Published in: Knowledge-Based Systems (KBS), 293(111678), June 2024. Elsevier.

Better Entity Matching with Transformers through Ensembles

Jwen Fai Low[a], Benjamin C. M. Fung[a,*], Pulei Xiong[b]

[a] *School of Information Studies, McGill University, Montreal, QC, Canada, H3A 1X1*
[b] *Cyber Security, National Research Council Canada, Canada*

Abstract

In this paper, we introduce *AttendEM*, a framework for entity matching (EM), i.e., pairwise identification of duplicates across databases. Eschewing the prevalent focus on text cleaning and training data augmentation of other transformers-based EM solutions, *AttendEM* leverages intra-transformer ensembling of distinctively rearranged text, additional aggregator tokens, and extra self-attention to enhance the base transformer architecture. Against state-of-the-art (SOTA) solutions on the ER-Magellan benchmark datasets, *AttendEM* achieved higher F1 scores in most cases. These SOTA solutions are Ditto (mean improvement of 0.21% with Ditto's own reported results, 3.93% with DAEM's Ditto replication, 2.99% with HierGAT's Ditto replication), DAEM (0.53%), and HierGAT (0.54%). *AttendEM*'s improvements are comparable to solutions that claimed to have outperformed Ditto, HierGAT (Yao et al., 2022) (2.46% compared to *AttendEM*'s 2.99%) and DAEM (Huang et al., 2022) (3.42 % compared to *AttendEM*'s 3.93%), when calculated using results from their respective Ditto replications.

Keywords: entity resolution, entity matching, deduplication, neural networks, deep learning, transformers

*Corresponding authors
Email addresses: jwen.low@mail.mcgill.ca (Jwen Fai Low), ben.fung@mcgill.ca (Benjamin C. M. Fung), pulei.xiong@cnrc-nrc.gc.ca (Pulei Xiong)

1. Introduction

As e-commerce grows in popularity and continues displacing brick-and-mortar retail sales, especially during the COVID-19 pandemic as indicated by the US Census Bureau (Brewster, 2022), so too does the problem of counterfeit listing on e-commerce sites, in which consumers are presented with listings that attempt to deceive the customers into purchasing non-genuine products. The scale of the counterfeit problem afflicting one leading e-commerce site, Amazon, necessitated the deployment of automated tools incorporating machine learning and an investment of over $900 million to tackle the problem, which succeeded in blocking 4 billion bad — "fraudulent, infringing, counterfeit, at risk of other forms of abuse, or presenting significant product quality concerns" — listings in 2021 alone (Amazon, 2022). Even if a listing is genuine, duplicates can be part of concerted efforts by malicious actors to flood marketplaces so that their products gain better visibility (Besedo, 2016; Davies and Mudge, 2018).

The specific problem of detecting counterfeit offers (deceptive listings) have received little attention in scholarly works. However, as "determining offers referring to the same product is a special case of object matching" (Arnold et al., 2016), advances in the more mature field of object matching can have spillover benefits for counterfeit detection.

Object matching is often referred to as *entity matching* (*EM*), where the objective is to determine if two similar records across databases should resolve into one unique real-world entity. Figure 1 shows an example of EM where the left dataset has two candidate pairings found in the right dataset with the second pair being a valid match, i.e., the pair consists of duplicates. As one can see, a typical instance of the EM problem is characterized by the presence of two entities with multiple columns of text, where algorithms may struggle to identify relevant or informative text spans. Algorithms may also lack the capacity to capture all available information of each entity and each column.

Comparing all possible pairs for large databases can be infeasibly time-consuming, hence the existence of an *entity blocking* (*EB*) phase prior to EM that aims to reduce the number of candidate pairs through comparatively computationally cheap algorithms. The combined EB and EM phases is called *entity resolution* (*ER*).

In this paper, we present *AttendEM*, a classifier framework built to accommodate pre-trained

transformer models. The framework is meant to be used during the EM phase of ER and as such has been extensively validated against widely used EM benchmark datasets. Results show that *AttendEM* offers improved performance over previous EM classifiers for a majority of the datasets.

The key contributions made in our paper are as follows:

- *A transformer-based classifier framework for EM.* Our implementation explores pre-trained transformer models and optimization strategies that previous works on transformer-based EM solutions did not, namely domain-agnostic solutions for preserving informative text spans and methods for keeping the classifier from being "distracted" by the less relevant parts of the extra preserved information.

