training set with LM DistilBERT.



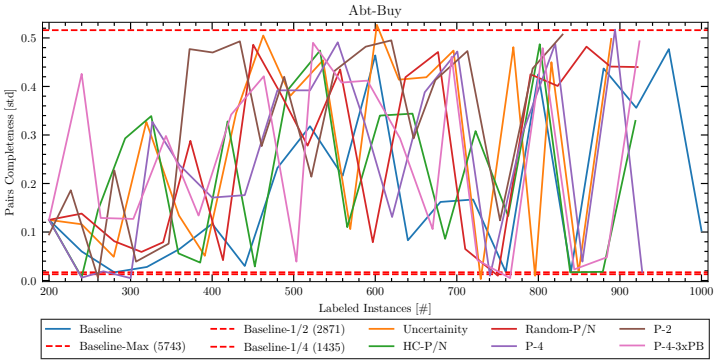
(d) 50/50 initial training set with LM RoBERTa.

Figure C.5: Experimental results of iteration time with respect to labeled instances for dataset Walmart-Amazon.

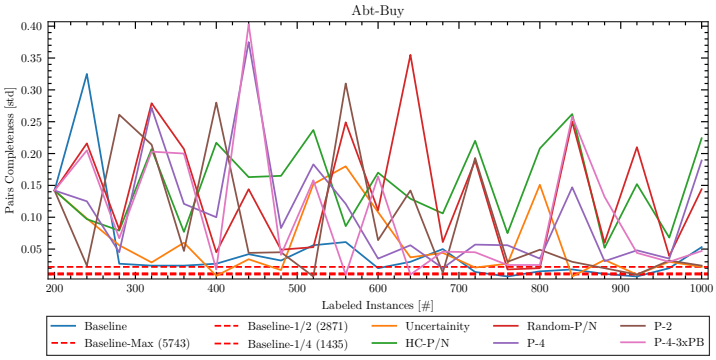
Appendix D Pairs Completeness Standard Deviation Results



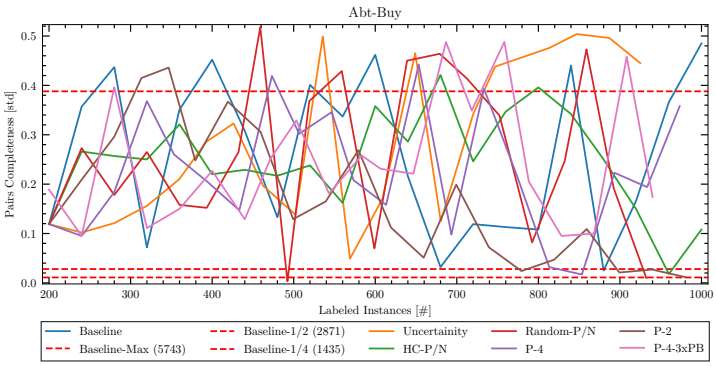
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

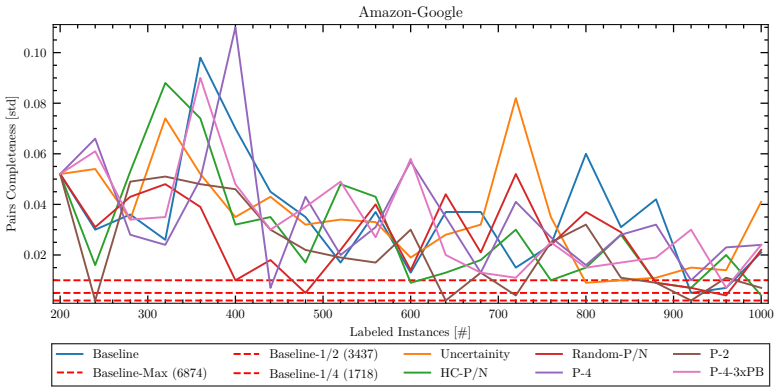


(c) 50/50 initial training set with LM DistilBERT.

