Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Thea Tokstad

Pre-trained transformers with transfer learning and handcrafted-features for entity matching

Masteroppgave i Datateknologi
Veileder: Jon Atle Gulla
Medveileder: Nils Barlaug

Juni 2020

NTNU
Kunnskap for en bedre verden

COGNITE

Thea Tokstad

Pre-trained transformers with transfer learning and handcrafted-features for entity matching

Masteroppgave i Datateknologi
Veileder: Jon Atle Gulla
Medveileder: Nils Barlaug
Juni 2020

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU

Kunnskap for en bedre verden

Abstract

Entity matching (EM) is the task of identifying records from different data sources that refer to the same real-world entity [1]. Recently, transformer-based language models such as RoBERTa have been introduced to the task of EM, and proven to be very effective and achieve state-of-the-art results [2, 3]. However, this success comes with some limitations, the most important being that the methods require a significant amount of training data. Transfer learning is a promising technique to solve the problem with limited data.

This thesis examined the use of transfer learning with RoBERTa by pre-training the language model on EM benchmark datasets. Traditional machine learning algorithms based on string similarity metrics have achieved good performance with few training samples [4]. This thesis also examined if the strengths of handcrafted features such as string metrics combined with RoBERTa could improve the performance with limited data. The models have been evaluated on 12 EM benchmark datasets from various domains such as citations, product data, music and restaurants.

RoBERTa with transfer learning consistently outperformed the baseline, which in this study is a traditional machine learning EM system called Magellan. RoBERTa with transfer learning achieved a relative high F1-score with very few samples, and outperformed the baseline with an average F1-score of 30% when less than 200 training samples were used. Further, it achieved an average F1-score of 67.6% with no training data, with a max of 96.4% and a min of 58.03%. When all available training data was used, RoBERTa with transfer learning had the highest F1-score on 7 out of 12 datasets. In the less than 200 training samples range, the RoBERTa model with handcrafted features was outperformed by the baseline on 9 out of 12 datasets. When using all available training data, both models based on RoBERTa outperforms the baseline.

Sammendrag

Entitetsmatching (EM) er det å identifisere dataoppføringer fra forskjellige databaser som referer til samme entitet [1]. Nylig så har transformer-baserte språkmodeller blitt introdusert til EM. Disse har vist seg å være svært effektive til å løse EM problemet [2, 3]. Selv med stor suksess, så kommer disse modellene med utfordringer med at de trenger en betydelig mengde treningsdata. "Transfer learning", det å øverføre kunnskap fra andre datasett, er en lovende teknikk for å løse problemet med lite treningsdata.

Denne masteroppgaven har sett på "transfer learning" med den transformer-baserte språkmodellen, RoBERTa, ved å forhåndstrene den på EM datasett. Tradisjonelle maskinlæringsalgoritmer basert på strengmetrikker har også vist seg å være effektive med få treningseksempler [4]. Denne oppgaven så også på om styrkene til håndlagde funksjoner, slik som strengmetrikker, kan kombineres med RoBERTa for å forbedre effektiviteten med få treningseksempler. De beskrevne modellene har blitt testet på 12 forskjellige datasett fra ulike domener som bibliografi, produktdata, musikk og restauranter.

De transformer-baserte modellene slo konsekvent basislinjen, et tradisjonelt maskinlæringsbasert EM system ved navn Magellan, når de fikk nok treningsdata. RoBERTa forhåndstrent på EM datasett oppnådde et sterkt resultat med veldig få treningseksempler, og overgikk Magellan med en gjennomsnittlig F1-score opp til 30% når mindre enn 200 treningseksempler var brukt. Videre oppnådde den en gjennomsnitt F1-score på 67.6% med null treningseksempler, med et minimum på 58.03% og maksimum på 96.4%, på datasettene. Den hadde ogå den høyeste F1-scoren på 7 av 12 datasett når all treningsdataen var brukt. Når mindre enn 200 treningseksempler var brukt så klarte basislinjen å slå RoBERTa med håndlagde funksjonenr for 9 av 12 dataset.

Preface

This master thesis is submitted to the Norwegian University of Science and Technology (NTNU), and conducted at Department of Computer Science (IDI) as a part of the course TDT4900 - Master's Thesis. This thesis has been a part of a collaboration with Norwegian Research Center for AI Innovation (NorwAI) and Cognite. It was supervised by Prof. Dr. Jon Atle Gulla and Nils Barlaug.

Acknowledgements

I would like to thank my supervisor Professor Jon Atle Gulla for facilitating the collaboration with NorwAI and Cognite, and for valuable feedback. I would also thank and express my gratitude to my co-supervisor Nils Barlaug for guidance and assistance throughout the master. Thank you for constant pushing me forward and for always being available for all sorts of questions.

The experiments conducted were performed on resources provided by the NTNU IDUN computing cluster [5].

Contents

List of Figures

List of Tables

Abbreviations

AL Active learning

BERT Bidirectional Encoder Representations from Transformers

EM Entity matching

DL Deep Learning

FP False Positive

FN False Negative

LM Language model

NLP Natural language processing

RF Random Forest

RoBERTa Robustly optimized BERT approach

SOTA State-of-the-art

TL Transfer learning

TN True Negative

TP True Positive

Chapter 1

Introduction

1.1 Motivation

Cognite is a Norwegian company that develops and delivers a data operations platform aimed towards the industry. This platform receives large amount of data from industrial companies. The data is processed, analyzed and optimized by Cognite in order to be able to, amongst other things, predict maintenance, create 3D models and do simulations. The data that comes into the platform can be very dirty. The data comes from multiple sources and domains, and can contain duplicates, have missing data, use different schemas and use different naming conventions. There is usually no common identifier across the data that enters the platform. For the data platform to be efficient, it will have to know which data refers to the same real-world identity. The platform might have to link time series data from a sensor to a physical device given in an equipment list. This is difficult when the time series data is stored in one database, and the data about the physical sensors are stored in another database without any explicit connection to the time series. This linking has to be achieved with a very low tolerance of errors, which is challenging.

The aforementioned industrial data linking problem can be cast as an *entity matching* problem. Entity matching is the task of identifying records that refer to the same real-world entity. Table 2.1 gives an example of an entity matching problem. In the example, it can be challenging to determine which of the following four combinations are true matches. Across the records there is no common identifier, which could have solved the problem easily. Due to a lack of a common reference, the comparison of records needs to be based on the attribute values. Different methods such as string comparison metrics exist for doing this. However, two string that do not refer to the same entity can be as similar as two strings that do refer to the same real-world entity. Recently, better hardware and new machine learning based algorithms has allowed us to tackle this problem from a new direction.

Natural language processing (NLP) techniques, such as transformer-based lan-

guage models like BERT, have been introduced to the entity matching task and received state-of-the-art results [2, 3]. With these methods, there is a need for a larger amount of labeled data compared to traditional learning based entity matching algorithms. The process of labeling data is often a time-consuming task, and in the industry a domain expert is often needed. It is therefore of great interest to develop entity matching systems that excels on limited amount of training samples.

Transfer learning is a promising technique to solve the aforementioned problem where there is a limited amount of available training data [9]. The aim of transfer learning is to transfer knowledge from one domain to another.

Table 1.1: An example of the entity matching problem, where one want to match one record from Database A to one record from Database B.

(a) Database A: Product information from Amazon.

ID	**Title**	**Manufacturer**	**Price**
A1	microsoft word 2004 upgrade (mac)	microsoft	109.99
A2	microsoft word 2004 (mac)	microsoft	229.99

(b) Database B: Product info from Google.

ID	**Title**	**Manufacturer**	**Price**
B1	microsoft word 2004		209.99
B2	microsoft excel 2004 spreadsheet software mac apple & mac software		228.95

(c) Matches.

ID	**label**
A1-B1	no-match
A1-B2	no-match
A2-B1	match
A2-B2	no-match

1.2 Project goal and research questions

The goal of this thesis is to improve on the aforementioned industrial entity matching problem by utilizing state-of-the-art (SOTA) entity matching techniques. This somewhat broad goal can be summarized in the following research questions:

1. *How does the performance of transformer models compare to traditional machine learning based methods in a low-resource setting?*

The aim of this research question is to examine how well a transformer-based language model performs when there is a limited amount of labeled data available. The performance of the transformer model is compared to a traditional machine learning algorithm that is known for needing less data and has achieved SOTA results.

2. *How is the performance on entity matching affected when combining transformers with transfer learning on entity matching datasets in a low-resource setting?*

   Aforementioned, deep learning models are known for being data hungry, and it is of interest to develop models that perform well with limited training data. Transfer learning has given promising results when applied in similar contexts where training data is limited [9]. This research question seeks to examine the use of transfer learning applied to the RoBERTa language model for entity matching by pre-training the model on entity matching datasets.

3. *What is the impact of handmade features in combination with transformers in a low-resource setting?*

   Handmade features are in this setting string similarity metrics. String metrics measures the distance and/or the similarity between two strings. Traditional machine learning methods based on string metrics have achieved SOTA results on structured data. The aim of this research question is to combine the strength of string metrics on structured data with the strength of the transformer models which have given good results on dirtier datasets.

1.3 Contributions

The main contributions from this thesis can be summarized as:

- Present a transformer-based language model, RoBERTa, pre-trained on EM benchmark datasets. The model is called the *Transfer learning model* and are fine-tuned on a dataset for the EM task. The main advantage of the model is with very few samples were it consistently outperforms state-of-the-art EM solutions with up to 42% in F1-score for 50 samples. It also achieves a high F1-score of 58-96% with zero samples. With larger sample sizes, the advantage of the model decreases. To prior knowledge, this thesis is the first to pre-train a transformer-based model on entity matching datasets.

- This thesis also presents RoBERTa injected with handcrafted features, called the *Hybrid model*. The handcrafted features are generated from five string metrics. The impact of the string metrics are highest with few examples, where it achieves a score up to 80%, while RoBERTa only achieves a F1-score of 0%. However, with sufficient data the Hybrid model achieves a F1-score only 1-4% better than the RoBERTa model.

1.4 Approach

To answer the research questions, four different models will be trained and utilized. The first of these models is based on *Magellan*, a state-of-the-art supervised learning based EM system, and is used as the baseline. The second model is a transformer-based language model RoBERTa. The RoBERTa model is pre-trained on a large corpus of English text data. The third model combines transfer learning with the RoBERTa model by pre-training it on a set of related datasets. The last model aims to utilize the strength of traditional machine learning models based on string metrics by injecting these similarity measures into the RoBERTa model.

The models were tested and developed on twelve benchmark datasets for EM from different domains. The datasets can be split into three categories; structured, textual and dirty. F1-score is used to evaluate the models.

The four models can be summarized in the following list

- Magellan baseline model.
- RoBERTa transformer-based language model.
- RoBERTa transformer-based language model with pre-training (transfer learning).
- RoBERTa transformer-based language model with handcrafted features/string metrics (hybrid).

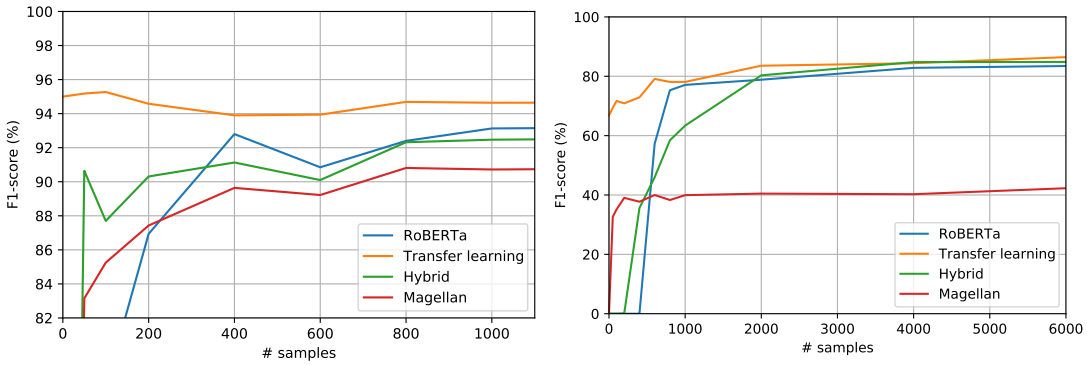
1.5 Results

The results can be summarized by the plots shown in Figure 1.1. The pattern shown in those two plots is present for all datasets, and the more in-depth results can be found in chapter 6. It is clear that Magellan is outperformed by the transformer-based models by a fair margin. Moreover, it is evident that using transfer learning with RoBERTa makes it easier for the model to perform well with few training samples. Across all datasets, the transfer learning based model achieves a high starting point score, with the lowest being a F1-score of 56% with 0 samples. In some cases, the F1 score can be as high as 95%. It is not immediately clear that adding handcrafted features improves the performance of the transformer model. The RoBERTa model was never the best model.

For many of the datasets, the transformer-based models converge when given enough training samples. What is enough training samples is case dependent and varies from dataset to dataset.

In addition to F1-score, training time needs to be considered when assessing performance. The Transfer learning model has the overall longest training time if

considering the pre-training on EM dataset. However, the pre-training only need to be performed once. If excluded, the Transfer learning model has the fastest training time by a few seconds compared to RoBERTa. Of the three transformer-based language models, the Hybrid model has the slowest training time by an average of 30 seconds with few samples.



(a) Structured DBLP-Scholar

(b) Dirty Walmart-Amazon

Figure 1.1: The average F1-score for the four models RoBERTa, RoBERTA pre-trained on entity matching datasets (Transfer learning model), RoBERTa in combination with handcrafted features (Hybrid model) and Magellan (baseline) for two datasets representing the results obtained in this thesis.

1.6 Thesis Outline

Chapter 1 - Introduction This chapter includes motivation, research questions and a summarization of the approach and results obtained.

Chapter 2 - Theory and background This chapter goes through the theory needed for understanding the development and usage of the models. It covers string similarity metrics, transformers and transfer learning.

Chapter 3 - Related Work In this chapter the related research as well as related problems are presented.

Chapter 4 - Data This chapter summarizes the benchmark datasets used with a focus on the challenges the datasets presents.

Chapter 5 - Method This chapter goes through the practicalities of the development, training and testing of the models. It contains a description of the tools and setup used.

Chapter 6 - Results The Result chapter presents the results from the conducting experiments.

Chapter 7 - Discussion The seventh chapter gives a discussion of the results obtained in the light of the research questions provided in this chapter.

Chapter 8 - Conclusion and further work The last chapters concludes the work done in this thesis, with a specific focus on the industrial applicability of the presented models, and provides suggestions for further work

Chapter 2

Theory and Background

This chapter gives the theoretical background needed to answer the research questions. It starts out by presenting the entity matching problem more formally, with a distinct focus on the challenges the entity matching problem phases. This section is followed by an go through of string matching, which is an important part of some of the entity matching models. The next section, section 2.3 introduces transformer models, and section 2.4 introduces the transformer-based language models BERT and RoBERTa. At last, section 2.5 goes through transfer learning, which is an important part of research question 2.

2.1 Entity matching

Entity matching (EM) is the task of identifying record pairs from two or more data sources that refer to the same real-world entity [1, 4]. More mathematically, the entity matching problem can be defined as: let A and B be two data sources with the attributes $(A_1, A_2, ..., A_n)$ and $(B_1, B_2, .., B_n)$. The attributes form the schema of the two data sources. The goal of entity matching is to find all the record pairs $(a_i \in A, b_i \in B)$ that refer to the same real-world entity. If a record pair does refer to the same real-world entity, it is defined as a true *match*. If not, the record pair is defined as a *non-match*. The attribute values are usually descriptions of the entities, such as name, date of birth or address [1].

In the literature, entity matching is also referred to as the following, depending on the domain

record linkage	string matching	merge/purge problem
link linkage	data matching	de-duplication
entity resolution	reference reconciliation	duplicate detection

Duplicate detection is the task of identifying records that refer to the same entity within one single data source [1], and can be cast as an entity matching problem. *String matching* focuses on matching of two strings, an important part of entity matching.

2.1.1 The Entity matching problem

The entity matching problem can be challenging for several reasons. Some of the major challenges for EM are described below.

No unique identifier

Usually, there is no common unique identifier across the datasets to be matched, such as serial numbers, social security number, an index or a key. On the other side, even with unique identifier one need to be confident that the identifiers are not corrupt. The identifiers can change over time and be different across the data sources. Additionally, one also needs to be sure that there are no duplicates within a dataset which have different identifiers that refer to the same real-world entity.

Low quality

Since there is generally no common identifiers in the data, EM often needs to rely on the attributes. This can be challenging when the data is of low quality. Some examples of factors that contributes to low quality are

- Missing values. Not all information are inserted into the datasets. This can lead to not having enough information about an entity as shown in Table 2.1. For the record in A2, the surname is missing. This means that the matching only can rely on the first name and the address.
- Wrong information. The information about i.e. a person can change over time. One can change name after marriage or move to another address. This leads to incomplete and wrong information about the entities.
- The attribute values have been shuffled. An example of this is given in Table 2.1.
- Abbreviations. As seen in Table 2.1 *William Henry* is abbreviated to *Bill H.* This can be intuitive to humans, but can be hard to machines to understand.
- Misspellings. The data can have been manually inserted, and due to human errors misspellings can occur.

The computational complexity

In this thesis, it is assumed that only one record from one database, match to exactly one record in another database. When matching two databases with size N, this could potentially lead to $N \times N$ comparisons in order to determine if a record pair refer to the same entity or not. The computational complexity therefore grows quadratically with larger datasets. At the same time, the amount of true matches only grows linearly.

Table 2.1: An example of the entity matching problem, where one want to match one record from Database A to one record from Database B.

(a) Database A: Product information from Amazon.

ID	First name	Last name	City
A1	William Henry	Gates	Seattle
A2	Bill		Arkansas

(b) Database B: Product info from Google.

ID	First name	Last name	City
B1	William S.	Gates	New York
B2	Bill H. Gates		Seattle

(c) Matches.

ID	label
A1-B1	no-match
A1-B2	match
A2-B1	no-match
A2-B2	no-match

Lack of training data

The status of true matches and non-matches are often unknown in many data matching applications, which means there is no golden standard or dataset to train on for machine learning based models [1]. Creating datasets consisting of matches/non-matches can be time consuming and may require domain experts. In academia, there exists quality assured and different datasets to train on. This if often not the case in the industry. For example, consider the industrial scenario where measurements from a pump need to be mapped to a list of equipment for maintenance. It is critical that the matches are correct, and it can be very time consuming, ineffective and expensive for a company to perform this mapping manually.

2.1.2 The entity matching process

The entity matching process is often divided into the two steps; *blocking* and *matching*. However, the process can be further divided into five steps [1] as shown in Figure 2.1, and this subsection will briefly go through these five different steps. This thesis focuses mostly on the matching step, which relies heavily on *record pair comparison* and *classification*.