- *Text preprocessing strategies.* Many transformers cannot accommodate all the rich textual information from the multiple columns/attributes that an entity may have, so we investigated if prioritizing the inclusion of sections which are likelier to be informative can improve classifier performance.

- *Information aggregation design for transformers.* We investigated if a single aggregator token, as is standard, is sufficient for aggregating all information within a text span composed of two entities or if spreading out the aggregation over two or more tokens can confer a performance advantage by potentially avoiding the unintentional omission of important information.

- *Attention in the classification head.* The classification head used to process the outputs from the transformer block typically uses only feedforward layers; we examined how adding a self-attention layer to the classification head could improve performance, especially when there is information from multiple aggregator tokens instead of a single token to process.

- *Ensembles.* Different text preprocessing and information aggregation strategies have shortcomings and advantages that could result in performance variance across datasets. Rather than devising a way to select the appropriate strategy for each problem, we investigated the performance improvement that can be found from ensembling the strategies, as the sum of multiple models can often be greater than its parts.

- *Comprehensive empirical evaluation over major benchmark datasets. AttendEM* is validated using the same ER-Magellan datasets that were used to evaluate the previous state-of-the-art

name	category	modelno	price
hp c6020b coated paper 1 roll ⋯	stationery	c6020b	49.88

title	category	modelno	price	Match
hp coated paper 36 inches ⋯	photo paper	c6980a	27.99	✗
hp coated paper 36 inches ⋯	roll paper	c6020b	29.99	✓

Fig. 1. Example of an entity matching problem between two datasets.

(SOTA) transformer-based EM classifiers. An additional ablation study demonstrates how each of the enhancements introduced within our framework may have contributed to the improved performance.

- *Performance improvement.* *AttendEM* performs better on a majority of datasets than SOTA EM solutions, namely Ditto (Li et al., 2020) (average improvement of 0.21% compared against Ditto's own reported results, 3.93% using DAEM's Ditto results, 2.99% using HierGAT's Ditto results), DAEM (Huang et al., 2022) (average of 0.53%), and HierGAT (Yao et al., 2022) (average of 0.54%). As stated in the abstract, the improvements observed in *AttendEM* are comparable to two other solutions that claimed to have outperformed Ditto, HierGAT (Yao et al., 2022) and DAEM (Huang et al., 2022). The mean of their percentage improvements over Ditto, calculated using their respective replications of Ditto results, were 2.46% for HierGAT (compared to *AttendEM*'s 2.99%) and 3.42% for DAEM (compared to *AttendEM*'s 3.93%).

2. Preliminaries and Related Work

2.1. Entity resolution

Entity resolution is "matching records that refer to the same entity across databases" (Christen, 2008) and has also been called deduplication or record linkage. Entity resolution is not to be confused with coreference resolution, which is concerned with finding out whether references within a single document point to the same entity (e.g., whether "I" and "Trump" are the same entity in the following sentence: *Trump was quoted saying, "I would vote for Biden."*).

While many ER solutions have EB and EM phases, not all do. For instance, there are some unsupervised methods, often approaching the ER problem from an information retrieval and graph-theoretic perspective, that achieve sufficient accuracy in identifying duplicates in a single step for certain datasets such that the need for separate EB and EM phases is obviated, e.g., Hall et al. (2008); Kejriwal and Miranker (2013); Zhu et al. (2016); Zhang et al. (2020); Kirielle et al. (2023).

The framework introduced in our work, *AttendEM*, is meant only for the EM phase of ER. Having an EM solution that is decoupled from EB allows *AttendEM* to be freely paired with the many existing EB solutions found in the literature, e.g., Wang et al. (2016); Dou et al. (2019); Shao et al. (2019).

While classification can be performed with the help of humans through methods such as crowdsourcing, e.g., Gokhale et al. (2014) or some hybrid method that mixes human and machine classification, e.g., Haruna et al. (2019), our work is solely concerned with solutions based purely on machine learning (ML). A number of notable ER and EM solutions relying solely on ML have been proposed in recent years, and they include Magellan, DeepER, DeepMatcher, and Ditto.

2.2. Machine learning, deep learning, word embeddings

Deep learning is a specific type of ML featuring many layers of neural networks stacked together, hence the term "deep". Traditional ML commonly refers to learning solutions prior to the advent of deep learning that do not use these stacked neural layers.

As an example of traditional ML, there is Magellan, an ER solution that uses attribute equivalence and overlap to perform EB and uses algorithms such as random forest, Naive Bayes, and SVM for EM. Magellan also represents the upper limits in performance achievable by traditional machine learning, performing well on Structured and Textual datasets from the ER-Magellan benchmark (average F1 score of 88.8 and 83.4) but still lacking when evaluated against Dirty datasets (68.5) (Mudgal et al., 2018).