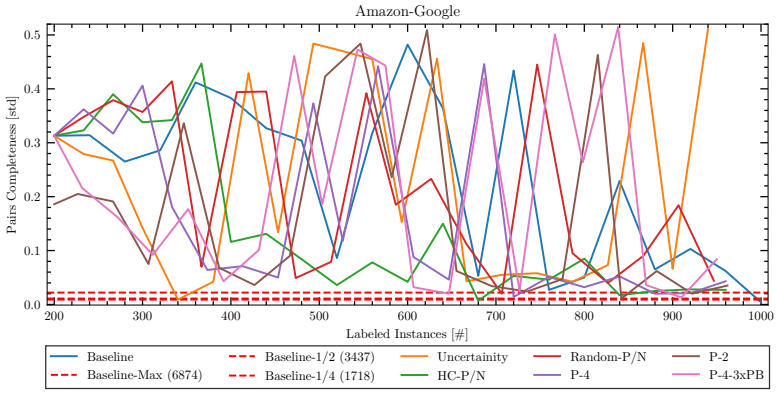


(d) 50/50 initial training set with LM RoBERTa.

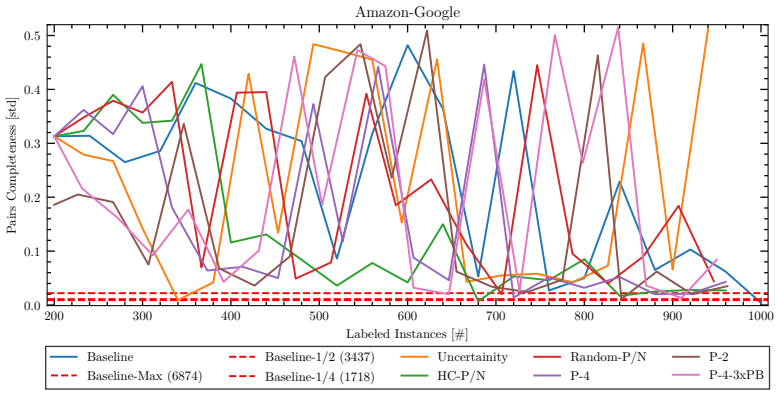
Figure D.1: Experimental results of PC score standard deviation with respect to labeled instances for dataset Abt-Buy.



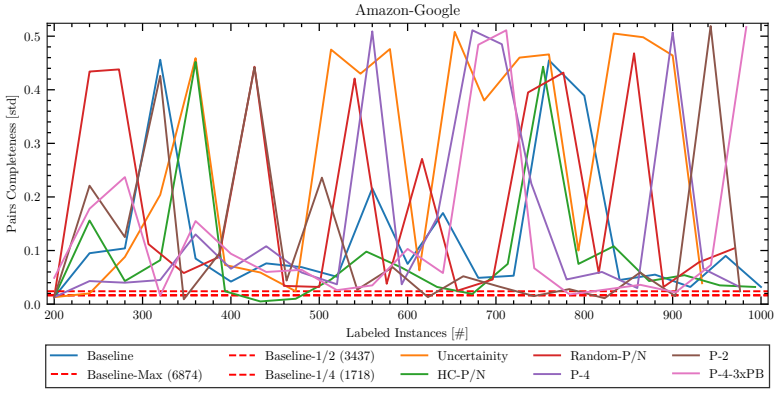
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

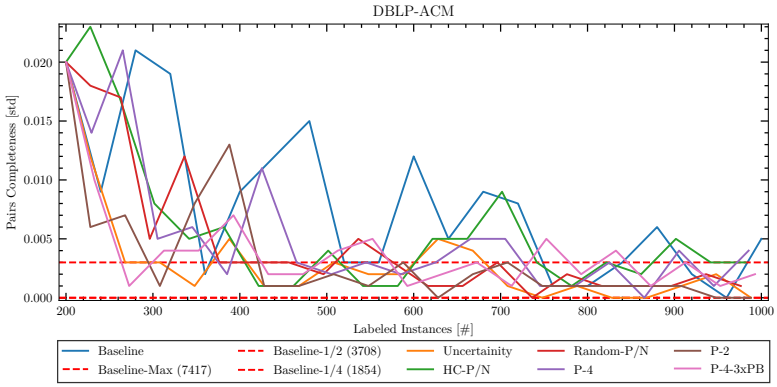


(c) 50/50 initial training set with LM DistilBERT.