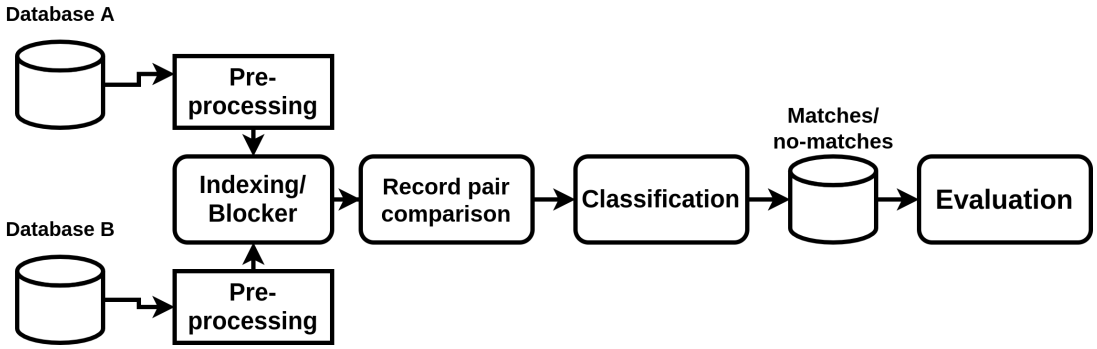


Figure 2.1: The general entity matching (EM) process of matching two databases.

Data pre-processing

The main purpose of this step is to assure that the format to the two data sources are the same, and the attributes follow the same structure. The data sources are also cleaned and standardised in this step. This step is similar to the pre processing step often done in machine learning.

Indexing/Blocking

After pre-processing, the data is ready to be matched. The naïve approach is to compare every record from one database to all records in the other database. Since only one record from one database only matches to one record in the other database, a large amount of the record pairs are clearly non-matches. This does not scale well in the long run, as the computational complexity grows quadratic with the size of the databases [1]. At the same time, the true matches only grows linearly. This leads to very imbalanced datasets in entity matching with few true matches compared to non-matches. To reduce this imbalance and the complexity of $O(n^2)$, different techniques such as *blocking* has been applied to the entity matching task. The resulting dataset after the filtering is called a *candidate set*. It is assumed that the candidate set does not include false negatives and only the records pairs that are likely to be a match based on the blocking technique. The common goal for all blocking techniques is to maximize the recall for the true matches.

In the blocking step, candidate sets are often generated with the use of a *blocking key*, also known as a *blocking criteria*. The blocking key splits the dataset into smaller blocks depending on the criteria to the blocking key. The record pairs that meet the requirement of the blocking key, and share the same value for the requirement are inserted into the same block. For example, when matching two citations databases, the blocking key could be based on having the same *Year*. Two publications with different year are obvious non-matches and filtered out from the candidate set. Only the records in the same blocks, have the same year, will be compared in the matching step.

Other strategies for blocking are *filtering* techniques or *n-nearest neighbours*. Fil-

tering often use similarity thresholds to apply rules [10]. For example every record pair with Jaccard score distance less than 0.3 are removed. The nearest-neighbour technique finds the n-nearest record pairs.

The downside of traditional blocking techniques is that usually the blocking function need to be specified, often based on domain-knowledge [11]. Recently, blocking techniques based on deep learning embeddings have been applied to maximize recall. Recall is further described in chapter 5.

Record pair comparison

After the blocking step, the record pairs $(a, b) \in C$ are compared, resulting in a *comparison vector* of numerical similarity values [1]. The comparison vectors are often the basis of traditional machine learning algorithms.

Traditionally, string similarity has been used to indicate the similarity or dissimilarity of the two records [12]. It is assumed that the more similar a record pair is, the more likely it is that the pair refers to the same real-world entity. The string metrics are often chosen based on the attribute values.

With the rise of deep learning in NLP, new techniques have been applied to the record comparison step such as word embeddings [2]. The advantage of word embeddings is that it removes the need to generate features based on string metrics which often need to be manually defined.

Classification

The main task in this step is to identify all the true matches in the candidate set. The simplest model used for classification just sums up the values from the comparisons vectors obtained. Based on a threshold, all record pairs with a sum over the threshold is classified as a match.

Traditionally, classification for entity matching used probabilistic models. The downside with these models is that they only classified each record pair independently.

Supervised classification algorithms use the comparisons vectors as feature vectors. The features are employed to train the models, in order to classify the record pairs into a match or a non-match [1].

As with the record pair comparison, deep learning models have also been applied to the classification step. Brunner and Stockinger [2] and Li et al. [3] recently introduced transformer-based language models to the entity matching task. The pre-trained transformer models has received state-of-the-art result for EM.

Other techniques that also can be used in the classification step are active learning, transfer learning and unsupervised models.

Evaluation

The main goal of the classification models mentioned is to achieve a high quality. To be able to assess the quality a golden standard with known true matches and non-matches are required. With the help of a variety of evaluation measures, the performance of the models can be computed [1]. Some data applications has a very low tolerance of error, so it is important to evaluate the matching.

2.2 String matching

String matching is the problem of finding strings that refer to the same real-world entity [13]. For example, the string *William Gates* from one database refers to the person *William H. Gates* in another database. In data integration, string matching plays a critical role to perform for example schema matching or data matching. The definition of the string matching problem is similar to entity matching: Given a set X and another set Y, the problem is to find all string pairs $(x \in X, y \in Y)$ such that x and y refer to the same real-world entity [13]. One of the major challenges in string matching is accuracy, due to the two string that refer to the same real-word entity may be different. Some reason for the difference are misspelling of words, abbreviations, different formatting and shuffling of the characters in the string. A solution to solve the accuracy problem is to use similarity measures.

2.2.1 Similarity measures

Similarity measures are used to find the similarity score in the range [0,1] between two strings. The higher the similarity score is, the more likely it is that the two strings match. Two other terms with the same concept are *cost* and *distance measures*. The smaller the value of the score for these two terms, the higher similarity of two strings [13].

The similarity measures can be categorized into four groups according to how the measures treat the input strings. The four categories are:

Sequence-based The sequence-based similarity functions view the input as a sequence of characters. The main computation of these measures are to compute the cost of transforming one string into another string. Examples: edit distance, Needleman-Wunch and Jaro-Winkler measure.

Set-based The set-based measures consider the input as a set or multisets of tokens. Several methods can be used to generate a set of tokens from the input string. A method is delimiter, which splits the words by the space character. Another is q-gram, which split the string into sub-string of size q, often 2 or 3. Examples: Jaccard, Dice and TF/IDF.

Hybrid The hybrid measures combine components from the sequence-based measures and the set-based measures. Examples: Monge-Elkan and Generalized Jaccard.

Phonetic The phonetic measures look at how the input strings sound, in contrast to the three other which consider the characters and its place in the strings. These measures are often used to match names, because names often can be misspelled. Examples: Soundex.

Edit distance/Levenshtein distance

The basic Edit distance, also known as Levenshtein distance $d(x, y)$, computes the smallest number of operations that needs to transform string x into string y. The various operations that can be performed on a character is: delete, insert or substitute. Since the minimal cost of transforming x to y is the same as the minimal cost of operations to transform y to x, $d(x, y)$ is symmetric.

To obtain the similarity function $s(x, y)$, the edit distance $d(x, y)$ is converted as follow:

$$s(x, y) = 1 - \frac{d(x, y)}{\max(|x|, |y|)}. \tag{2.1}$$

Levenshtein performs better on short strings, than on longer strings, and where small number of differences is expected. It can computed on longer strings, but the computational cost is higher, so to speed up the process shorter strings is smarter.

The Jaro-Winkler measure

The Jaro-Winkler measure is a type of edit distance mainly used to compare names [1]. The measure is a modification of the Jaro-measure, which was developed to focus on comparing short strings. The Jaro-score is computed as follows:

$$\text{jaro}(x, y) = \frac{1}{3}(\frac{m}{\mid s_1 \mid} + \frac{m}{\mid s_2 \mid} + \frac{m - t}{m}), \tag{2.2}$$

where $\mid s_i \mid$ is the total length of string s_i, t is the number of transpositions and m is the number of characters which are matching between s_1 and s_2. Transposition is the number of operations needed on characters to transform s_1 into s_2. The matching of two characters respectively from s_1 and s_2 is only counted as *matching* if they are the same and are not $\lfloor max(\frac{|s_2||s_2|}{2}) \rfloor$–1 characters apart from each other.

The Jaro-Winkler measure checks if the two strings share a prefix and can therefore likely be a match, even though the Jaro-score is low. Jaro-Winkler score is as follows:

$$\text{jaro-winkler}(x, y) = (1 - PL \times PW) \times \text{jaro}(x, y) + PL \times PW, \tag{2.3}$$

where PW is the weight given to a prefix, and PL is the length of the longest common prefix between the string [13].

Dice coefficient

The Dice score is a set-based measure that calculates how similar a set of q-grams, X is to another set of q-grams, Y [1].

The Dice score is as follows:

$$\text{dice} = \frac{2\times \mid X \cap Y \mid}{\mid X \mid + \mid Y \mid}. \tag{2.4}$$

If using true positives (TP), false positives (FP) and false negatives (FP) the Dice score can be written as:

$$\text{dice}(X, Y) = \frac{2TP}{2TP + FN + FP}. \tag{2.5}$$

The difference with Dice coefficient from Jaccard is that Jaccard only looks at true positives when calculating the similarity.

Generalized Jaccard

The Jaccard measure examines the overlapping tokens between two strings s_1 and s_2 and is computed as:

$$\text{jacc}(x, y) = \frac{|B_x \cap B_y|}{|B_x \cup B_y|}, \tag{2.6}$$

where B_x and B_y is the set of tokens for string x and y respectively. The disadvantage of the Jaccard score is that the overlap of tokens are restricted to be the same to be considered to be the same. The generalized Jaccard measure is an extension to the Jaccard measure. The benefit of the generalized Jaccard is that is capture misspellings of the words in the input strings, where the Jaccard score would not.

The first step to find the generalized Jaccard score is to convert the input strings into a set of tokens, B_x and B_y [13]. A similarity measure s is applied to every pair $(x_i \in B_x, y_i \in B_y)$. Only the scores that are equal to or a above a given threshold α is kept. The result is a bipartite graph with the sim score s as weights on the edges between the pairs. The maximum-weight matching M is found and normalized to get a score in the range $[0, 1]$. The generalized Jaccard is:

$$GJ = \frac{\sum_{(x_i,y_i)\in M} s(x_i, y_i)}{\mid B_x \mid + \mid B_y \mid - \mid M}, \tag{2.7}$$

where the normalization is the sum $\mid B_x \mid + \mid B_y \mid -M$.

Monge-Elkan

Monge-Elkan measure is a type of hybrid similarity and finds the similarity between two sets of tokens. The measure was originally developed to compare strings containing several words [1]. This can occur when matching business

names or addresses. The first step is to covert the two strings s_1 and s_2 into multiple substrings, $s_1 = A_1, A_2, ..., A_n$ and $s_2 = B_1, B_2, ..., B_m$ [13]. A solution to achieve the the lists of tokens A_i and B_j are to split the input strings by whitespace or delimiter. The Monge-Elkan uses a secondary similarity measure s', such as Levenshtein or Jaro-Winkler. This secondary similarity function is used to find the best matching pairs between the tokens from A_i and B_j. The Monge-Elkan score is computed as:

$$s(s_1, s_2) = \frac{1}{n} \sum_{i=1}^{n} \max_{j=1}^{m} \text{s'}(A_i, B_j), \tag{2.8}$$

2.3 Transformers

The Transformer is a special type of neural networks first introduced by Vaswani et al. [6]. Transformers are primarily used in natural language processing (NLP) tasks [6]. However, it generalizes also well to other tasks such as entity matching. The Transformer uses self-attention mechanisms to draw global dependencies between the input and the output. This means that the model is able to learn the relationships between words in a sentence, regardless of the position of the word in the sentence. The self-attention mechanism also allows the transformer to perform calculations in parallel, which has been a bottleneck for Recurrent Neural Networks (RNN).

2.3.1 The architecture

The Transformer has an encoder/decoder structure heavily based on attention, which is further described below. Given an input sequence of symbol representations $(x_1, x_2, ..., x_n)$, the encoder maps this representation to a sequence of continuous representations $\mathbf{z} = (z_1, z_2, ..., z_n)$. The decoder takes the output from the encoder, z, and generates an output sequence $(y_1, y_2, ..., y_m)$.

The encoder and the decoder follows the structure shown in Figure 2.2. Both are composed of a stack of N, often 6, identical layers. Each layer consists of the two sub-layers: a self-attention layer and a feed-forward neural network layer [6].

Positional Encoding

The be able to know the position of the tokens in the input sequence, *positional encoding* are added to the input embeddings. The positional embeddings are added to both the inputs at the first encoder and to the first decoder in the stack.

Encoder

The first step is to convert the input sequence into a vector of dimension 512 by using an embedding algorithm. The list of word embeddings are the input to the first encoder. Each token in the input sequence flow through the two sub-layers

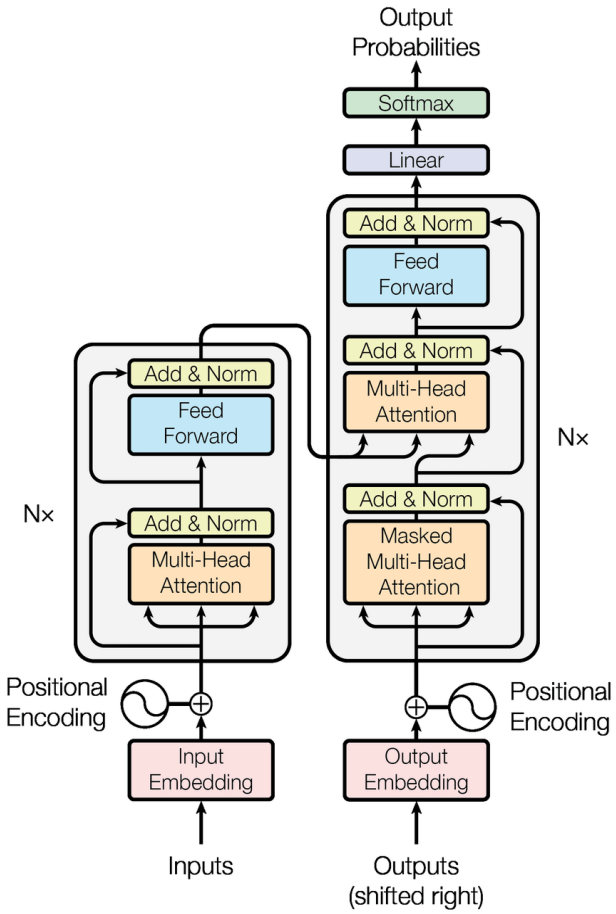


Figure 2.2: The model architecture for transformers. The architecture consists of N encoders and decoders. The encoder and decoder is built with two sub-layers: *attention* layer and a *feed forward neural network* layer. Source [6].

in its own paths. In the self-attention layer, there are dependencies between the different paths. This layer look at other words in the input sentence as it encodes a specific word. A further description of how this layer operates are given below. The output of the attention-layer is then given to the feed-forward neural network. This layer does not have any dependencies, which makes the transformer able to operate in parallel. The encoder also has a residual connection around it. This allows the gradients to flow directly through the network.As shown in figure 2.2 the Residual layer is followed by a normalization layer. This layer normalize the neurons in the network, in order to reduce the training time. The output of each sub-layer is:

$$\text{LayerNorm}(x + \text{SubLayer}(x)), \tag{2.9}$$

where SubLayer(x) is a function in the sub-layer. The last encoder of the stack transforms the output into a set of attention vectors, K and V, which are passed to every decoder.

Decoder

The decoder follows the same structure as the encoder, but it includes a third layer. The third layer, the *encoder-decoder attention* performs multi-head attention on the output from the encoder, see figure Figure 2.2 [6]. The input to the encoder-decoder attention layer are the set of attention vectors from the encoder, containing key vector, K, value vector, V, as well as the queries from the previous decoder layers in the stack. The benefit of this, is that every position in the decoder can attend over all the positions in the input sequence [6]. The self-attention layer in the decoder has been modified compared to the attention layer in the encoder. The self-attention layer mask future positions, and the output embeddings are shifted one position to the right. Due to the offset and the masked positions, the predictions of x_i can only be dependent on the preceding outputs. Also, the outputs less than x_i.

Similar to the encoder, the decoder also has a residual connection around the sub-layers in the decoder, followed by a normalization layer. The output of the decoder stack is a vector of floats.

The linear layer and the softmax layer

The two final layers are the linear layer followed by the softmax layer. Simplified is the linear layer a fully neural network. This layer takes the output from the decoder and projects it into a larger vector, called *logits vector*. If the model knows 5000 unique Norwegian words, then the logits vector will contain 5000 cells, one cell for each word. Each cell contains a score for the unique word. The softmax layer turns the scores from the linear layer into probabilities. The cell that contains the highest probability is chosen, and the associated word is produced.

Training the transformer model

The training of the transformer model is typically with labeled datasets. The process of training is the same as the aforementioned techniques. The output of the training are compared to the correct result.

2.3.2 Self-attention

The idea of self-attention is that a word can be expressed as a weighed combination of its neighborhood in the input sentence [2]. The language model can then be able to pay attention to relevant words that are close in its neighborhood, which can be seen in figure 2.3. For example, the computation of y_2 is based on the comparisons between x_2 and the preceding element x_1. One way to compute the comparison is to take the dot product, then normalize it with a softmax function to get the output y_2.

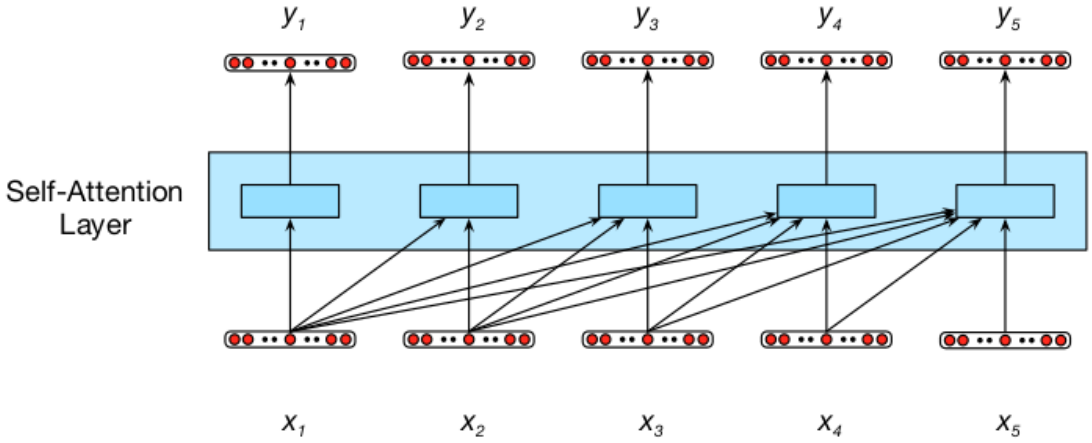


Figure 2.3: The flow of the input sequence $x_1, x_2, ..., x_n$. Source [7].