Deep learning led to the creation of ER solutions, e.g., DeepER (Ebraheem et al., 2018) and Seq2SeqMatcher (Nie et al., 2019), and EM solutions, e.g., DeepMatcher (Mudgal et al., 2018) that typically outperform solutions using traditional ML. In the case of DeepMatcher, while it has a lower average F1 score than Magellan on Structured datasets (87.9), it performed significantly better when evaluated against the Textual (88.0) and Dirty (87.9) groups of datasets (Mudgal et al., 2018). Much of deep learning's advantage in *natural language processing* (NLP) tasks can be attributed to the use of word embeddings, e.g., GloVe and fastText, which are low-dimensional vector representations of words that can model the relationship between words. A well-known example of the relationship that embeddings can capture is the "$king - man + woman = queen$" equation.

2.3. Transformers

Transformers are a further evolution of deep learning. The distinguishing feature of transformers is the presence of an attention mechanism, which allows transformers to assign greater weights to relevant context words when generating the embedding for a word as opposed to static word embeddings such as GloVe and fastText that return the same vector for a word as they treat context equally (e.g., bank in "river bank", "bank loan", and "bank the plane" should be encoded differently but static embeddings do not do this).

Larger — more tunable parameters — transformers being correlated with increased performance in NLP tasks have led to the pursuit ever of larger models, leading to the development of a family of "large language models".

Many researchers (Brunner and Stockinger, 2020; Li et al., 2020; Huang et al., 2022) have looked into solving EM with the aid of transformers. Owing to all the interest garnered by the application of transformers to EM, some researchers have even attempted to tease apart the factors contributing to the success of transformers on EM (Paganelli et al., 2022).

EMTransformer (Brunner and Stockinger, 2020) did not make alterations to the transformer architecture as the authors' intention is simply to evaluate how vanilla transformer varieties fare in EM. EMTransformer was only evaluated against datasets with long textual data and "dirty" datasets with inappropriate or missing values because the authors regard other benchmark datasets as being close to solved problems (near perfect F1 scores) when using non-transformer EM approaches.

Ditto (Li et al., 2020), contemporaneous to the 2020 EMTransformer, emphasized the use of a number of "optimizations" to enhance the performance of the transformer that lies at the heart of Ditto. These optimizations include domain knowledge, data augmentation, and TFIDF summarization. Domain knowledge aims to increase the amount of machine-readable informative signals by identifying important text spans and fencing them in with special tokens. Data augmentation augments the training data by adding additional training data. The additional training examples are generated by altering existing examples through operations such as deletion, shuffling, and swapping — learning from these "harder" examples may make the model more robust against noise. Summarization attempts to keep transformers' attention from wandering through unimportant bits of text by distilling long text spans to only the most important segments.

DAEM (Huang et al., 2022) is a more recent (2022) EM solution featuring transformers and it claims better performance compared to Ditto and all other competing solutions. Unlike EM-Transformer and Ditto, DAEM does not rely on pre-trained transformers; it uses the original Transformer's encoder block sans pre-trained weights. However, in lieu of pre-trained weights, DAEM leveraged the pre-trained embeddings from Facebook's fastText to vectorize words. The Transformer encoder is used to encode the vectorized words. Another key feature of DAEM is its use of an adversarial active learning framework, which serves similar purposes to Ditto's data augmentation: active learning presents the model with "good" training examples while adversarial learning augments existing training data for better robustness. To deal with missing data, DAEM also trains an inter-attribute completion algorithm.

The Hierarchical Graph Attention Transformer, or HierGAT for short, (Yao et al., 2022) is another recent (2022) EM solution that relies on transformers. The key innovation of HierGAT is its combination of the Transformer attention mechanism with the hierarchical graph attention network into a Hierarchical Heterogeneous Graph (HHG). Within the HHG there exists three layers in the hierarchy, which are, in descending order, entities, attributes, and tokens. The lowest level token nodes relies on contextual word embeddings from pre-trained transformers such as BERT. Each higher level node aggregates the embeddings of the constituent component nodes in the level below it; attribute nodes aggregate token nodes, entity nodes aggregate attribute nodes. A variant of HierGAT, HierGAT+, extends the use of graph from pairwise EM to the problem of collective ER where multiple candidates matches exist. As the scope of problem (pairwise EM vs collective ER) alters the testing basis, we restrict ourselves to examining only HierGAT in our paper.