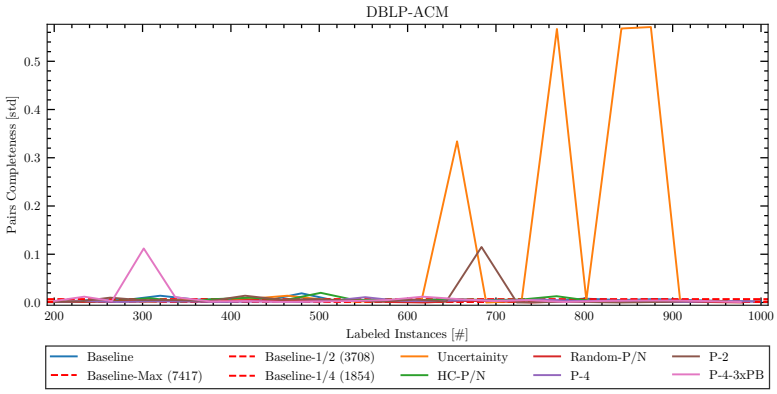


(d) 50/50 initial training set with LM RoBERTa.

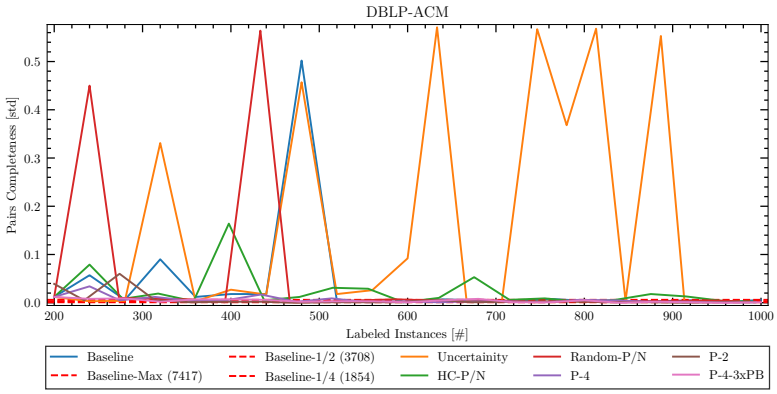
Figure D.2: Experimental results of PC score standard deviation with respect to labeled instances for dataset Amazon-Google.



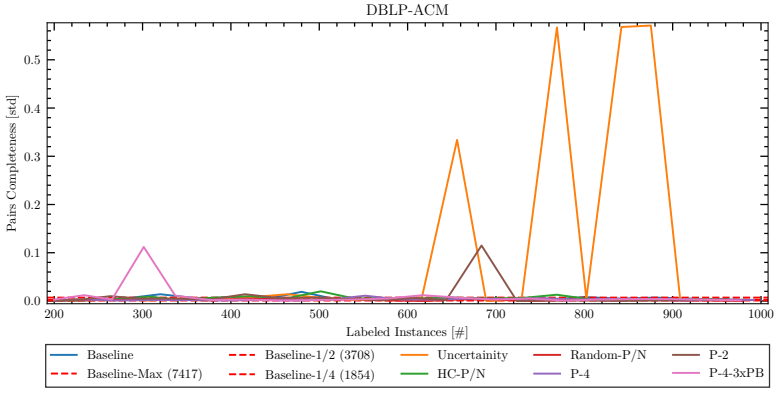
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

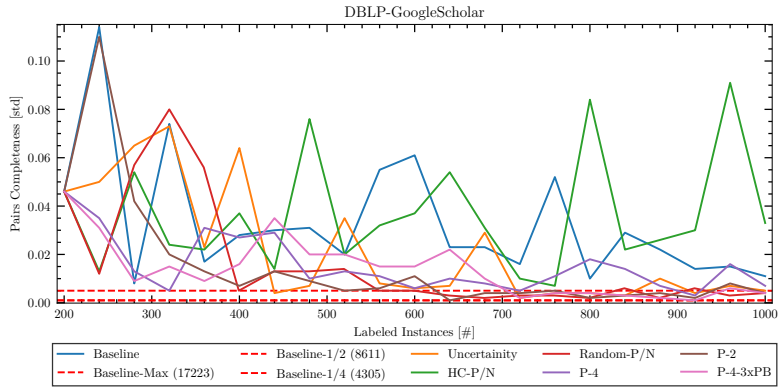


(c) 50/50 initial training set with LM DistilBERT.