2.3.3 Multi-Head Attention

The different words in a sentence can have multiple relations to other words in a sentence. With only one single layer of attention it is hard to capture all of the parallel relations in the input [7]. Transformers solve this with *multihead attention layers*. These attention layers, also called heads, runs through several attention layers in parallel with its own set of parameters. Each head with its distinct parameters can now learn the different relations between the words.

The input to each head is a set of key, query and value vectors. The output of the h heads are h vectors with the same length. Since the feed forward network only expects one matrix with a vector for each word, the h matrices are concatenated into one matrix and linearly transformed to the original dimension. Figure 2.4 shows the overall structure of the multi-headed attention layer.

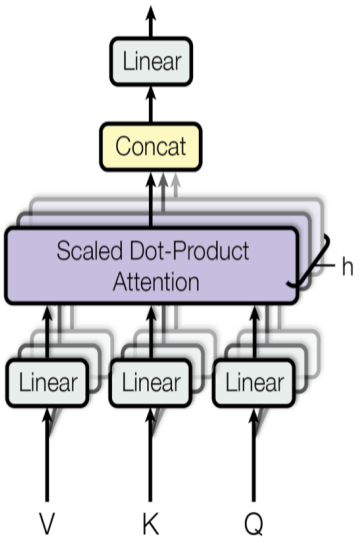


Figure 2.4: Multi-head attention consisting of h self-attention layers. Source [6].

2.4 Language Models

Various deep learning models such as recurrent neural networks (RNN) and long short-term memory with word embeddings have been used to train a matching model [14, 15]. Recently, pre-trained language models have proved to be effective for improving a wide range of NLP task [8], and has achieved state-of-the-art results in entity matching. Examples of such pre-trained models are BERT [8], RoBERTa [16], GPT-2 and XLNet. Many of these models are based on deep neural networks with multiple transformer layers. In surveys conducted by Li et al. [3] and [2], RoBERTa gave on average the best scor for entity matching and it is based on the popular transformed-based language model, BERT [2].

2.4.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a language model (LM) developed by Google in 2019 [2]. It has achieved state-of-the-art results in various Natural Language Processing (NLP) tasks. BERT is pre-trained on large text large text corpora such as Wikipedia. There are two steps in the BERT framework: pre-training on large unlabeled data and fine-tuning on downstream tasks with labeled data [8].

BERT architecture

The input to BERT needs to be converted into a tokenized sequence, for it to be meaningful for the model. BERT uses WordPiece embeddings to tokenize the input sequence. Segment embeddings are also added to every token, to indicate which sentence the token belongs to. The position embeddings are used to denote the position of a token in the sequence. The input representation for the BERT model is shown in figure 2.5. Two special tokens, [CLS] and [SEP] are also added to the input. [CLS] is always the first token of the sequence and are used for classification. [SEP] is inserted at the end of each sentence to separate two sentences [8].

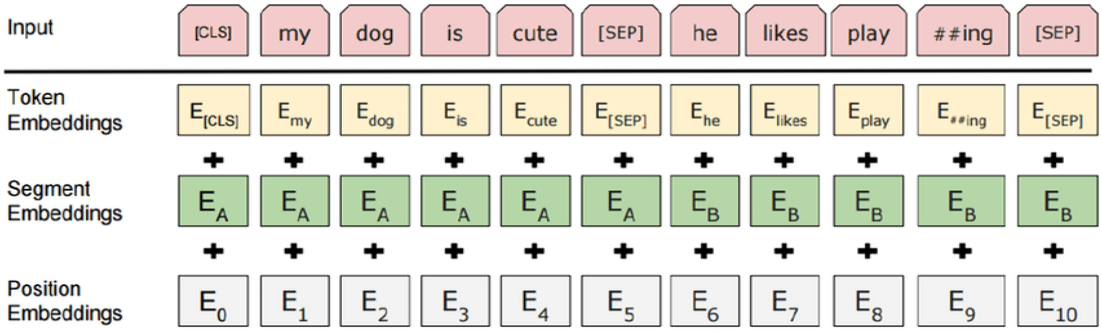


Figure 2.5: BERT input representation. E indicates the input embedding. Source [8]

BERT is built with the same transformer structure proposed by [6] with multiple layers, typically 12 or 24 [3]. As the name implies, BERT is bi-directional. Instead

of predicting the next word as most unidirectional language models do, BERT uses both the right and the left context of the query token to predict the masked token, [MASK] see figure Figure 2.6 [2]. With the masked token, BERT is designed as a masked language model (MLM). This means that BERT randomly mask a certain percentage of the tokens from the input sequence [8]. During fine-tuning of the parameters in the downstream task, [MASK]-tokens are not represented. To compensate for this, during the pre-training of the parameters not all masked tokens are replaced with [MASK]. Approximately 10% are replaced with a random token, and 10% unchanged leaving 80% to be replaced with [MASK]. MLM makes BERT able to learn to reconstruct the original data by predicting the masked tokens.

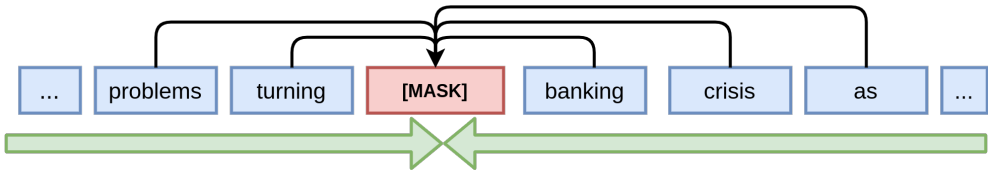


Figure 2.6: BERT use both the left and right context of the masked token, [MASK]. Source [2].

For some downstream tasks such as NLP and Question/Answering, it is important to be able to understand the relationship between two sentences. BERT solves this by introducing another training task named Next Sentence Prediction (NSP) [2]. When training the model with two inputs, sentence A and B, BERT chooses B 50% of the time as the actual next sentence and labels it *isNext*. 50% of the time BERT chooses a random sentence from the corpus, which are labeled *notNext* [8]. This is to make BERT able to predict if a sentence is followed by the next sentence.

2.4.2 RoBERTa

Robustly optimized BERT approach (RoBERTa) is another language model released in 2019, and is mainly a modification of BERT. The authors of the RoBERTa paper [16] meant that BERT was under-trained. The modifications proposed was [2]:

- More training data. RoBERTa is trained on 160 GB of text, while BERT is trained on 16 GB.
- Longer training
- Training with larger batch sizes. BERT uses batch size of 256, while RoBERTa train with batch sizes of 2K and 8K [16].
- Exclude next sentence prediction (NSP). RoBERTa shows that by removing NSP, the performance of the downstream tasks were slightly improved.

- Dynamic masking instead of static masking. BERT only performs masking one time during pre-processing, so the same mask is used for each training instance. RoBERTa looks at generating masks every time a sequence is fed into the model. The authors concluded with dynamic masking performed slightly better than static masking, in addition to be more efficient [16].

2.5 Transfer Learning

Transfer learning seeks to improve the performance on a target domain by transferring knowledge from a source domain [9]. In this way, the dependence for large training data for the target domain can be reduced. Therefore, TL is a promising technique in low resource settings [17]. In recent years, transfer learning has been used to pre-train models on large corpus of text, such as BERT, in order to use this pre-trained model to be fine-tuned on specific tasks in another domain.

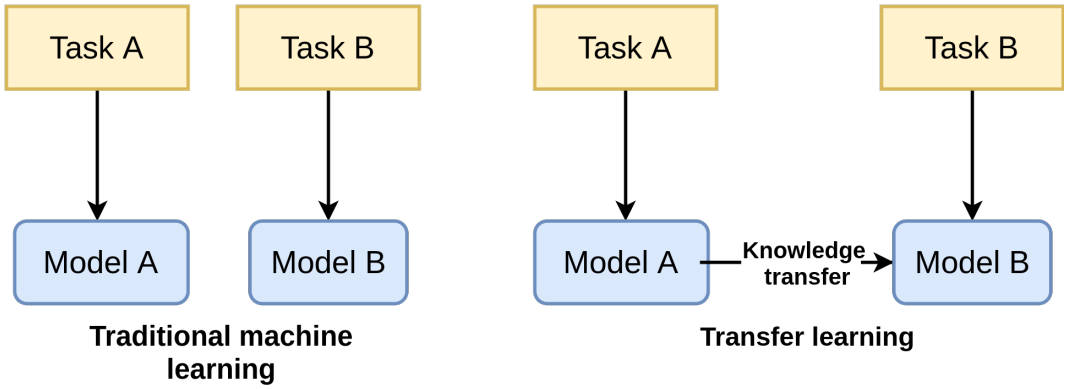


Figure 2.7: A simple flow of transfer learning from one domain or dataset (Task A) to another domain or dataset (Task B) compared to traditional machine learning.

Given one or multiple source datasets $D_{S1}, D_{S2}, ..., D_{Sn}$ the aim of transfer learning is to utilize the knowledge from the *source* and transferring it to the classifier for a *target* dataset, D_T, with limited training data [18].

Transfer learning can be categorized into the following categories depending on how much training data the source has compared to the target dataset:

Unsupervised domain adaption The target dataset contains zero training data. The source dataset contains adequate data. [18].

Supervised domain adaption A second scenario is when there is adequate labeled training data for the source datasets, but the labeled data in the target dataset is limited.

Semi-supervised domain adaption The amount of labeled data are limited for both source and target.

2.5.1 Instance weights for multiple datasets in source

The same way entity matching suffer from imbalanced datasets, transfer learning also suffer from this when multiple datasets are used in source. D_S^L which is the number of training labels in the training set for source, is often much larger than D_T^L, which is the number of training labels for the target dataset. Consider the case where $D_S^L = 1000$, while $D_T^L = 50$ are used for training a model. Then if this is blindly used for training, the source could in worst case *swamp* the target dataset [18]. A solution could be to reduce the datasets for source to get a balanced dataset, but it is not always the case to just discard 950 samples due to data scarcity. This limitation can be addressed by applying instance weights to the training data. The instance weights should be different, with a lower *instance weight* to $t \in D_S^L$ and a higher weight to $t \in D_T^L$.

2.5.2 Negative transfer

An important issue to be aware of with transfer learning is negative transfer. This happens when the transferred knowledge from the source domain lead to a negative effect of learning for the target domain, also a reduced performance. This mostly occurs when the source domain and target domain are dissimilar [9].

Chapter 3

Related Work

Entity matching (EM) has received much attention since Dunn introduced *record linkage* in 1946 [1]. Today, it is still a challenge in data management. As a consequence, a lot of studies has been done on the topic. Since, the main topic for this thesis are on transfer learning, handcrafted features and transformer-based language models, the three topics will be the focus for the related work in the field of EM.

Few books have been published on this topic to provide an overview. Christen [1] covers the data matching process in depth and is a comprehensive source for EM. Newcombe et al. [19] developed Probabilistic methods and looked at EM from a statistical perspective. The blocking process, a sub-task in EM, has been studied by Papadakis et al. [10], Christen [1], Ebraheem et al. [20] and Konda et al. [4]. The goal of the studies are to reduce the computational complexity for blocking.

EM systems

The focus in EM has mainly been on developing matching algorithms. However, some believe that it is important to look at the whole pipeline in EM to advance in the field [4].

Konda et al. [4] did an extensive research and developed an EM system that covers the whole EM pipeline called Magellan. The system includes several blocking techniques and traditional machine learning methods for matching. Magellan has achieved state-of-the-art (SOTA) results on structured data. The disadvantage with Magellan is the weak performance on dirty data and text-heavy data.

In recent years, several studies have focused on deep learning methods. Mudgal et al. [14] developed DeepMatcher. The DeepMatcher system uses a RNN architecture for the matching task and compares different methods of attribute representations from word embeddings. The Hybrid model based on bidirectional RNN and decomposable attention achieved the overall strongest performance. Mudgal et al. concluded that DeepMatcher is competitive with Magellan

on structured data, but Magellan is outperformed by the deep learning models on textual and dirty data. The disadvantage for DeepMatcher is the long training time for DL models compared to traditional models.

Following DeepMatcher, Ebraheem et al. [20] presented DeepER that uses DL methods both for blocking and matching. DeepER are based upon uni- and bi-directional RNN in combination with LSTM. Also, DeepER outperforms Magellan on some of the datasets.

Transformer-based language models

Two studies have examined pre-trained transformer language models (LM) for EM. Li et al. [3] presented a novel EM system called Ditto. They define the EM problem as a sequence-pair classification task and experiments with four pre-trained LMs, BERT, RoBERTa, DistilBERT and XLNet. Although, they do not test the LMs in a low-resource setting, as will be tested in this thesis, the LMs achieve SOTA results with only half of the data. To improve the F1-score of Ditto and to reduce the size of required labeled data, they presents three optimization techniques; data augmentation, injecting domain knowledge and summarization of long strings.

Data augmentation are used to add training data by applying an operator (del, shuffle, swap) to a span of tokens, to the attributes or to the entries. The second optimization technique, inject domain knowledge into Ditto to emphasize the potentially important part of the input. This is based on the intuition of humans that match the data, which often look at key details in the records before making a decision of match/non-match. The injected domain knowledge is span types e.g. street number, product IDs, publisher etc or span normalization by rewriting spans in a string to be syntactically equal. The experiments showed a small improvement with a F1-score up to 0.74% on four of the dataset. To summarize long textual values, Ditto use TF-IDF to only remove stopwords an not relevant information in the entities. The results in F1-score for Ditto improved from 41% to over 93% for a specific text-heavy dataset.

The results for Ditto shows that it achieves SOTA results and outperforms DeepMacther with an average F1-score up to 31%.

The other study by Brunner and Stockinger [2] analyze the same four language models BERT, RoBERTa, DistilBERT and XLNet for the matching task. In comparison to Ditto, Brunner and Stockinger focus on dirty and textual data. The main findings for Brunner and Stockinger are that transformers improve the score for challenging data up to 35.9% in F1-score compared to classical deep learning models. The transformers can also be used out-of-the-box for EM and still improve the performance on small clean datasets.

In both Brunner and Stockinger and Li et al., RoBERTa achieved the overall best score of the four transformer-based language models evaluated.

Pre-trained language models, such as BERT has also been applied to a specialized EM problem, namely product matching in e-commerce. A study by Peeters et al. [21] showed that BERT are able to achieve a strong performance with a F1-score above 90% for unseen product data without fine-tuning it on product-specific data. This is promising results for language models in the domain of entity matching.

Transfer learning

So far only studies which look at model optimization has been mentioned. However several studies have also looked at other techniques in combination with the matching models. Thirumuruganathan et al. [18] examined transfer learning (TL) for EM. Similar to this thesis, transfer learning was used from a high resource setting to a low resource setting. However, as TL is used with transformer-based LM, Thirumuruganathan et al. used TL with traditional machine learning algorithms. The authors transferred both features and training data, and concluded that transfer learning for entity matching are feasible and effective.

A disadvantage with TL is that the source dataset and the target dataset may have different schema, which can influence the transfer of knowledge from the source to target. Thirumuruganathan et al. [18] solves this by encode the tuples into a standard feature space of dimension d. To do this they use distributed word representations such as FastText.

Another study for TL with EM has been conducted by Kasai et al. [15]. They developed deep learning models for EM through a novel combination of TL and active learning. Similar to Thirumuruganathan et al. [18] and this work, it was tested in a low resource setting.

Transfer learning was first used in a setting with a significant amount of labeled data. A transferable model are pre-trained on multiple datasets to a target setting where the transferable model are fine-tuned with active learning. Transfer learning followed by active learning with deep learning models achieved strong performance with limited data. Transfer learning in isolation resulted in an unreliable and unstable EM solution.

Zhao and He [22] use transfer learning with a pre-trained language model. They propose an end-to-end entity matching (EM) system called HI-EM. Unlike this thesis which pre-train on EM bechmark datasets, the model proposed by the aforementioned authors leverage pre-trained EM models from large-scale knowledge base (KB) data. This system was motivated by real-world application for enterprises that wants to match their customers. HI-EM suggests that pre-training a model makes it effective for EM and outperforms existing solutions.

While there has been several work on transfer learning with EM, there has not yet been made a publicly pre-trained LM specialized for EM.

Handcrafted features

Magellan has achieved strong performance on structured data with string similarity metrics. With the rise of NLP, word embeddings have become a beneficial technique for deep learning models. However, for values without any semantic meaning, such as numerical values, word embeddings do not provide an accurate result. Chen et al. [23] proposes a hybrid similarity model that combine word-embeddings with handpicked string similarity metrics for classical machine learning models. The string metrics are based on the attribute values. The hybrid method was evaluated on 3 different datasets with the models; XGBoost, random forest (RF) and KNN. The result shows that the proposed hybrid method achieved the highest score on 1/3 datasets for 2/3 models by up to 10% in F1-score.

The downside with this method is that it requires domain knowledge about the data to manually choose appropriate string metrics.

Although string metrics combined with word embeddings did not have the overall best performance, string metrics alone have given comparative results and perform well with limited data [4]. The strengths of string metrics have not yet been tested with pre-trained language models in a low resource setting. This will be analyzed in this thesis.

Other techniques

Active learning (AL) has become a popular technique for EM. Active learning is a machine learning technique were a user can interactively label a proposed query by the model. Meduri et al. [24], Doan et al. [25], Kasai et al. [15] have evaluated different AL strategies to utilize traditional machine learning algorithms and deep learning models for EM. The studies have achieved promising results for EM with limited labeled data.

Motivation

To summarize, the aforementioned traditional machine learning models and deep learning models have achieved state-of-the-art results in EM. However, there are some limitations to the models. Magellan, which are based on traditional machine learning models show weak performance on textual and dirty data. This has been (to a degree) solved by deep learning models such as Ditto and DeepMatcher, but the challenge with DL models is the need for a sufficient amount of data to achieve SOTA results. The labeling process to obtain a sufficient amount of training data can be time-consuming and expensive for industrial companies, and a domain expert is often needed due to domain specific names in the data. Transfer learning as used by [18] and [15] has given good result for EM in a low resource setting. Transfer learning are therefore used with transformer-based LMs to developed a model more specialized for EM that can be used in a setting with low resources. As aforementioned, also string metrics in combination with

pre-trained transformer-based language models will be tested in a low resource setting.

Chapter 4

Data

In entity matching (EM), several benchmark datasets have been generated by crawling HTML pages from various websites. These has been pre-processed and are used to evaluate different methods for EM. This chapter gives an overview of the public datasets used. First, the datasets are introduced. Then, the various characteristics for the data is discussed. Finally, the taxonomy of the data is presented.

4.1 The public datasets

The data used in this thesis are the Magellan datasets. They are obtained and downloaded as CSV-files from the DeepMatcher repository[1]. The data have been created by students at UW-Madison by crawling HTML sites from two websites from various domains such as software, citations and restaurants [26]. The data have then been pre-processed and blocked. Since the dataset are already blocked, this thesis does not need to focus on the blocking stage. The blocked data has been split into a train set, validation set and a test set with ratio 3:1:1. The fixed split on the datasets makes it easier to compare the results on the data across different studies who have used the same split.