2.4. State-of-the-art EM

In the sea of transformers-based EM solutions, Ditto is considered the quintessential baseline SOTA architecture. Despite its age, Ditto's consistent presence in benchmarks of multiple EM works published within the last two years (2022–2023) — e.g., EM with hierarchical graph attention networks (Yao et al., 2022), EM with adversarial active learning (Huang et al., 2022), domain adaption within EM (Tu et al., 2022), and explainability of deep learning ER solutions (Teofili et al., 2022) — is a testament to its enduring quality. Works in other fields closely related to ER that were published within the last two years have also favored including Ditto as one of the SOTA

baselines for comparison. These include a non-neural entity alignment (EA) solution (Leone et al., 2022) and an unsupervised text matching (TM) solution (Ahmadi et al., 2022).

The towering stature of Ditto is cemented by its re-publication as a Special Issue Paper in the VLDB Journal (Li et al., 2023b). This paper has been updated with additional tables, data, and discussion giving greater insight into the inner workings of Ditto. Crucially, although separated by 3 years, the experimental results from the 2023 paper re-introducing Ditto (Table 5 in Li et al. (2020)) are exactly the same as the ones found in 2020 (also Table 5 in Li et al. (2023b)). The results from papers presenting solutions that have demonstrated themselves to have outperformed Ditto such as HierGAT and DAEM are not found in the 2023 Ditto paper. The new Ditto paper did include results from EMTransformer (Table 13 in Li et al. (2023b)) and described EMTransformer's approach as being similar to "baseline" Ditto without the enhancements that make up the "full" Ditto; the results in the new Ditto paper still shows EMTransformer outperforming the "full" Ditto when evaluated against the Abt-Buy dataset.

In this paper, when validating *AttendEM*, we replicated as closely as possible the experimental conditions that produced the results found in the Ditto paper as we intend to compare *AttendEM* against Ditto.

We did note that the two different 2022 papers that evaluated EM solutions on much of the same ER-Magellan benchmark datasets, DAEM (Huang et al., 2022) and HierGAT (Yao et al., 2022), did not reuse the scores from Ditto but evaluated the open source Ditto code in conditions that appeared similar to the original paper's. The reported scores for Ditto in the DAEM and HierGAT papers are significantly lower than those found in the original Ditto paper. DAEM and HierGAT, naturally, outperformed Ditto in their respective publications. For the sake of fairness and comprehensiveness, we are including in our comparison (1) DAEM's replication of Ditto and (2) HierGAT's replication of Ditto as well as (3) DAEM's own scores, (4) HierGAT's own scores, and (5) the scores of the second best model in the DAEM paper, EMTransformer (Brunner and Stockinger, 2020).

2.5. Architectures of transformers as they pertain to AttendEM

This section describes the distinguishing architectural features of transformers, with particular emphasis on parts of the architecture that we either modified or repurposed, to facilitate better

understanding of the enhancements found in our framework. A more complete picture of the different architectures can be found in the respective papers that introduced them.

The Transformer, first introduced by Vaswani et al. (2017), transformed the NLP research landscape by achieving then state-of-the-art results purely through the use multi-head self-attention and the deliberate omission of recurrence in its architecture, although it is still auto-regressive. Copies of the self-attention mechanism are grouped into encoders and decoders, with encoders responsible for parsing inputs and decoders responsible for generating text as outputs (Section 2.5.2). To distinguish between the original Transformer and its many derivatives, we refer to all descendants as *transformers*, always in the lowercase though not necessarily always in plural.

The next major machine learning architecture to emerge in the NLP domain after the introduction of the Transformer is BERT (Bidirectional Encoder Representations from Transformers). By omitting the original Transformer's decoder layer, which expects temporally masked inputs and is therefore unidirectional, and incorporating only the encoder layer which is direction-agnostic as it attends to both left and right contexts, BERT generates bidirectional encoder representations. BERT improved the model's language representation through unsupervised pre-training, with masked language modeling (MLM) and next sentence prediction (NSP) as its objectives. Weights for pre-training are used to initialize the transformer during fine-tuning, where BERT is trained to perform supervised downstream tasks such as question answering. Since there are a variety of pre-training and downstream task objectives, BERT allows the output layer on top of the encoder stacks to be swapped based on the task at hand. The swappable output layers are also known as task-specific heads (Section 3.4).