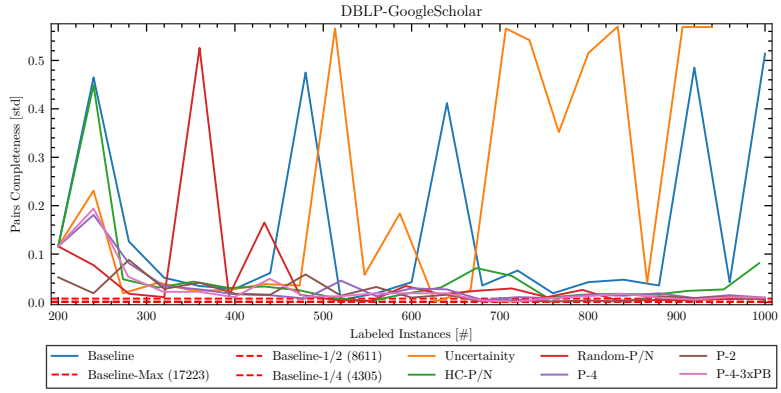


(d) 50/50 initial training set with LM RoBERTa.

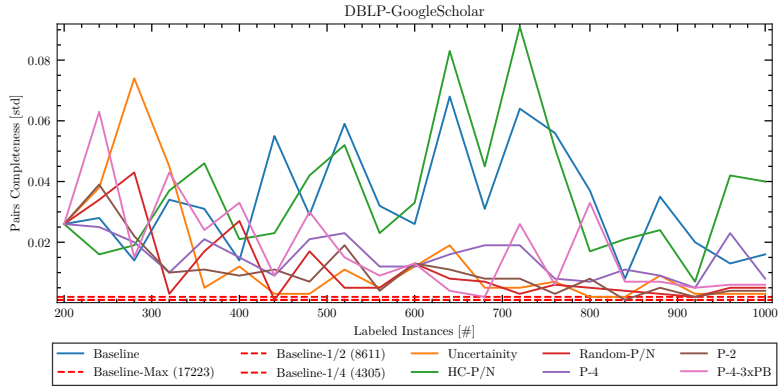
Figure D.3: Experimental results of PC score standard deviation with respect to labeled instances for dataset DBLP-ACM.



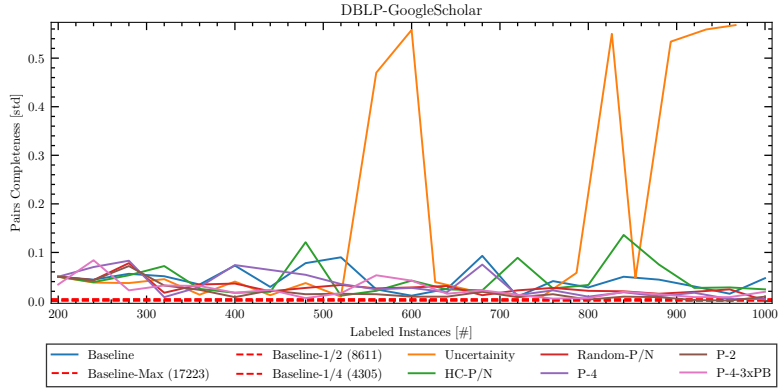
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

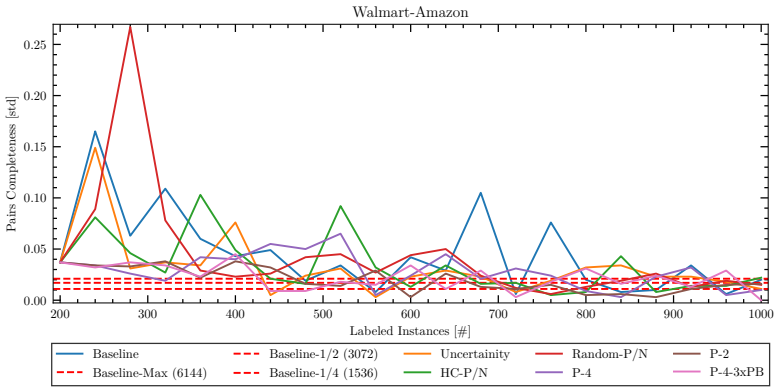


(c) 50/50 initial training set with LM DistilBERT.