The datasets presented in Table 4.1 are distributed into the five datasets; *record A*, *record B*, *train*, *validation* and *test*. Record A and record B contain the attribute names and values. Train, validation and test contain *ltable_id*, the id for a record from record A, *rtable_id* the id for the record from record B and *label* which takes the value 1 and 0 corresponding to match or non-match.

The 12 datasets used for the experimental study are summarized in Table 4.1. The data have varied sizes from 450 to 28,707, origins from different domains and have different amount of attributes.

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

Table 4.1: The 12 public datasets used for the experimental study.

Type	Dataset	Domain	Size	True matches	Attributes
Structured	BeerAdvo-RateBeer	beer	450	68	4
	iTunes-Amazon	music	539	132	8
	Fodors-Zagats	restaurant	946	110	6
	Walmart-Amazon	electronics	10,242	962	5
	Amazon-Google	software	11,460	1,167	3
	DBLP-ACM	citation	12,363	2,220	4
	DBLP-Scholar	citation	28,707	5,347	4
Textual	Abt-Buy	product	9,575	1,028	3
Dirty	iTunes-Amazon	music	539	132	8
	Walmart-Amazon	electronics	10,242	962	5
	DBLP-ACM	citation	12,363	2,220	4
	DBLP-Scholar	citation	28,707	5,347	4

4.2 Dataset characteristics

This section presents the data characteristics that can influence the performance of the classification.

4.2.1 The three datatypes

The aforementioned datasets have been categorized into *structured*, *textual* and *dirty* based on the structure of the attribute and the attribute values. These three types are described below.

Structured EM A dataset is classified as structured when: Record A and Record B follow the same schema with the attributes $A_1, A_2, .., A_n$. Every attribute value contains information associated with the attribute name and are cleaned values [27]. The attribute can contain text-based attributes such as product names, but the length is restricted. An example of this can be found in Table 4.2.

Textual EM The attribute values for Record A and Record B contain long textual values, such as product information or long descriptions. Table 4.3 shows textual data.

Dirty EM Record A and Record B follow the same schema. However, the attribute values do not necessarily respond to the corresponding attribute name. The dirty datasets have been made by modifying the corresponding structured datasets. Attributes values are randomly moved into another attribute. This is to simulate dirty datasets that can occur in the real world, with missing values and were the attributes values have been shuffled. An example of dirty attributes are shown in Table 4.4.

Table 4.2: Structured

Song_Name	Artist_Name	Released
Lips are Movin	Meghan Trainor	3:01,9-Jan-15

(a) Record A

Song_Name	Artist_Name	Released
Lips are Movin	Meghan Trainor	3:01,9-Jan-15

(b) Record B

Table 4.3: Textual

description
speck toughskin black case for 4th generation ipod nano nn4tsblk tough custom-fit protective case durable shock-absorbing material detachable rotating belt clip openings to all connections and controls black finish

(a) Record A

description
4g nano toughskin black nn4-ts-blk

(b) Record B

Table 4.4: Dirty

Title	Brand	Price
Macbook Pro 4th generation Apple		199.99

(a) Record A

Title	Brand	Price
MacBook Pro 4th generation	Apple	199.99

(b) Record B

4.2.2 Profiling dimensions

The structuredness and the textuality of the datasets are not always enough to understand the difficulties in some of the datasets for EM [27]. To understand the challenges, consider the case where the textual similarity for true matches is low, but the semantic similarity is close. For Magellan, which is heavily string based this can be hard to capture. This is because the string metrics typically look at the syntactic similarity. Therefore, when the records are only semantically close, the results of the metrics can be negatively affected, since the metrics do not capture that behaviour as seen in Table 4.5. Further, the result of matching strings can also differ since only some metrics have been developed to capture eg. OCR errors, misspellings and long textual values [13]. To better understand the aforementioned problem and the specific data difficulties which may affect the matching, Primpeli and Bizer [27] did an extensive profiling on some of the benchmark datasets and proposed several profiling dimension. The most important dimensions for this thesis are described below.

Textuality The textual values contain long sequences of words [27]. This can be long descriptions or product titles that may have been divided into multiple attributes or reside in one attribute.

Corner cases Corner cases describe the case where non-matching pairs have a high textual similarity, while the textual similarity for true matches is low [27].

Small sets Small datasets in this case respond to datasets with a size less than

1000. A challenge with small datasets is that it can lead to the models overfitting more easily.

Dense vs sparse data Sparsity is the amount of missing values in a datasets. It is calculated by counting the occurrence of missing values in the dataset. Missing values can be a challenge for some supervised machine learning algorithm, and alter their ability to make correctly predictions [27]. Dense data contain few missing values.

Table 4.5: Examples of corner cases and textual attribute values for two matches and a non-match. Even though it can be easier for a human eye to match these records, the Jaccard score of true matches is only 0.52 and 0.16.

Dataset	**Record A**	**Record B**	**Jaccard score**	**Match/no-match**
Amazon-Google	motu digital performer 5 digital audio software competitive upgrade (mac only)	motu digital performer dp5 software music production software	0.52	match
Amazon-Google	norton antivirus 2007	ca antivirus 2007	0.5	non-match
Abt-Buy	whirlpool duet wfw9200sq white front load washer wfw9200swh 4.0 cu . ft. capacity 6th sense technology quiet wash plus noise reduction built-in water heater add-a-garment feature sanitary cycle 4 temperature selections white finish	whirlpool 27 ' duet washer horiz axis wp	0.16	match

4.2.3 Imbalanced datasets

Matching two databases with $n = 1000$ records each, would lead to $n \times n = 1\ 000\ 000$ tuple comparisons. Since only one record from one database matches to exactly one in the other, the majority of the tuple pairs are clearly not matches. To reduce the amount of tuple pair comparisons and the complexity of the matching, blocking has been introduced as described in chapter 2. The aim of the blocking technique is to reduce the size of non-matches, but keep as many true matches as possible. The challenge with blocking is to not have a too strict blocking criteria where true matches also gets filtered out. A solution to this challenge is to have a less strict blocking criteria, where a larger size of non-matches is accepted in the candidate set.

This results in skewed datasets in entity matching as seen in Figure 4.1. When evaluating the models it is important to choose evaluation metrics that captures the imbalance of matches/non-matches.

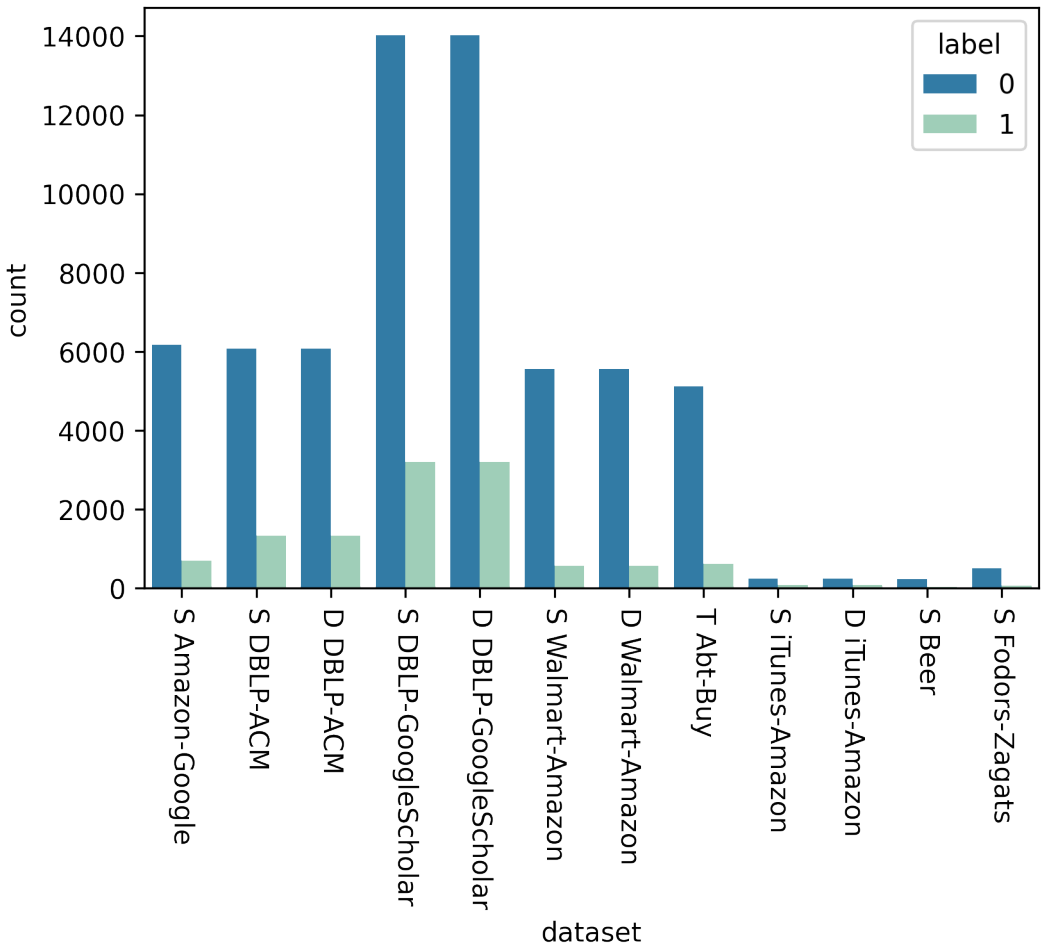


Figure 4.1: The distribution of true matches and non-matches for the train sets. The amount of true matches is much lower than the number of non-matches. S refers to structured, D refers to dirty and T to textual. The same patterns is present in the validation and test set.

4.3 Taxonomy

The twelve datasets can be divided into three different groups based on the aforementioned dimensions as presented in Table 4.6. In addition, three datasets are also categorized as challenging datasets. This is due to longer textual values, a larger amount of missing values and many corner cases.

Table 4.6: The dataset are divided into three different groups based on its characteristics. Three datasets are also categorized as challenging due to longer textual values, missing values and corner cases.

Group	**Dataset**
Small, dense data	Beer iTunes-Amazon Fodors-Zagats
Textual, few corner cases	DBLP-ACM DBLP-Scholar Walmart-Amazon (challenging)
Textual, many corner cases	Amazon-Google (challenging) Abt-Buy (challenging)

Chapter 5

Method

Whereas the previous chapters have focused on background and theory, this chapter will go through the four different models and how experiments were ran on them. First, a summary of the tools and libraries used followed by the experimental setup. Then the implementation of the four models and the evaluation metrics are discussed.

5.1 Tools and libraries

The Python programming language has been used to implement the experiments. The major frameworks used are:

Pandas Provides efficient data structures and data analysis tools in the form of DataFrames. [1]

scikit-learn Provide data analysis tools and several classification and clustering algorithms.[2]

PyTorch Open-source machine learning framework which provides tools and algorithms for deep learning. Commonly used in NLP and image recognition.[3]

py_stringmatching Provides string similarity and tokenizer tools.[4]

py_entitymatching Provides algorithms for supervised-learning based entity matching.[5]

Transformers NLP add-on for PyTorch by Huggingface. The library contains multiple NLP models, such as RoBERTa.[6]

[1] https://pandas.pydata.org/pandas-docs/stable/index.html
[2] https://scikit-learn.org/stable/
[3] https://pytorch.org/
[4] https://github.com/anhaidgroup/py_stringmatching
[5] https://github.com/anhaidgroup/py_entitymatching
[6] https://huggingface.co/

5.2 Experimental Setup

This section provides the experimental setup. The experiments were run 3 and 5 times to validate the results obtained.

5.2.1 Hardware

When training deep learning models one are often dependent on sufficient and efficient computational resources. The transformer-based models have been performed on the IDUN computing cluster at NTNU [5]. They provide a a high-availability compute resources with multiple GPUs and CPUs for research conducted at NTNU. The less crowded NVIDIA Tesla P100 GPU with 16 GB memory on an Intel Xeon CPU with 24 cores was used.

5.2.2 Dataset

The twelve datasets presented in chapter 4 have been used to evaluate the four models. The datasets origins from various domains and represents different types of data i.e. *structured*, *textual* and *dirty*. Since, the models are tested on different types of data the wide applicability of the models are tested.

The models were trained on varied sizes from the training datasets, in order to examine the performance of the models in a low resource setting.

5.2.3 Hyperparameters

The transformer models were trained with either 3 or 5 epochs. The number of epochs was based on the experiments conducted by Brunner and Stockinger [2] which showed that the transformer models mostly converges after 3-5 epochs. The default value was 3, but for the more challenging datasets it was increased to 5. However, from the experiments the models was not always able to learn anything with few samples and resulted in a F1-score of 0%.

The batch sizes used was 8, 16 and 32 due to the varied sizes of training samples. After a small empirically study a batch size of 8 was used up to 200 samples, 16 up to 1000 samples and 32 with sample sizes larger than 1000. For simplicity the batch sizes was not individually tuned to the different datasets.

The Adam optimizer was used with a linear learning rate of 3e-5 [3]. The other hyperparameters were left to their default value.

5.3 The four models

To answer the research questions stated in chapter 1, three experiments were conducted based on four different models. The models, with their names, is presented in the list below.

- **Magellan** Baseline model based on Magellan using Random Forest.
- **RoBERTa model** The model is based on the transformer-based language model RoBERTa published by Liu et al. [16].
- **Transfer learning model** This model combines the RoBERTa model with transfer learning by further pre-train the model on entity matching datasets.
- **Hybrid model** The RoBERTa model combined with handcrafted features.

The following subsections give an overview of each model.

5.3.1 Baseline model with Magellan

The *py_entitymatching* package provides an implementation of Magellan, and this has been adapted to create baseline models. The package offers several traditional machine learning algorithms, string similarity metrics and tokenizers to be used in the Magellan matching process. Magellan is used as the baseline since it has achieved state-of-the-art results for structured and semi-structured data in entity matching and are based on supervised traditional machine learning.

In the baseline models created, Random Forest was chosen as the classifier for Magellan. Random Forest was selected because it gave the best results. The hyperparameters were left to their default values.

Magellan was chosen as the baseline as it has achieved state-of-the-art results on structured datasets. Also, on these types of datasets Magellan still outperforms some DL models, such as the one provided by DeepMatcher and DeepER as mentioned in chapter 3. Magellan is easy to use as it provides tools and libraries for the whole EM pipeline from data pre-processing to the matching task. Further, Magellan is a popular baseline in EM and has been used as a baseline by Li et al. [3], Brunner and Stockinger [2], Mudgal et al. [14], Thirumuruganathan et al. [18] to mention a few.

Feature generation

Magellan models require features to be made explicitly in the input. The automatic feature generation provided by Magellan was used to obtain the features from the public datasets. As described in chapter 2, the various similarity measures use different tokenizers. The features used and the corresponding tokenizer are shown in Table 5.1.

In Magellan, the corresponding attributes from the two records are compared. This means that author are matched against author, title against title, and so on. This is problematic for the dirty datasets, which can contain missing values, because the aforementioned classifier and features do not work with missing values. The missing values were imputed by setting the attribute value to zero.

Table 5.1: The five feature generators and the corresponding tokenizer used in the Magellan-based baselines.

Type	**Feature**	**Tokenizer**
Sequence based	Levenshtein similarity	none
	Jaro-Winkler	none
Set-based	Dice	q-grams
Hybrid	Generalized Jaccard	q-gram
	Monge-Elkan	whitespace

5.3.2 RoBERTa

The RoBERTa language model has been thoroughly examined in the previous chapters, and it was stated that it achieved the overall best results of the transformer-based language models in experiments conducted by Brunner and Stockinger [2] and Li et al. [3]. Consequently, RoBERTa has been selected as the transformer classifier. The language models is present in the Python transformer library provided by Huggingface. The steps described in this section are based on Ditto developed by Li et al. [3].

RoBERTa has been pre-trained on a large corpus of English words using masked language modeling (MLM) as described in chapter 2 [16]. The RoBERTa model can then be fine-tuned with task-specific datasets. In this case, the language model is fine-tuned for the EM task using the labeled training data consisting of known true matches and non-matches.

Serializing the candidate pairs

The transformer-based language models require the input to be a tokenized sequence. Therefore, for the candidate pairs to be meaningful for the model, they are converted into tokens and embedded. Recall that each record pair consists of entities from record a and record b, so to preserve the information in the entities, two special tokens [COL] and [VAL] are added to each entity as presented in Ditto by Li et al. [3]. Each data entry e is serialized as follows:

$$e = \{(\text{attr}_i, \text{val}_i)\}_{1 \leq i \leq k}, \tag{5.1}$$

$$\text{serialize}(e) ::= [\text{COL}]\text{attr}_1[\text{VAL}]\text{val}_1 ... [\text{COL}]\text{attr}_k[\text{VAL}]\text{val}_k, \tag{5.2}$$

where [COL] indicates the start of an attribute name and [VAL] indicates the start of an attribute value. The candidate pairs e and e' are then serialized with [CLS] and [SEP] as described in chapter 2.

$$\text{serialize}(e, e') ::== [\text{COL}]\text{serialize}(e)[\text{SEP}]\text{serialize}(e')[\text{SEP}]. \tag{5.3}$$

Each token is added with a positional embedding and a segment embedding. The embeddings are serialized into one sequence and fed into the model as input. Figure 5.1 show how the entities are fed into the network.

Fine-tune RoBERTa for entity matching

Whereas the theory section only discussed RoBERTa architecture from a general point of view, the following will explain how RoBERTa was fine-tuned to fit the entity matching tasks as done by Li et al. [3] in Ditto.

On top of the pre-trained LM two new uninitialised layers are added. The two task-specific layers are a linear layer and a softmax output layer used for classification as shown in chapter 2 and presented by Li et al. [3].

The pre-trained RoBERTa model are fine-tuned with task-specific training data for EM. The RoBERTa model train on the training set for the data presented in chapter 4. The result is a pre-trained model fine-tuned for the EM task [3]. The output of the model is either a match or a non-match.

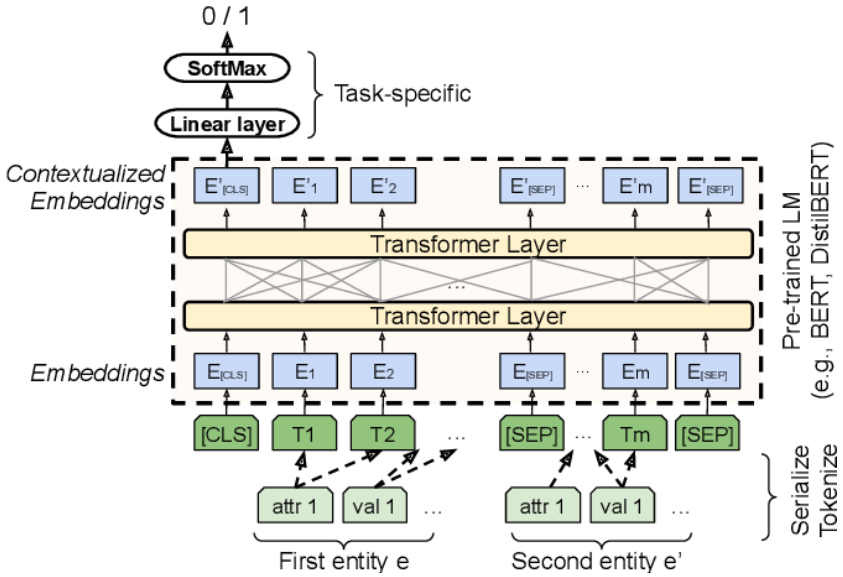


Figure 5.1: The model architecture used for entity matching with pre-trained transformer-based language models (LM) (e.g. BERT, RoBERTa). Source [3].

5.3.3 RoBERTa with transfer learning

Transfer learning (TL) has already been applied to RoBERTa by it being pre-trained on English text. The limitation of RoBERTa is that it need a sufficient amount of data to perform well. Two studies conducted by Thirumuruganathan et al. [18] and Kasai et al. [15] have used TL with deep learning models and traditional machine learning models. The use of TL showed performance improvement and a need for less training data. The third model, the Transfer learning model, utilizes the strengths of TL with RoBERTa, by further pre-train

it on EM datasets. The motivation behind the model is to develop a pre-trained transformer-based language model specialized for EM.

The development of the Transfer learning model was done through the following steps:

1. The 12 EM datasets presented in chapter 4, except for a *target* dataset, are concatenated into one big dataset, the *source* dataset. The EM datasets pooled together have already been pre-processed and are candidate datasets containing known true matches and non-matches. These *source* datasets are used to pre-train the RoBERTa model.

2. The RoBERTa model are trained on the source datasets. In each epoch, the model is trained on a subset of each dataset individually.

3. The pre-trained RoBERTa model are fine-tuned on a target dataset.

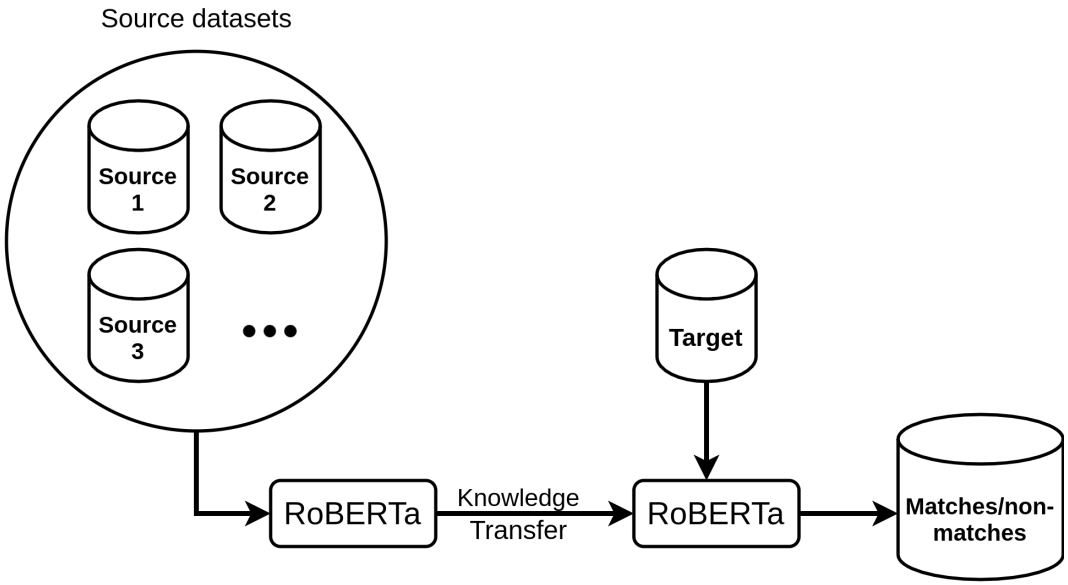


Figure 5.2: The pipeline for the Transfer learning model. First, N datasets are concatenated. Then the RoBERTa model train on the concatenated datasets, *source datasets*. The pre-trained RoBERTa model are then fine-tuned on a *target dataset*. The output is either a match or a non-match.

TL with different EM scenarios

The Transfer learning model developed are used for three different scenarios in EM based on the available training data in source and target.

Adequate, nothing In TL this is called *unsupervised domain adaption*, and can be looked at as an extreme situation. The training set, D_T^L, for the target domain do not have any training data available, $D_T^L = 0$. The training dataset, D_S^L for source contains an adequate amount of labeled data.

Adequate, limited The target datasets has a limited amount of training data, while the training set for source has an adequate amount. This scenario is called *supervised domain adaption* in TL.

Adequate, adequate Both D_T^L and D_S^L contain an adequate amount of training data.

5.3.4 RoBERTa with handcrafted features - hybrid

The fourth model named the Hybrid model combines the RoBERTa model (without transfer learning on the EM datasets) with handcrafted features. Magellan has shown strong performance on structured data compared to deep learning models [14]. Further, Magellan which uses traditionally machine learning methods need less data [4]. The idea behind this model is to evaluate if a transformer-based LM in combination with string metrics is able to improve the performance for EM in a low resource setting.

Feature generation for the hybrid model

The same tokenizers and five string metrics as presented in Table 5.1 are also used for this model to generate features. In comparison to Magellan, which compares attribute to attribute, this model treated the entire tuple as a single text string, and therefore ignored the attribute boundaries. This is mostly due to the dirty datasets where the attribute values are not in their correct cells, but instead swapped into other attributes.

The features where implemented through the *py_stringmatching* package. The similarity features were concatenated with the embeddings from RoBERTa, and injected into the network in the classification layer.

5.4 Evaluation of the models

Evaluation metrics are used to evaluate the performance of the different models. A challenge in EM, is that the datasets are often imbalanced, meaning that the distribution of true matches are often smaller compared to the size of non-matches. In order to evaluate the matching models, it is important to use evaluation metrics which captures this behaviour [1]. Natural choices for evaluation metrics are therefore precision, recall and F1-score. These are described below.

In EM, predictions can be divided into the categories (also known as a *confusion matrix*) seen in Figure 5.3.

		Predicted classes	
Actual Matches	Matches	Matches	Non-matches
		True Positives (true matches)	False Negatives (false non-matches)
	Non-matches	False Positives (false matches)	True Negatives (true non-matches)

Figure 5.3: Confusion matrix illustrating the four outcomes of a classification algorithm for matching. The goal of the matcher is to maximize the *true matches* and minimize *false positives* and *false negatives*. Source: [1].

5.4.1 Precision

Precision is the amount of correctly predicted true matches out of all the predicted record pairs. It measure the exactness of the classifier. Precision is calculated as

$$\text{precision} = \frac{tp}{tp + fp}, \tag{5.4}$$

where tp is true positives and fp is false positives. Precision does not include true negatives. Due to this, precision is robust against class imbalance and is a good metric for entity matching.

5.4.2 Recall

Recall is the amount of correctly predicted matches from the whole set of actual true matches. It measures the classifiers completeness and is computed as

$$\text{recall} = \frac{tp}{tp + fn}. \tag{5.5}$$

Similar to precision, recall does not include true negatives and therefore does not suffer from imbalanced data [1].

5.4.3 F1-score

In entity matching, when evaluating the matching one often ideally wish to have a high precision and a high recall. As an example, out of all the predicted records pairs, you want to be sure that a record pair is correctly predicted (precision), and captures as many true matches as possible (recall). The F1-score manages this trade-of.

F1-score combines precision and recall, and is the harmonic mean between the two. It is calculated as

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{5.6}$$

A high F1-score indicates both a high precision and a high recall. F1-score is the main metric used to evaluate the aforementioned four models.

Chapter 6

Results

This chapter presents the results from the conducted experiments, described in chapter 5, on the different datasets presented in chapter 4. The presentation of the results is organized in groups, because there is a natural group in the data and it makes it easier to compare across the groups. In Section 6.1, the similar datasets are presented first, followed by the challenging datasets, and the small datasets presented at last.

Then, section 6.2, contains a table expressing the standard deviation and the average F1-score for each model, dataset and sample size. The last section present the run time for each model on every dataset.

6.1 Result for the datasets

For each group/subsection below, the results are presented in the form of plots of each model on each dataset for an increasing sample size. The F1-score displayed is the average of the 3 or 5 runs for each model.

6.1.1 Textual data, few corner cases

Figure 6.1 shows the average F1-score for DBLP-ACM and DBLP-Scholar. Both datasets contain textual data with few corner cases. Another similarity between the two, is that both origin from DBLP (computer science bibliography).

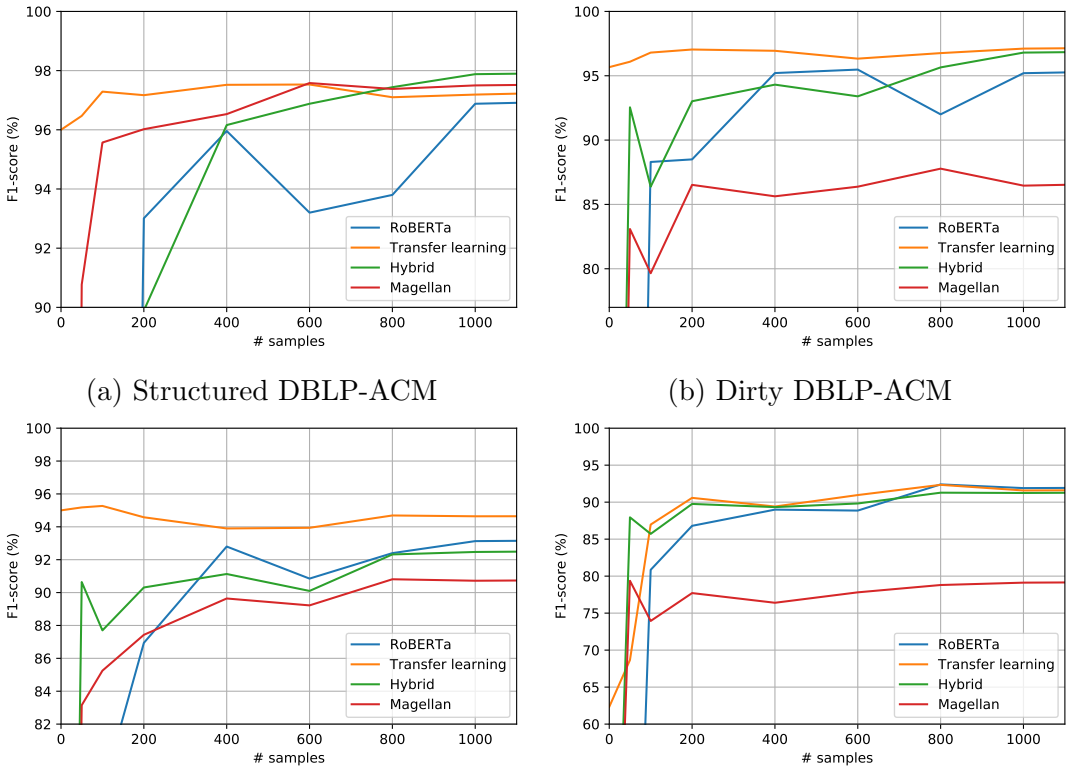
Magellan, the baseline, gives competitive results to Hybrid and RoBERTa on the structured datasets. On DBLP-ACM, it achieves almost 98% in F1-score, and on DBLP-Scholar slightly over 90%. On the dirty datasets, Magellan is outperformed by the three transformer-based LMs by approximately 10-15%.

The Transfer learning model obtains a high starting point on all datasets. With zero training examples it achieves up to 96% in F1-score. In comparison, RoBERTa and the Hybrid model need 600-1000 samples before obtaining the same score. However, the advantage with to the Transfer learning model decreases with in-

creasing training examples, as it is caught up by RoBERTa and Hybrid.

In the plots for RoBERTa in Figure 6.1a, Figure 6.1b and Figure 6.1c, the score fluctuates with different sample sizes. For instance, the score for Structured DBLP-ACM drops from 96% with 400 samples to approximately 93% with 600 samples. The same effect is also present for Dirty DBLP-ACM and Structured DBLP-Scholar.

In all the cases, the pattern is that the transformer based models converges to the same F1-score.



(a) Structured DBLP-ACM

(b) Dirty DBLP-ACM

(c) Structured DBLP-Scholar

(d) Dirty DBLP-Scholar

Figure 6.1: The reported average F1-score for the 3 or 5 runs for RoBERTa, Hybrid, Transfer learning and Magellan on DBLP-ACM and DBLP-Scholar. The main observations from the plots are the high starting points fro Transfer learning and the weak performance of Magellan on the two dirty datasets.

6.1.2 Textual data, many corner cases (challenging datasets)

Figure 6.2 shows the results with respect to F1-score for Walmart-Amazon, Amazon-Google and Abt-Buy. The datasets presented contain textual values and/or many corner cases, and are for this reason considered challenging. The Walmart-Amazon dataset is also challenging due to textuality as discussed in chapter 4. As shown in Figure 6.2, the transformer-based LMs need more than 1000 samples

before the F1-score converges. The plots for the datasets with less than 1000 samples are found in Appendix A.

In Figure 6.2, it is evident that Magellan has a weak performance on the challenging datasets. The difference between Magellan and the transformer-based LMs is on average 20%. On Dirty Walmart-Amazon the difference is up to 40% in F1-score. Moreover, Magellan need less data than Hybrid and RoBERTa before it converges. For the majority of sample sizes less than 1000, Magellan outperforms both Hybrid and RoBERTa.
The Transfer learning model achieves strong performance on all datasets, and achieves a score above 58% for every dataset even with zero samples. After 1000 samples the Transfer learning model are caught up by Hybrid and RoBERTa. However, for Structured Walmart-Amazon, RoBERTa and the Hybrid model need more than 5000 samples to catch up with the Transfer learning model.

The difference in score for RoBERTa and the Hybrid model is small. Both follows each other after given sufficient data of around 1000 samples. The benefit of the Hybrid model in comparison to RoBERTa is that it have a tendency to need less samples to get to the score plateau. The Hybrid model achieves a score above 70% with approximately 400 samples, while RoBERTa needs more than 1000 samples to achieve the same on Abt-Buy.

6.1.3 Small datasets

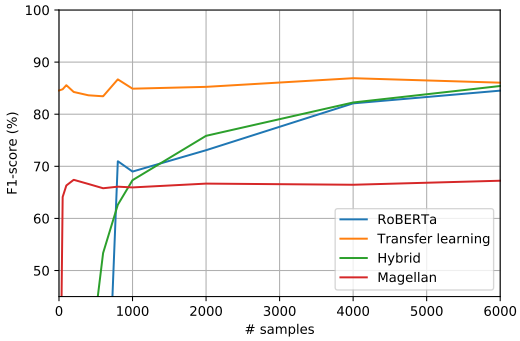
Figure 6.3 shows the results with respect to the average F1-score for Magellan, RoBERTa, Hybrid and Transfer learning for the small datasets.

From Figure 6.3, observation shows that Magellan is outperformed on 4/4 datasets. The margin between Magellan and the three transformer-based LMs is up to an average of 40 % on Dirty iTunes-Amazon. On the structured datasets, the margin is much smaller.

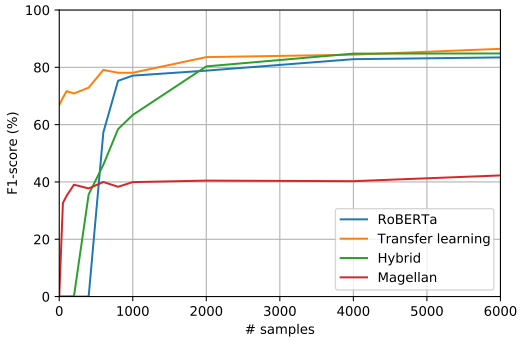
The Transfer learning model achieves a high score ($> 80\%$) with zero samples on all datasets. The plots for Structured Beer, as shown in Figure 6.3c show that the F1-score drops from approximately 94% to 85% with 100 samples. For the other datasets, the score for the Transfer learning model is increasing monotonically. Compared to the other models, Transfer learning is the only model that does not achieve a score of 100% for Fodors-Zagats.

Overall, the RoBERTa model and the Hybrid model need more samples than Transfer learning and Magellan to achieve the same score as the two aforementioned models. For iTunes-Amazon, RoBERTa and the Hybrid model have a similar performance.

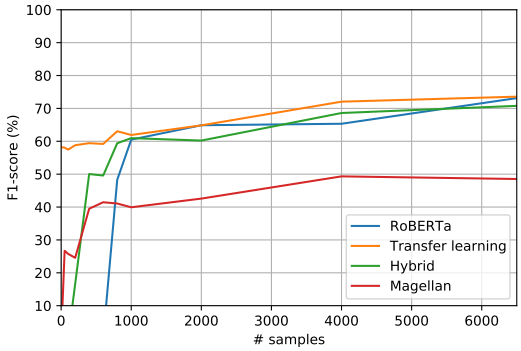
To summarize the performance for the four models on small datasets, one can observe that Transfer learning achieves the highest performance on low resources,



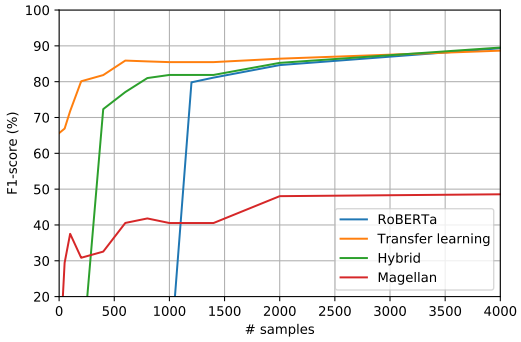
(a) Structured Walmart-Amazon



(b) Dirty Walmart-Amazon



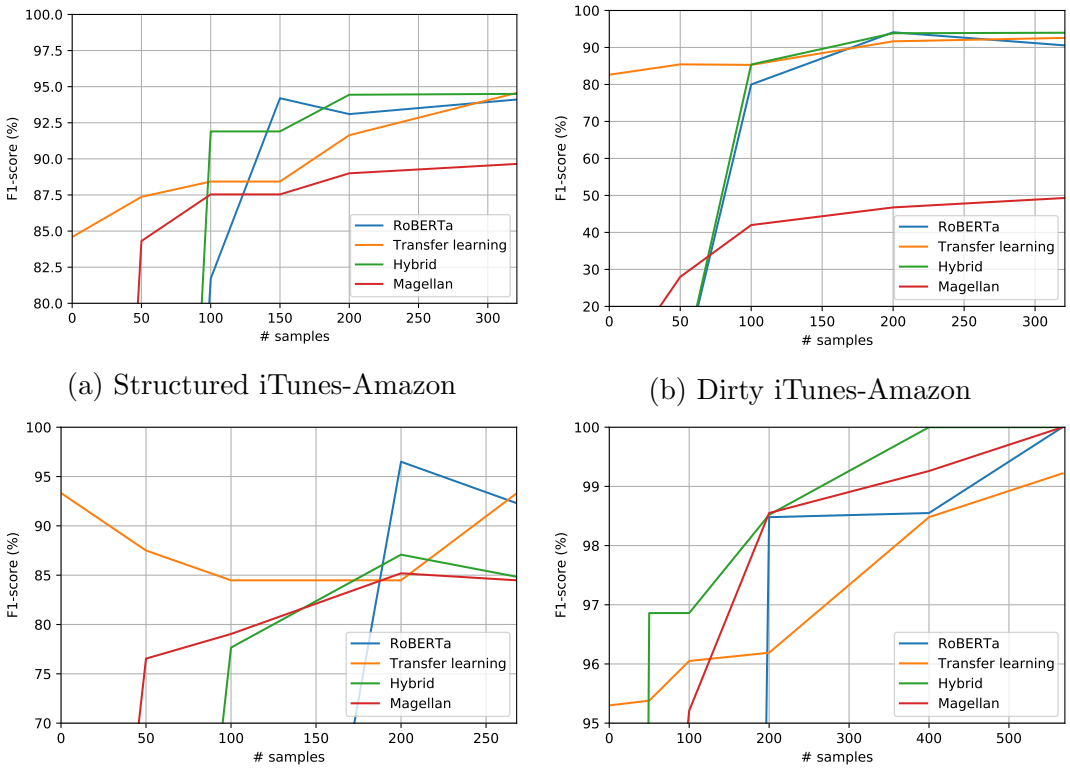
(c) Structured Amazon-Google



(d) Textual Abt-Buy

Figure 6.2: The average F1-score for the 3 or 5 runs achieved by the four models (RoBERTa, Transfer learning, Hybrid and Magellan) on the challenging datasets (textual values and corner cases). RoBERTa and Hybrid need a sufficient amount of data before they converges. Magellan shows weak performance on all four datasets, but outperform RoBERTa and Hybrid with limited data.

however after 100 samples RoBERTa and Hybrid have the same or a better performance than the Transfer learning model. Magellan is the model with the weakest performance.