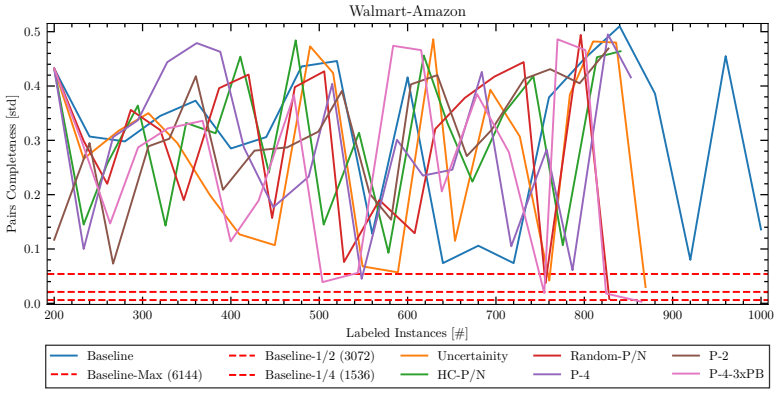


(d) 50/50 initial training set with LM RoBERTa.

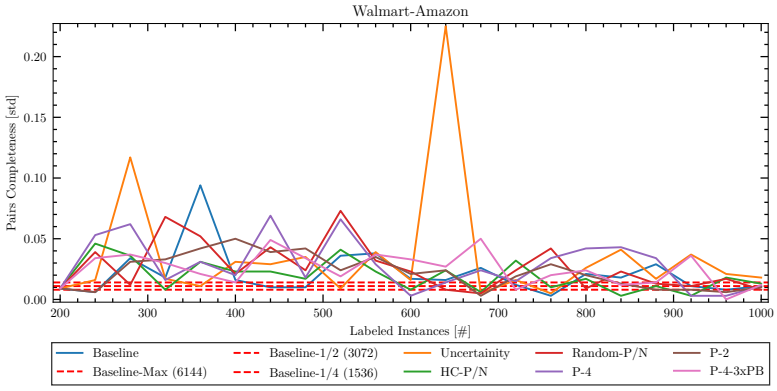
Figure D.4: Experimental results of PC score standard deviation with respect to labeled instances for dataset DBLP-GoogleScholar.



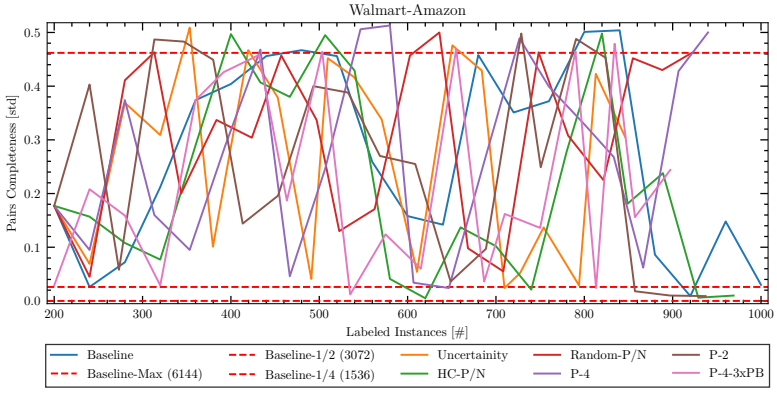
(a) Random initial training set with LM DistilBERT.



(b) Random initial training set with LM RoBERTa.