(a) Structured iTunes-Amazon

(b) Dirty iTunes-Amazon

(c) Structured Beer

(d) Structured Fodors-Zagats

Figure 6.3: The average F1-score of the 3 or 5 runs for the four models for the small public datasets (all with less than 1000 training samples).

6.2 Variance

Table 6.1 and Table 6.2 show the average F1-score on the 3 or 5 runs, and the standard deviation for the 12 datasets.

Table 6.1 presents the variance and the average F1-score for the small datasets. As mentioned, the main advantage of the Transfer learning model is the high score with few samples, with zero and 50 samples where it achieves a strong performance above 80% in F1-score. The variance for the Transfer learning model is overall less than the variance for RoBERTa and Hybrid. However, for Dirty iTunes-Amazon the variance is up to almost 7%. A trend for RoBERTa and Hybrid, it that the variance decreases with more samples. However, for the Transfer learning model the variance is constant or increasing, except for Dirty iTunes-Amazon.

The variance for the RoBERTa model is decreasing with an increase in samples. As an example, the variance for Structured iTunes-Amazon is 2.44% for 100 samples, and 0.11% for all (321) samples. The same trend is also applicable to the Hybrid model. For RoBERTa and the Hybrid model two values stand out with 100 samples, 36.73% for Hybrid on Structured Beer and 42.43% for RoBERTa on Structured Fodors-Zagats.

Table 6.2 shows the average F1-score and the standard deviation for the remaining datasets.

In Table 6.2 the variance for the Hybrid model is up to 55% with 50 samples for all dataset. In comparison, the variance for Transfer leaning is only below 2-3%, except for Dirty DBLP-Scholar where it is 14%.

Overall, the variance decreases drastically with more training data. For Amazon-Google the variance is as high as 24.4% with 400 samples, whereas with all the available training data ($\sim$ 6000) the variance is only 0.14%. In comparison, the variance for Transfer learning is nearly constant. Another observation is the significant gap for the challenging datasets. As an example, for Amazon-Google the gap is 11% between 1000 samples and all samples when only considering the F1-score. The improvement for the DBLP-ACM and DBLP-Scholar is smaller. The difference for Structured DBLP-ACM is significantly small, 2%, with all samples ($\sim$ 7000) used, compared to 1000 samples used.

6.3 Training time

This section presents the training time for the transformer-based LM for each sample size. The training time could have been affected by the load on IDUN, so it is important to keep this is mind. The training time for zero samples for the Transfer learning model represent the time usage to pre-train the model on EM datasets.

From Table 6.3 it is observed that the Transfer learning model overall has the shortest training time. However, for Structured Fodors-Zagats and Dirty DBLP-Scholar it is notably high. Additionally, Transfer learning model has the highest increase in time from 1000 sample to all samples.

When comparing Hybrid and RoBERTa, the Hybrid model are 20-40 seconds slower than RoBERTa. The difference in time between the two models increases with larger sample sizes.

Table 6.1: The average F1-score of the 3-5 runs and the standard deviation for RoBERTa, Transfer learning and the Hybrid model for the small datasets. The score reported for Magellan is the highest score achieved for each sample. The highest score for each sample in each dataset is highlighted.

Dataset	**Algorithm**	**Sample size**					
		0	**50**	**100**	**200**	**400**	**All**
Structured iTunes-Amazon	RoBERTa	0	0	81.73 (±2.44)	93.10 (±2.46)	-	94.21 (±0.11)
	TL	**84.59** (±1.67)	**88.25** (±1.04)	**88.44** (±1.67)	91.25 (±1.31)	-	**95.15** (±1.62)
	Hybrid	0	0	87.69 (±5.99)	**94.45** (±1.57)	-	92.89 (±2.66)
	Magellan	0	85.71	91.23	89.65	-	89.66
Dirty iTunes-Amazon	RoBERTa	0	0	79.52 (±0.67)	**94.11** (±0.89)	-	90.57 (±0)
	Transfer learning	**82.63** (±6.96)	**84.53** (±6.24)	**87.93** (±5.44)	90.70 (±0.74)	-	91.88(±0.50)
	Hybrid	0	0	85.38 (±7.32)	93.85 (±2.23)	-	**93.84** (±0.69)
	Magellan	0	29.17	45.28	52.83	-	50
Structured Beer	RoBERTa	0	0	0	84.51 (±3.97)	-	87.06 (±2.35)
	Transfer learning	**88.44** (±1.33)	**87.56** (±2.23)	86.62 (±1.25)	**86.72** (±3.3)	-	**87.73** (±4.49)
	Hybrid	0	0	51.77 (±36.73)	83.05 (±5.70)	-	82.27 (±1.84)
	Magellan	0	81.48	**86.67**	86.67	-	84.84 (±0)
Structured Fodors-Zagats	RoBERTa	0	0	30 (±42.43)	97.09 (±1.93)	98.91 (±1.88)	**100** (±0)
	Transfer learning	**95.30** (±0.10)	**95.38** (±0.10)	96.05 (±1.15)	96.19 (±2.24)	98.48 (±1.06)	99.22 (±1.10)
	Hybrid	0	0	96.86 (±1.15)	98.48 (±2.14)	**100** (±0)	**100** (±0)
	Magellan	0	90.0	**97.78**	**100**	**100**	**100**

Table 6.2: The average F1-score of the 3-5 runs and the standard deviation for RoBERTa, Transfer learning and the Hybrid model. The score reported for Magellan is the highest score achieved for each sample. The highest score for each sample in each dataset is highlighted.

Dataset	Algorithm	Sample size								
		0	50	100	200	400	600	800	1000	All
Textual Abt-Buy	RoBERTa	0	0	0	0	0	0	0	0.045 (±0.05)	88.62 (±1.84)
	Transfer learning	**63.78** (±4.9)	**66.29** (±2.84)	**71.40** (±1.70)	**74.53** (±0.44)	**82.48** (±3.91)	**84.06** (±4.11)	**84.73** (±3.32)	**83.71** (±3.09)	**89.77** (±0.74)
	Hybrid	0	17.44 (±30.21)	17.74 (±25.52)	18.24 (±31.60)	72.95 (±2.19)	77.10 (±2.64)	81.01 (±1.36)	81.66 (±0.39)	89.71 (±0.65)
	Magellan	0	36.01	42.51	39.30	45.03	43.77	46.93	43.99	51.47
Structured Walmart-Amazon	RoBERTa	0	0	0	0	0	0	70.98 (±2.36)	68.99 (±4.44)	84.90 (±1.97)
	Transfer learning	**84.57** (±1.01)	**84.50** (±1.36)	**85.33** (±1.01)	**84.27** (±1.21)	**83.96** (±0.56)	**83.52** (±1.56)	**86.41** (±0.74)	**85.07** (±1.53)	**86.16** (±1.14)
	Hybrid	0	0	0	0	16.89 (±22.02)	53.31 (±8.37)	59.23 (±8.09)	64.51 (±2.92)	84.89 (±0.81)
	Magellan	0	68.18	67.35	68.46	67.56	67.11	67.06	67.51	67.92
Dirty Walmart-Amazon	RoBERTa	0	0	0	0	0	0	57.29 (±6.14)	75.23 (±2.02)	85.15 (±1.31)
	Transfer learning	**68.47** (±2.02)	**70.25** (±1.29)	**73.36** (±1.74)	**73.11** (±1.64)	**77.57** (±2.77)	**79.71** (±1.33)	**79.68** (±1.43)	**79.21** (±2.78)	**85.18** (±0.84)
	Hybrid	0	0	0.02 (±0.04)	0.17 (±0.32)	20.21 (±17.33)	47.11 (±12.06)	58.06 (±6.54)	65.16 (±10.53)	84.98 (±1.03)
	Magellan	0	38.67	45.61	40.49	45.25	43.21	42.66	41.02	42.60
Structured Amazon-Google	RoBERTa	0	0	0	0	15.97 (±22.58)	9.39 (±13.29)	50.65 (±7.13)	60.54 (±2.7)	70.07 (±0.53)
	TL	**58.03** (±0.733)	**58.37** (±0.58)	**58.32** (±0.49)	**59.32** (±0.22)	**60.57** (±1.45)	**62.73** (±2.7)	**62.88** (±0.96)	**64.02** (±1.45)	**73.55** (±2.02)
	Hybrid	0	0	0	4.14 (±7.16)	33.39 (±24.4)	33.77 (±7.08)	47.36 (±12.09)	49.67 (±11.59)	71.41 (±0.14)
	Magellan	0	43.28	37.18	42.98	42.86	39.89	40.58	44.81	52.08
Dirty DBLP-ACM	RoBERTa	0	0	0	88.26 (±1.47)	88.05 (±3.57)	94.53 (±1.57)	93.51 (±2.36)	92.48 (±1.15)	**99.16** (±0.06)
	TL		**95.67** (±0.43)	**96.09** (±0.58)	**96.74** (±0.31)	**96.95** (±0.35)	**96.77** (±0.33)	**97.00** (±0.47)	**97.12** (±0.40)	98.95 (±0.14)
	Hybrid	0	46.29 (±46.29)	90.32 (±2.03)	92.13 (±1.27)	95.43 (±1.12)	94.98 (±1.58)	96.06 (±0.41)	96.96 (±0.16)	98.90 (±0.09)
	Magellan	0	86.02	84.16	86.87	88.52	87.64	88.40	87.05	90.77
Structured DBLP-ACM	RoBERTa	0	0	0	93.05 (±2.05)	95.96 (±1.63)	93.20 (±1.32)	93.85 (±1.09)	96.88 (±0.31)	98.57 (±0.28)
	Transfer learning	**96.39** (±0.33)	**96.49** (±0.07)	**97.19** (±0.11)	97.22 (±0.07)	**97.49** (±0.07)	97.18 (±0.31)	97.07 (±0.26)	97.42 (±039)	**99.02** (±0.17)
	Hybrid	0	43.60 (±50.65)	67.65 (±45.18)	69.39 (±46.29)	95.74 (0.62)	95.83 (±1.33)	97.50 (±0.68)	**97.96** (±0.23)	98.87 (±0.11)
	Magellan	0	92.09	96.80	**97.97**	97.38	**98.19**	**97.62**	97.62	98.54
Structured DBLP-Scholar	RoBERTa	0	0	87.59 (±0.36)	86.94 (±3.10)	92.80 (±0.47)	90.85 (±2.07)	92.40 (±0.67)	92.80 (±0.69)	94.49 (±0.76)
	Transfer learning	**95.00** (±0.29)	**95.18** (±0.19)	**95.27** (±0.16)	**94.58** (±0.35)	**94.11** (±0.44)	**93.94** (±0.96)	**94.69** (±0.20)	**94.64** (±0.56)	95.13 (±0.26)
	Hybrid	0	30.21 (±52.33)	87.70 (±2.58)	90.31 (±0.75)	91.13 (±1.17)	90.10 (±3.22)	92.32 (±0)	92.47 (±0.02)	**95.57** (±0.35)
	Magellan	0	87.61	85.93	89.55	90.14	89.40	91.02	91.43	93.26
Dirty DBLP-Scholar	RoBERTa	0	0	80.85 (±1.90)	86.80 (±1.74)	**91.52** (±1.75)	88.86 (±1.12)	**92.42** (±1.07)	**91.89** (±1.47)	94.95 (±0.33)
	Transfer learning	**62.33** (±1.85)	78.66 (±14.40)	**86.97** (±4.42)	**90.47** (±0.86)	89.41 (±2.14)	89.67 (±1.33)	92.34 (±0.28)	91.58 (±0.30)	93.32 (±0.25)
	Hybrid	0	32.18 (±55.74)	85.72 (±3.05)	88.76 (±1.80)	89.33 (±0.54)	**89.92** (±025)	91.29 (±0.05)	91.24 (±0.86)	**95.47** (±0.12)
	Magellan	0	**82.05**	79.25	78.29	77.56	78.95	79.43	79.77	83.23

Table 6.3: The training time (seconds) for each sample size for the models; RoBERTa, Transfer learning and Hybrid.

Dataset	**Algorithm**	**Time/sample size [s]**								
		0	50	100	200	400	600	800	1000	All
Structured DBLP-Scholar	RoBERTa	0	6	7	17	31	45	88	97	386
	Transfer learning	3461	2	5	10	22	32	40	53	681
	Hybrid	0	25	27	36	51	65	83	97	1215
Structured Amazon-Google	RoBERTa	0	5	7	10	20	33	45	54	178
	Transfer learning	3980	1	3	7	13	20	27	33	233
	Hybrid	0	24	28	38	59	51	65	76	310
Structured DBLP-ACM	RoBERTa	0	0	10	19	40	50	67	81	378
	Transfer learning	3479	2	5	12	26	39	52	66	380
	Hybrid	0	30	33	42	64	85	104	124	670
Structured Walmart-Amazon	RoBERTa	0	6	9	15	30	47	65	56	254
	Transfer learning	4101	3	6	11	23	34	46	58	271
	Hybrid	0	27	31	44	68	79	102	122	439
Structured Beer	RoBERTa	0	7	10	21					12
	Transfer learning	4382	2	4	8					8
	Hybrid	0	37	45	65					54
Structured iTunes-Amazon	RoBERTa	0	9	144	28					23
	Transfer learning	4228	6	11	22					27
	Hybrid	0	28	33	48					74
Structured Fodors-Zagats	RoBERTa	0	6	9	18	36				21
	Transfer learning	4303	65	69	78	93				88
	Hybrid	0	34	31	44	61				63
Textual Abt-Buy	RoBERTa	0	8	10	21	41	62	82	90	377
	Transfer learning	3718	4	9	18	37	55	73	91	383
	Hybrid	0	27	35	53	79	112	142	173	675
Dirty DBLP-Scholar	RoBERTa	0	6	8	18	30	33	47	55	722
	Transfer learning	2455	91	93	97	106	113	122	129	1245
	Hybrid	0	26	30	40	55	70	87	104	1209
Dirty DBLP-ACM	RoBERTa	0	6	10	20	41	63	87	91	389
	Transfer learning	3476	3	7	16	34	50	67	84	390
	Hybrid	0	25	36	42	62	83	104	125	687
Dirty Walmart-Amazon	RoBERTa	0	5	11	17	34	46	62	59	271
	Transfer learning	3840	2	5	11	23	35	46	58	70
	Hybrid	0	23	27	34	52	68	88	103	449
Dirty iTunes-Amazon	RoBERTa	0	10	14	29					24
	Transfer learning	4330	6	12	24					28
	Hybrid	0	31	39	53					66

Chapter 7

Discussion

This chapter starts by going through the training of the models, with a specific focus on the tuning of the hyperparameters. After this, each of the four different models is discussed independently. At the end, the usage of the models for a real-world application is discussed.

7.1 Tuning the hyperparameters

The transformer model contains millions of hyperparameters that can potentially be analyzed and tuned. Due to the time limit of this thesis, little time has been used to tune all the hyperparameters. Instead, two hyperparameters have been focused on: *batch size* and *epochs*.

Ideally, more time should have been spent on tuning the hyperparameters. This could have resolved the issues where the models were unable to give any results. Then again, developing and training models on such small sample sizes is unlikely to happen in a real-world setting. The research questions are mostly concerned about the relative performance of the models. Spending much time improving the results of each model is unlikely to greatly affect the relative performance of the models.

Batch size

As mentioned, for the small datasets and with few resources, the model gave 0% as output F1-score. The batch size was therefore lowered to 8 at most. The advantage with a small batch size is that the model converges quickly. The disadvantage is that the quickly convergence comes at a cost of noise in the data, which can lead to a less accurate performance.

Epochs

Based on the empirical study conducted by [2], which showed that transformer models RoBERTa, DistilBERT, BERT and XLNet achieved peak performance for the datasets Textual Abt-Buy, Dirty iTunes-Amazon and Dirty Walmart-Amazon

after 3-5 epochs, number of epochs was set to 3 for the "easier" dataset and 5 to the more challenging datasets. An increased epochs size could perhaps have helped the model achieve results in the cases when in achieved a 0% in F1-score.

7.2 Baseline - Magellan

Good or bad performance of a model is relative, and in this case it also differs for each dataset. It is therefore helpful to have a baseline model to use for comparison. As an example, everything above a F1-score of 97.5% is considered good performance on Structured DBLP-ACM, and everything above a F1-score of 50% on Structured Amazon-Google is considered good performance because it outperforms the baseline.

7.2.1 Weak performance on challenging and dirty datasets

The baseline, Magellan, was outperformed by the transformer-based models on almost all datasets. For the structured datasets, it performed particularly bad for the Walmart-Amazon and Amazon-Google datasets. A reason for the weak performance for these two structured dataset may be due to the content in the dataset. Both Structured Walmart-Amazon and Amazon-Google contain product data, which may include long textual descriptions. The product data also contain many corner cases which means the attribute values are semantically similar, but have a similarity distance that is large, as shown in Table 7.1. For instance, for Amazon-Google, a true match only got a Jaccard score of 0.52, and was classified as a false negative.

Table 7.1 shows an example from the textual dataset Abt-Buy. The same arguments for Structured Walmart-Amazon and Amazon-Google can be applied to the textual data. The records in Abt-Buy include long textual blobs which is hard for Magellan to capture as a true match only got a Jaccard score of 0.16. One can hypothesize that Magellan has difficulties with summarizing long textual strings [27].

The weak performance on the dirty datasets may be due to a lot of missing values and because the attribute values has been moved around. Recall that Magellan compare attribute to attribute. Magellan is very sensitive to noise like this in the data.

7.2.2 Strong performance with limited data

The advantage of Magellan is that it needs few examples to achieve a good score. For most of the experiments, Magellan outperforms the Hybrid and RoBERTa model for a limited amount of samples.

Table 7.1: Examples of a true matches where the similarity between the attributes values are low.

Dataset	**Record A**	**Record B**	**Jaccard score**
Amazon-Google	motu digital performer 5 digital audio software competitive upgrade (mac only)	motu digital performer dp5 software music production software	0.52
Abt-Buy	whirlpool duet wfw9200sq white front load washer wfw9200swh 4.0 cu . ft. capacity 6th sense technology quiet wash plus noise reduction built-in water heater add-a-garment feature sanitary cycle 4 temperature selections white finish	whirlpool 27 ' duet washer horiz axis wp	0.16

7.3 The RoBERTa model

The RoBERTa model outperforms the baseline with a good margin on the majority of the datasets, especially on the dirty and textual datasets, when the amount of training samples get high. Because the model performs well relative to the baseline on the dirty datasets, it seems more robust against noise such as shuffling of attribute values. One possible reason for the better performance on dirty and textual data may be because RoBERTa treats the input strings as one, and do not only compare attribute to attribute as Magellan. Due to this, and the attention mechanisms, RoBERTa can easier see the dependence between words and what words it should focus on in the sentence.

For all the models and datasets, RoBERTa never achieves the best performance when given enough training samples. It also gets the most amount of 0% in F1-scores, indicating that the model has problems with limited data.

However, it is important to note that RoBERTa is outperformed by the baseline with limited data available. The difference is for some datasets up to 90%. Therefore, it is safe to conclude that the baseline is still preferable until RoBERTa is given sufficient data.

7.3.1 Unstable with few samples

The RoBERTa model has the most cases when the performance drops with more samples. It also has a high variance from 2-42 % when given limited data. The main reason for this may be due to overfitting. Another reason can be bad batches. The n training samples are picked from random every time, and it

is therefore possible to get a particularly poorly skewed training batch. When combined with the fact that this model has problems with understanding the data with few samples, this can lead to these counter-intuitive in performance. Unlike the other transformer models, the RoBERTa model has no additional matching mechanism. This can make the model more sensitive. Since the model has a high variance, is unreliable and unstable with few samples, it is unusable in a low resource setting.

7.3.2 Data hungry for challenging datasets

Interestingly, for the challenging datasets, as shown in Figure 6.2, the RoBERTa model shows a tendency to be more data hungry. The model need almost 1000 samples before it starts to learn something from the data to achieve a F1-score above 60%. Additionally, the model does not converge unless given a sufficient amount of training samples.

The RoBERTa model achieves a F1-score above 90% for all datasets, except for the challenging datasets and Structured Beer as shown in Figure 6.1, Figure 6.2 and Figure 6.3, when given all available data. As mentioned in chapter 4, the challenging datasets are the most text-heavy datasets. A potential reason are that long textual values can be hard to capture for language models. This is because it can be challenging for a language model to understand what it needs to pay attention to in the matching process [3], when the input strings are too long, which leads to it needing more data.

7.3.3 Faster runtime

When looking at the training times in Table 6.3, it is evident that RoBERTa has lower training time compared to the to Hybrid. This is because the model does not need to generate features.

Low training time and generally good performance are two criteria for a good baseline model. The RoBERTa model contains this and can therefore be well suited as a baseline model compared to Magellan for these EM datasets.

7.4 The Transfer learning model

The third model is the RoBERTa model with transfer learning (TL). This model achieves strong performance for all sample sizes, and does very well with zero training samples. With such good results for the Transfer learning model, it would seem like the obvious first choice of the four models. This section will discuss the results of the TL-based model.

7.4.1 Smaller advantage with larger sample size

Aforementioned, the Transfer learning model achieves a significantly higher training score than the other models when there is 0 or few sample. For Structured DBLP-Scholar it achieves 96.39% in F1-score for 0 samples. This immediately gives merit to the transfer learning approach. It confirms that the model is able to learn traits and patterns relevant in the entity matching task from training on other datasets. As expected from this, the TL model outperforms all the other models when given 0 samples.

From the result, it is safe to say that the Transfer learning model plateaus fast, which means that the improvement of the performance faster slows down. There can be multiple reason for this, one may be that as more training data are utilized, the difference between the source and target gets larger. The Hybrid and RoBERTa models do eventually catch up with the the TL model. It is clear that the transfer learning gives the model a head start, and that this advantage gets smaller with larger sample sizes.

7.4.2 Stable performance

For all the datasets except for Textual Abt-Buy and Dirty DBLP-Scholar, the performance gain from 0 training samples to maximum training samples is fairly low for the TL models. Consequently, the variance with sample size remains low. For that two datasets where there is a strong gain from 0 samples to maximum samples, the gain is likely caused by specific properties in the target dataset. This is further discussed in section 7.4.4.

From the variance, it is clear that the TL model is more stable than the RoBERTa and Hybrid. The additional TL feature gives the model more legs to stand on when making predictions. Stability is important when applying models in a production settings as it makes it easier to minimize the probability for worst case performance.

However, for zero and very few training samples observations of the models shows high variance, as seen in Table 6.1 and Table 6.2. The variance are at most 2.84%, except for Dirty iTunes-Amazon where it is 6.24%. To build a model with reliable results that can be used in an entity matching setting, a certain amount of training examples may be required.

7.4.3 Longer pre-training

From the results, as shown in Table 6.3, it is evident that the Transfer learning model has the overall longest runtime. The longer training time is naturally attributed to the fact that the Transfer learning model has to run through the other datasets first, which takes time. However, the pre-training of a general transformer-based LM only need to be performed once. Then, the pre-trained model can be fine-tuned on several downstream tasks.

The Transfer learning model seems to have a faster runtime for the fine-tuning task than RoBERTa. Since, it is already pre-trained on related dataset, it appears that the Transfer learning model generalizes faster. However, the training time for the Transfer learning model on Structured Fodors-Zagats and Dirty DBLP-Scholar are on the other hand 60-90 seconds slower. Recall, that the experiments were run on IDUN, and one explanation for the longer run time for the two aforementioned datasets may be due to the load on the cluster.

It is also worth noting that these long training times for pre-training RoBERTa with transfer learning occurred when using NVIDIA P100, which is more computation power than most desktop PCs have access to. Training time using consumer GPUs is likely to be even longer. Extensive computational power can be hard and expensive to obtain.

For each target dataset, the transfer learning process starts from scratch when learning from the other datasets. This leads to lots of duplicated learning, which is very inefficient. The training time could be decreased significantly by caching the results of running a set dataset through transfer learning, and consequently reuse that.

7.4.4 Limitations of the Transfer learning model

For the Transfer learning model the knowledge was transferred directly from source datasets to the target dataset. This is not always ideal, as it can suffer from an imbalanced datasets or from data specific properties.

Instance weights

As mentioned in section 2.5, different instance weights can be added to the source dataset and the target dataset. This is because the transfer learning can suffer from imbalanced datasets, since the source datasets is often much larger than the target dataset. The source can in worst case *swamp* the target dataset [18]. In this thesis, such instance weights have not been added, so it can not be excluded that the source datasets may have swamped the smaller target datasets.

Data specific properties

As mentioned, the Transfer learning model has a lower starting point compared to maximum achieved F1-score for Dirty DBLP-Scholar and Textual Abt-Buy. This can be due to data specific properties. Even though both DBLP-ACM and DBLP-Scholar contain bibliographic data, the attribute values for *author* differs. DBLP-ACM contains the full name of the author, while DBLP-Scholar contains the first initials and last name. An example of this is given in Table 7.2. Since Dirty DBLP-Scholar has a low starting point, it seems like the Transfer learning model do not capture well initials in the start as is does not occur in any of the other datasets. The Transfer learning model in this thesis has not taken such

idiosyncratic properties into account and it is a suggestion for further work.

The same can be applied to Textual Abt-Buy. The datasets contain long textual values, which the model may have a hard time summarizing in the beginning.

Table 7.2: Examples of data-specific properties. Record A and Record B both include author, but Record A contains full name while Record B contains the first initials.

Record A	Record B
zengping tian , hongjun lu , yanlei diao , songting chen	h lu , k tan , s dao

7.5 Hybrid model

The Hybrid model combines the RoBERTa model with handcrafted features. The idea behind the Hybrid model is to combine the benefits of state-of-the-art string metric based methods (Magellan) with transformers (RoBERTa).

7.5.1 The impact of handcrafted features for RoBERTa with limited data

For all datasets except Structured DBLP-ACM, the Hybrid model outperforms the RoBERTa model in the small sample range, less than 200 samples. The models (Hybrid and RoBERTa) converge, and for 9 of the datasets this convergence occurs before they together converge with the Transfer learning model. The head start the Hybrid model gets over the RoBERTa gives stature to the idea of utilizing handcrafted features for improving the performance over pure RoBERTa when there is little samples. For the larger sample sizes, the benefits of using Hybrid over RoBERTa are small from a performance perspective, only on average 1-4% better for some of the datasets.

Despite being able to outperform the RoBERTa model when sample sizes are low, the model is still outperformed by Magellan with few samples, in all datasets. It seems that the handcrafted features was able to pull the RoBERTa model in the right direction, but that the pull was not strong enough. Recall, that only five features were injected into the RoBERTa model. This may have been too few features to have a significant impact in RoBERTa. A solution for further work is to insert more features and see if this gives a larger impact.

The training time for the Hybrid model is consistently higher than that for the RoBERTa model, which is natural given that the Hybrid model contains an additional feature for the RoBERTa model. At most, for the Structured Walmart-Amazon dataset, the training time for the Hybrid model is double that of the RoBERTa model and the Transfer learning model.

7.5.2 Very unstable model in a low resource setting

From Table 6.2, it is evident that the Hybrid model has a very high variance in its F1-score performance within each sample size. This is particularly evident in the Structured Amazon-Google, Structured DBLP-ACM and Dirty DBLP-ACM datasets. This high variance can be attributed to the fact the model has a hard time learning from limited data, which is also a pattern in the pure RoBERTa model. In many cases where the Hybrid model has a high variance, the RoBERTa model is not able to learn at all. This future strengthens the idea that the handcrafted features pull the RoBERTa model in the right direction, but that the pull is not strong enough.

From Figure 6.1, Figure 6.2 and Figure 6.3 it apparent that the sample-to-sample variance for the Hybrid model is less than that for the pure RoBERTa model. Intuitively, the added handcrafted features smooths the performance.

7.5.3 Best of both worlds?

Research question 3 seeks to understand the impact of handcrafted features in combination with transformers in a low resource setting. From the discussion in this section, it is conspicuous that the addition of handcrafted features is able to improve the performance of the the transformer RoBERTa when the sample size is low, but not enough to beat the baseline.

One potential reason why the Hybrid model gets beat by the baseline in the low resource setting is that even though the Hybrid model uses handcrafted features, it is still mainly a RoBERTa language model. The RoBERTa language model relies on initial parameters that must be learned by the model, and the fact that handcrafted features are being fed into the model does not change this. Therefore, the model still needs some samples before it is able to adjust its parameters and reflect the nature of the data.

7.6 Application

Whereas the previous sections have focused on the individual models and their performance, this section will assess the models from a more practical point of view, focusing on how these models can be used in software applications and in real-world settings for industrial companies such as Cognite.

In the industry, solid datasets for supervised learning can be hard, time-consuming and expensive to acquire. Depending on the context, strong domain knowledge can be required to do this. This is the case for the Cognite data platform. Cognite's platform gathers data from the industry, and much of the data going into their system is proprietary, raw and unlabeled. Manually labeling the data that goes into the platform can be very expensive for Cognite. Models that perform well with low samples, and systems that make it easier for them to label data,

could be of great benefit to them.

The TL model proves to work best in the low resource scenario, but it is expensive both computational and time wise to pre-train. Moreover, the RoBERTa and Hybrid model are very poor in the same scenario, but the performance catch up when given enough samples. When given enough samples, the RoBERTa and Hybrid models are easier to maintain, debug and utilize, all important aspects of software development. To utilize the best of both worlds, the following setup is interesting:

Today, there are no pre-trained language model especially developed for the EM task. TL could therefore be used to train language models on large amount of related datasets. The pre-trained model could then be used out-of-the-box for EM. In Cognite's case, they might have some labeled data that have been hard to acquire which they want to reuse as much as possible. Utilizing this data more than once through transfer learning could be very beneficial to them.

One could also use the Transfer learning model to generate potential matches with a *high* accuracy for a domain expert. The expert could then label these matches, which should have a high match rate. Through this process, one should be able to more quickly get a sample rich dataset which can be used in the other parts of a software suite. By utilizing even more datasets to train the TL model, it could be possible to create an even stronger general TL model which could be used in a variety of entity matching problems. To summarize, the TL model is suited to work in an unsupervised or semi-supervised entity matching context to generate potential training data/correct samples.

None of the models developed are ideal, and should therefore not be used in a settings where false positives or false negatives are detrimental. In settings where mistakes are allowed, the matches should be presented in a way that does not undermine the trust in the application.

Chapter 8

Conclusion

This thesis has tested and evaluated four different models on 12 different datasets for the entity matching (EM) task. The first model is the baseline model based on Magellan, a traditional machine learning EM system which have received state-of-the-art results in EM. The baseline was compared to a transformer-based language model, RoBERTa. With limited data, the RoBERTa model was outperformed by the baseline on some of the datasets. To boost the performance of RoBERTa, this thesis examined transfer learning with RoBERTa by pre-training it on entity matching datasets. In a low resource setting this model outperformed the other by a large margin with a F1-score up to 30%. The last model combine handcrafted features with RoBERTa. This model have shown to be unstable with limited data, and have a high variance, 22-55% with less than 400 samples. Overall, this model was also beaten by the baseline.

RoBERTa with transfer learning works best in a low-resource setting. For a real-world application, where it can be time-consuming to label data, transfer learning can be a good solution in the beginning to label a sufficient amount of training data. Further, it can also be used with a transformer-based language model to train on specific data to developed a specialized model for industrial companies.

Research question 1:

How does the performance of transformer models compare to traditional machine learning based methods in a low-resource setting?

The RoBERTa model outperforms the baseline on 10 out of 12 datasets when given enough samples. However, the baseline, Magellan, achieves the best performance of the two with very few samples. The downside with Magellan is the sensitivity to noise in the data as discussed in chapter 7. The RoBERTa showed to be more robust and has therefore a wider applicability. The RoBERTa model is for this reason recommended if compared to Magellan. The trade-off to label more data can be worth it, due to the stronger performance.

As discussed in chapter 7 and shown in chapter 6, RoBERTa is never the best model, and it should therefore be considered to use the Transfer learning model or the Hybrid model.

Research question 2:

How is the performance on entity matching affected when combining transformers with transfer learning on entity matching datasets in a low-resource setting?

Transfer learning boosts the performance of RoBERTa with limited data. With zero samples, the Transfer learning model are able to achieve a F1-score of 58-96%. Compared to the other models, this model converges faster, but is caught up by RoBERTa and Hybrid when given sufficient data. However, for all sample sizes, the Transfer learning model is the most stable model and achieves the highest score on 7 out of 12 datasets by a margin of 1-2%. The model has shown to be feasible for the EM task.

Transfer learning model is a good solution for applications that need a high starting point, and can be used with other models to label data in a low resource setting. It is recommended to use a transformer-based language models pre-trained on related datasets for EM.

As covered in chapter 2, the Transfer learning model did perform as expected as it achieved a relative high starting point and converged fast compared to RoBERTa and Hybrid. However, the most surprising observation was that it was easily caught up by the two other models for several datasets meaning the advantage of TL decreases. However, our experience from the experiments show that the Transfer learning model do outperform Hybrid and RoBERTa, and should therefore be considered in high resource settings.

Research question 3:

What is the impact of handmade features in combination with transformers in a low-resource setting?

The impact of handmade features injected into RoBERTa are highest with a sample size of 100-800. As discussed in chapter 7 the RoBERTa model did not always initialize and achieved a F1-score of 0%. In comparison, the Hybrid model, achieves a F1 score 70-80% above RoBERTa. However, the model shows tendencies to be unstable with variances up to 55% with few samples (< 400).

Also for Hybrid, the advantage of the model decreases with larger sample size. When given sufficient training data, mostly above 1000 samples, the Hybrid model achieves a score 1-4% above the RoBERTa model. The efficiency of the model are also affected. The Hybrid model uses 20-30 seconds longer than the RoBERTa model, mostly due to the generation of features.

The Hybrid model did perform slightly worse than expected in the start of this thesis with limited data. Our thoughts was that string metrics would achieve more reliable results with limited data. However, this was not the case as seen in Table 6.2 and Table 6.1.

8.1 Further Work

This section provides suggestions for further work.

- **Data interpretation for transfer learning** The transferred knowledge in this thesis was not adjusted or transformed. For further work is suggested to use instance weights or a feature transformation strategy.
- **Appropriate similarity metrics** The same five string metrics were used for every attributes in the datasets. Some metrics are developed to handle numerical values, misspellings in name etc. A solution for further work is to apply relevant string metrics for the attribute depending on the format.
- **Hyperparameter tuning** As described in both chapter 5 and chapter 7 only the batch size and epochs were somewhat tuned. Still, the model was not always initialized and got 0 in F1-score with few labels, especially for the RoBERTa model. This can lead to the model being unstable and unusable. A suggestion for further work is to have an extensive empirical study to find the most optimal parameters for the pre-trained language model.
- **Combine transfer learning with other techniques** The transfer learning model achieved great success on few samples. For other applications that rely on good initialization and a high starting point, transfer learning can be a good solution. With the results gotten, transfer learning could be used with i.e. active learning. Another suggestion is to combine transfer learning and handcrafted features.
- **Public available pre-trained EM model** This thesis has shown that transformer-based language models pre-trained on EM datasets are effective for EM. A suggestion for further work is to pre-train a language model on more EM datasets that can be used out-of-the-box to be fine-tuned on downstream tasks.

Bibliography

[1] Peter Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer-Verlag, Berlin Heidelberg, 2012. ISBN 978-3-642-31163-5. doi: 10.1007/978-3-642-31164-2. URL https://www.springer.com/gp/book/9783642311635.

[2] Ursin Brunner and Kurt Stockinger. Entity Matching with Transformer Architectures - A Step Forward in Data Integration, 2020. URL https://openproceedings.org/2020/conf/edbt/paper_205.pdf. Version Number: 1 type: dataset.

[3] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. Deep Entity Matching with Pre-Trained Language Models. *Proceedings of the VLDB Endowment*, 14(1):50–60, September 2020. ISSN 2150-8097. doi: 10.14778/3421424.3421431. URL http://arxiv.org/abs/2004.00584. arXiv: 2004.00584.

[4] Pradap Konda, Jeff Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, Vijay Raghavendra, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, and Haojun Zhang. Magellan: toward building entity matching management systems. *Proceedings of the VLDB Endowment*, 9(12):1197–1208, August 2016. ISSN 21508097. doi: 10.14778/2994509.2994535. URL http://dl.acm.org/citation.cfm?doid=2994509.2994535.