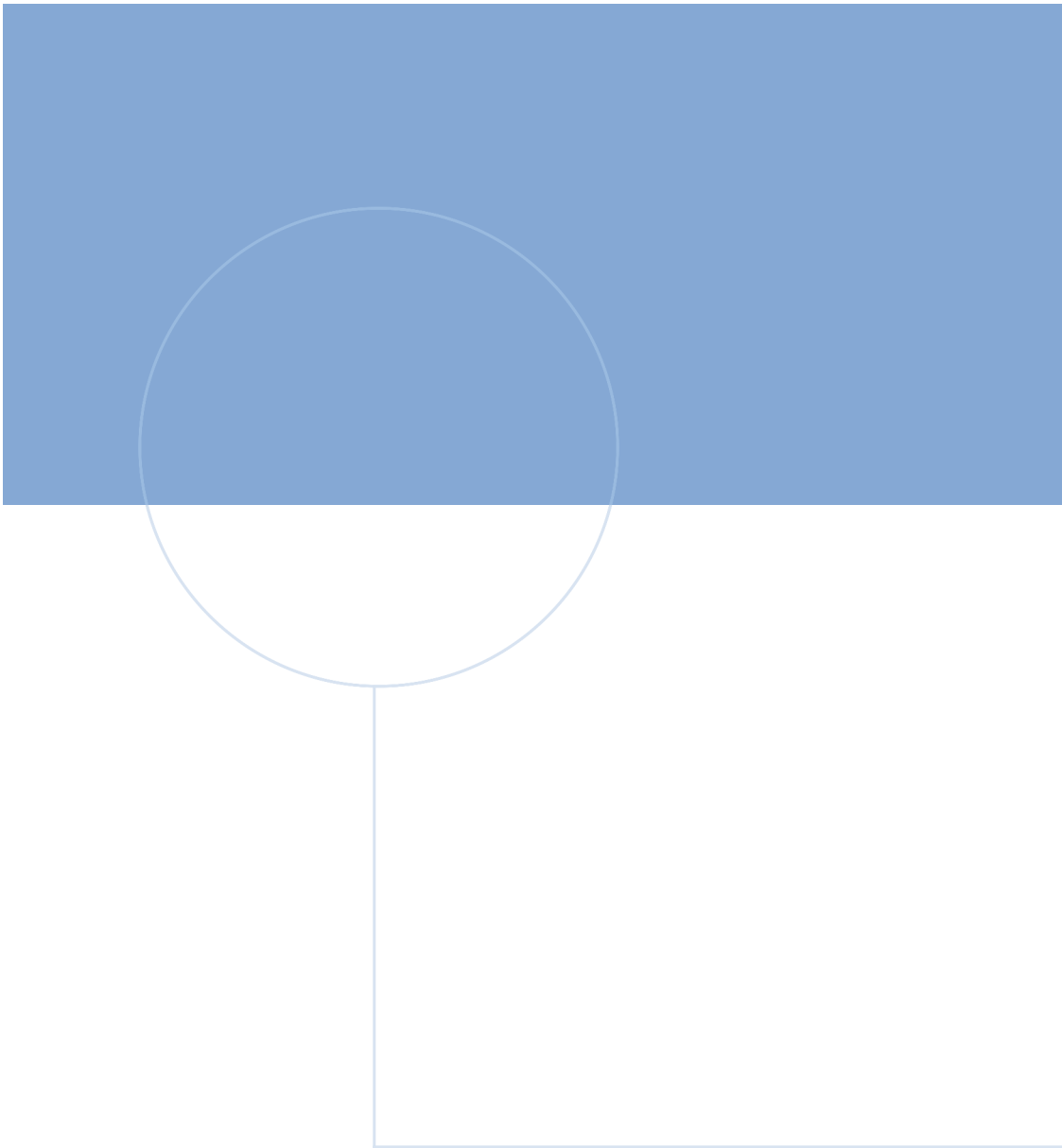


(c) 50/50 initial training set with LM DistilBERT.



(d) 50/50 initial training set with LM RoBERTa.

Figure D.5: Experimental results of PC score standard deviation with respect to labeled instances for dataset Walmart-Amazon.



NTNU

Norwegian University of
Science and Technology