[5] Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. *arXiv:1912.05848 [cs]*, December 2020. URL http://arxiv.org/abs/1912.05848. arXiv: 1912.05848.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017. URL http://arxiv.org/abs/1706.03762. arXiv: 1706.03762.

[7] Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2. February 2008.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. URL `http://arxiv.org/abs/1810.04805`. arXiv: 1810.04805.

[9] S. J. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010. ISSN 1558-2191. doi: 10.1109/TKDE.2009.191.

[10] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. Blocking and Filtering Techniques for Entity Resolution: A Survey. May 2019. URL `https://arxiv.org/abs/1905.06167v3`.

[11] Arjit Jain, Sunita Sarawagi, and Prithviraj Sen. Deep Indexed Active Learning for Matching Heterogeneous Entity Representations. *arXiv:2104.03986 [cs, stat]*, April 2021. URL `http://arxiv.org/abs/2104.03986`. arXiv: 2104.03986.

[12] Nils Barlaug and Jon Atle Gulla. Neural Networks for Entity Matching. *arXiv:2010.11075 [cs]*, October 2020. URL `http://arxiv.org/abs/2010.11075`. arXiv: 2010.11075.

[13] AnHai Doan, Alon Halevy, and Zachary Ives. 4 - String Matching. In AnHai Doan, Alon Halevy, and Zachary Ives, editors, *Principles of Data Integration*, pages 95–119. Morgan Kaufmann, Boston, 2012. ISBN 978-0-12-416044-6. doi: 10.1016/B978-0-12-416044-6.00004-1. URL `https://www.sciencedirect.com/science/article/pii/B9780124160446000041`.

[14] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data - SIGMOD '18*, pages 19–34, Houston, TX, USA, 2018. ACM Press. ISBN 978-1-4503-4703-7. doi: 10.1145/3183713.3196926. URL `http://dl.acm.org/citation.cfm?doid=3183713.3196926`.

[15] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. Low-resource Deep Entity Resolution with Transfer and Active Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5851–5861, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1586. URL `https://www.aclweb.org/anthology/P19-1586`.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa:

A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*, July 2019. URL http://arxiv.org/abs/1907.11692. arXiv: 1907.11692.

[17] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109(1):43–76, January 2021. ISSN 1558-2256. doi: 10.1109/JPROC.2020.3004555. Conference Name: Proceedings of the IEEE.

[18] Saravanan Thirumuruganathan, Shameem A. Puthiya Parambath, Mourad Ouzzani, Nan Tang, and Shafiq Joty. Reuse and Adaptation for Entity Resolution through Transfer Learning. *arXiv:1809.11084 [cs, stat]*, September 2018. URL http://arxiv.org/abs/1809.11084. arXiv: 1809.11084.

[19] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic Linkage of Vital Records. *Science*, 130(3381):954, October 1959. doi: 10.1126/science.130.3381.954. URL http://science.sciencemag.org/content/130/3381/954.abstract.

[20] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. DeepER – Deep Entity Resolution. *arXiv:1710.00597 [cs]*, November 2019. doi: 10.14778/3236187.3236198. URL http://arxiv.org/abs/1710.00597. arXiv: 1710.00597.

[21] Ralph Peeters, Christian Bizer, and Goran Glavaš. Intermediate Training of BERT for Product Matching. page 5, 2020.

[22] Chen Zhao and Yeye He. 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, 2019. ACM Press. ISBN 978-1-4503-6674-8. doi: 10.1145/3308558.3313578. URL http://dl.acm.org/citation.cfm?doid=3308558.3313578.

[23] Xiao Chen, Gabriel Campero Durand, Roman Zoun, David Broneske, Yang Li, and Gunter Saake. *The Best of Both Worlds: Combining Hand-Tuned and Word-Embedding-Based Similarity Measures for Entity Resolution*. Gesellschaft für Informatik, Bonn, 2019. ISBN 978-3-88579-683-1. doi: 10.18420/btw2019-14. URL http://dl.gi.de/handle/20.500.12116/21698. Accepted: 2019-04-11T07:21:16Z ISSN: 1617-5468.

[24] Venkata Vamsikrishna Meduri, Lucian Popa, Prithviraj Sen, and Mohamed Sarwat. 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, June 2020. Association for Computing Machinery. ISBN 978-1-4503-6735-6. doi: 10.1145/3318464.3380597. URL https://doi.org/10.1145/3318464.3380597.

[25] AnHai Doan, G. C. Paul Suganthan, Haojun Zhang, Adel Ardalan, Jeffrey Ballard, Sanjib Das, Yash Govind, Pradap Konda, Han Li, Sidharth Mudgal, and Erik Paulson. Human-in-the-Loop Challenges for Entity Matching: A Midterm Report. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics - HILDA'17*, pages 1–6, Chicago, IL, USA, 2017. ACM Press. ISBN 978-1-4503-5029-7. doi: 10.1145/3077257.3077268. URL `http://dl.acm.org/citation.cfm?doid=3077257.3077268`.

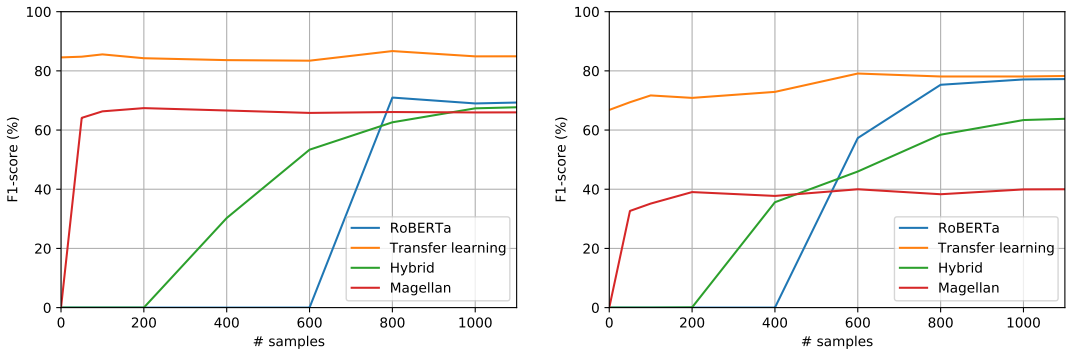
[26] anhaidgroup. anhaidgroup/deepmatcher, December 2020. URL `https://github.com/anhaidgroup/deepmatcher`. original-date: 2017-12-01T19:01:11Z.

[27] Anna Primpeli and Christian Bizer. Profiling Entity Matching Benchmark Tasks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3101–3108, Virtual Event Ireland, October 2020. ACM. ISBN 978-1-4503-6859-9. doi: 10.1145/3340531.3412781. URL `https://dl.acm.org/doi/10.1145/3340531.3412781`.

Appendix A

Graphs

This appendix includes the graphs for the challenging datasets with samples less than 1000. The plots for the datasets show the average F1-score for RoBERTa, Transfer learning model, Hybrid and Magellan.



(a) Structured (b) Dirty

Figure A.1: The average F1-score for RoBERTa, Transfer learning, Hybrid and Magellan up to 1000 samples.

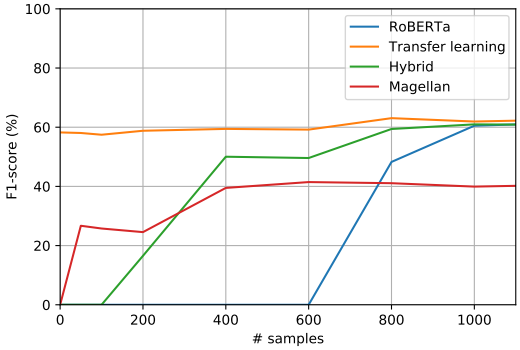


Figure A.2: The average F1-score for RoBERTa, Transfer learning, Hybrid and Magellan up to 1000 samples for Structured Amazon-Google.

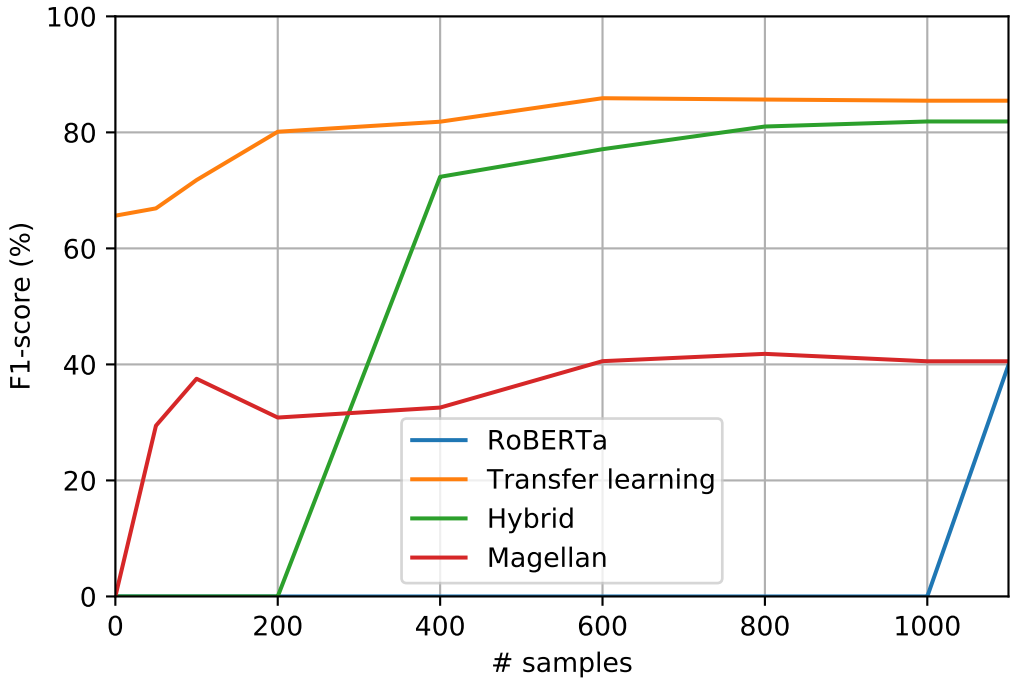


Figure A.3: The average F1-score for RoBERTa, Transfer learning, Hybrid and Magellan up to 1000 samples for Textual Abt-Buy.

Appendix B

Tables

This appendix includes the tables for the plots presented in chapter 6.

Table B.1: Dirty iTunes-Amazon

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	82.63	0	0
50	0	85.43	0	27.98
100	79.99	85.29	85.39	41.99
200	94.11	91.66	93.84	46.76
321	90.56	92.59	93.98	49.31

Table B.2: Dirty DBLP-ACM

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	95.67	0	0
50	0	96.09	92.55	83.09
100	88.3	96.8	86.38	79.65
200	88.5	97.04	93.02	86.52
400	95.21	96.94	94.31	85.63
600	95.48	96.33	93.40	86.38
800	92.0	96.76	95.65	87.78
1000	95.2	97.11	96.80	86.46
7417	99	98.77	98.87	90.59

Table B.3: Dirty DBLP-Scholar

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	62.33	0	0
50	0	68.66	87.95	79.37
100	80.85	86.97	85.71	73.94
200	86.8	90.58	89.76	77.71
400	88.99	89.41	89.33	76.4
600	88.86	90.96	89.82	77.81
800	92.4	92.34	91.28	78.80
1000	91.9	91.58	91.24	79.12
17223	95.6	95.28	95.47	83.21

Table B.4: Dirty Walmart-Amazon

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	66.8	0	0
50	0	69.4	0	32.67
100	0	71.7	0	35.15
200	0	70.87	0.08	39.04
400	0	72.9	35.61	37.74
600	57.3	79.1	45.99	39.99
800	75.3	78.1	58.43	38.31
1000	77.1	78.1	63.38	39.94
2000	78.84	83.56	80.32	40.47
4000	82.84	84.4	84.78	40.27
6144	83.5	86.6	84.84	42.42

Table B.5: Structured Beer

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	88.44	0	0
50	0	87.44	0	76.54
100	0	86.62	77.66	79.04
150	82.35	86.62	77.66	79.04
200	84.51	86.72	84.32	85.19
268	90.32	91.06	82.40	84.48

Table B.6: Structured Amazon-Google

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	58.24	0	0
50	0	58.05	0	26.7
100	0	57.46	0	25.77
200	0	58.8	16.54	24.55
400	0	59.43	50.03	39.49
600	0	59.19	49.59	41.46
800	48.27	63.06	59.40	41.08
1000	60.5	61.92	60.96	39.93
2000	64.88	64.84	60.22	42.57
4000	65.33	72.05	68.6	49.33
6874	74.3	73.80	71.1	48.409

Table B.7: Strucutred DBLP-ACM

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	96.0	0	0
50	0	96.47	87.19	90.77
100	0	97.29	88.78	95.57
200	93.01	97.17	89.89	96.02
400	95.96	97.52	96.16	96.53
600	93.2	97.53	96.88	97.58
800	93.8	97.1	97.44	97.38
1000	96.88	97.19	97.88	97.5
7417	98.9	99.21	98.87	98.46

Table B.8: Structured Fodors-Zagats

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	95.30	0	0
50	0	95.38	96.86	88.14
100	0	96.05	96.86	95.21
200	98.48	96.19	98.52	98.55
400	98.55	98.48	100	99.26
567	100	99.22	100	100

Table B.9: Structured DBLP-Scholar

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	95.0	0	0
50	0	95.18	90.64	83.15
100	77.92	95.27	87.7	85.25
200	86.94	94.58	90.31	87.43
400	92.80	93.9	91.13	89.64
600	90.85	93.94	90.1	89.221
800	92.4	94.69	92.32	90.81
1000	93.13	94.64	92.47	90.72
17223	96	95.12	95.45	93.04

Table B.10: Structured iTunes-Amazon

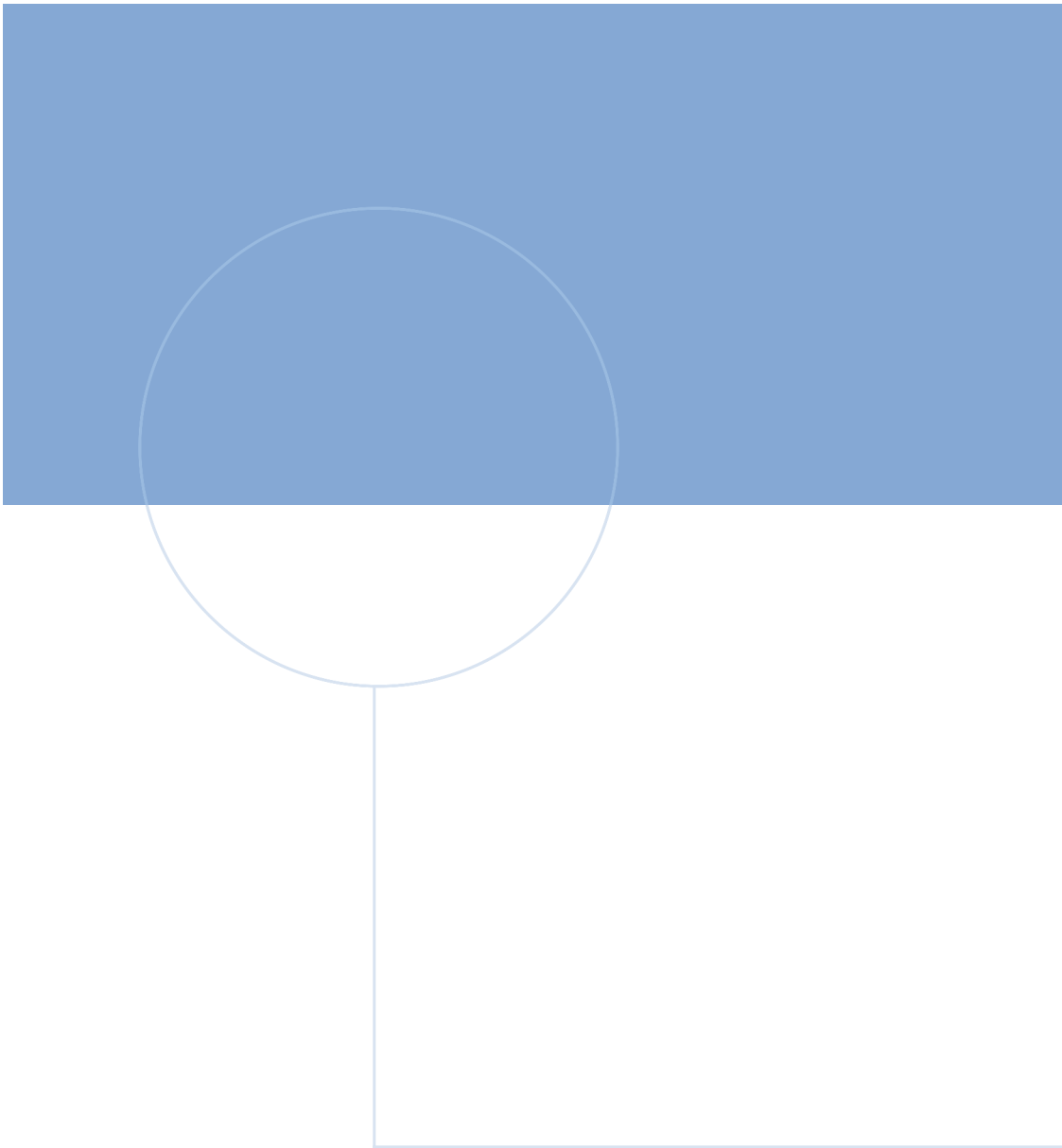
training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	84.59	0	0
50	0	87.37	0	84.31
100	81.71	88.43	91.90	87.54
150	94.2	88.43	91.90	87.54
200	93.1	91.64	94.44	89.0
321	94.11	94.57	94.5	89.65

Table B.11: Structured Walmart-Amazon

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	84.55	0	0
50	0	84.79	0	64.12
100	0	85.57	0	66.32
200	0	84.27	0	67.41
400	0	83.63	30.27	66.61
600	0	83.44	53.36	65.79
800	70.98	86.69	62.62	66.09
1000	68.99	84.90	67.34	65.95
2000	73.09	85.25	75.85	66.68
4000	82.08	86.91	82.25	66.46
6144	84.7	85.99	85.64	67.29

Table B.12: Textual Abt-Buy

training_examples	f1_score	f1_score_tl	f1_score_feat	f1_score_magellan
0	0	65.64	0	0
50	0	66.90	0	29.44
100	0	71.78	0	37.53
200	0	80.11	0	30.84
400	0	81.84	72.34	32.56
600	0	85.89	77.1	40.56
800	0	85.66	81.01	41.82
1000	0	85.46	81.88	40.54
1200	79.82	85.46	81.88	40.54
1400	81.13	85.46	81.88	40.54
2000	84.63	86.42	85.25	48.04
4000	89.33	88.66	89.53	48.56
5747	88.62	90.2	89.96	51.25



NTNU
Kunnskap for en bedre verden

COGNITE