DeBERTa (He et al., 2021) improved upon BERT with disentangled attention and enhanced mask decoder. Disentangled attention refers to a mechanism where each input word is represented by two vectors, one for its content and another for its position, instead of one vector combining content and position as is standard. The enhanced mask decoder refers to the improvements DeBERTa made to BERT's masked language modeling pre-training objective. It is important to note that the decoder in the enhanced mask decoder is not equivalent to the decoder found in the Transformer. In a number of benchmarks including MNLI and SQuAD, DeBERTa outperformed other popular BERT derivatives such as RoBERTa and ALBERT.

ConvBERT (Jiang et al., 2020) substituted the regular self-attention heads in BERT with a

form of mixed attention, which is self-attention with integrated convolution, that better captures local dependencies of different tokens. The convolution operation is called span-based dynamic convolution as it takes a local span of tokens as inputs instead of just a single token as in regular dynamic convolution. As ConvBERT's design goal is to be small and efficient, it has compared itself against other models of similar size, most notably ELECTRA, which was outperformed by ConvBERT in a number of benchmark tasks (MNLI, SQuAD, etc.); ELECTRA itself outperformed RoBERTa and ALBERT in a majority of the same benchmarks (Clark et al., 2020).

Both DeBERTa and ConvBERT, as derivatives of BERT, are bidirectional.

2.5.1. Tokenization

A feature common to transformers is that the first step for processing text spans is to tokenize them. The tokens are then vectorized into embeddings. Different tokenization algorithms such as byte-pair encoding (BPE) and WordPiece exist to break text down into words or subwords. For instance, common words in BPE are tokenized into full words while uncommon words are turned into subwords, e.g., "defragmentation" is decomposed into "defragment" and "ation". Subword tokenization allows the vocabulary of the tokenizer to remain small but still capable of returning tokens when it encounters out-of-vocabulary words.

In addition to word and sub-word tokens, transformers also utilize special tokens. The Transformer was built with translation in mind, where the number of input tokens often do not match the number of output tokens, hence the need to allow the decoder to dynamically determine the length of the output. The special tokens inform the decoder when to start (start-of-sentence/beginning-of-sentence token) and when to stop (end-of-sentence token) generating text.

In BERT, where text generation is not the goal and decoders are absent, the special beginning- and end-of-sentence tokens remained in the architecture though their purposes have evolved. BERT's `[CLS]` serves to both mark the beginning of a text sequence and to aggregate information of the entire sequence. The aggregated information can be used in downstream tasks such as classification, where the classifier head relies only on the embeddings of the `[CLS]` token to make predictions. BERT's `[SEP]` marks the end of a text span, but if two text spans are present such as in the case of question-answer pairs, an additional `[SEP]` separates the first and the second spans.

The flexibility of these special tokens, as can be seen from the transition of their purposes from

Transformer to BERT, motivated us to investigate if a new scheme for the insertion of the special tokens, e.g., using the special tokens in numbers greater than prescribed in BERT and Transformer, can improve classification performance (Section 3.4).

2.5.2. Self-attention, encoders, and decoders

The Transformer essentially consists of stacks of encoders feeding into stacks of decoders. An encoder layer consists of a multi-head self-attention mechanism followed by a fully connected feed-forward network. The self-attention mechanism in encoders attends to all words in a text sequence. A decoder layer differs from encoders in two ways. First, the multi-head self-attention sub-layer in the decoder takes the decoder's own masked output from the previous timestep as its input — the very first timestep uses the special beginning-of-sentence token (Section 2.5.1) as the seed as no previous timestep exists. Generation terminates when the decoder generates an end-of-sentence token. The second difference between encoders and decoders is that a decoder layer inserts an additional multi-head self-attention sub-layer that takes the output of the encoder layer as its input.

Having neural networks segmented into encoders and decoders is not unique to the Transformer. Various types of neural network architectures such as autoencoders and Recurrent Neural Networks (RNNs) have also used an encoder and decoder setup. The layout of transformers are not bound to an encoder-decoder arrangement though, as can be seen in the case of BERT, which relied only on encoders because BERT was never intended for text-generation purposes.

A key insight from BERT is that there are no strict rules in the use of the Transformer's constituent parts, e.g., encoders do not necessarily have to be accompanied by decoders. This was one of the motivating factors for our insertion of a decoder layer as an additional discriminator in the classifier head to process the embeddings from multiple aggregator tokens (Section 3.5).

Another lesson of transformers and BERT is that the context surrounding a token matters since context is taken into account by the self-attention mechanism when generating embeddings. Our experiment with different text preprocessing methods (Section 3.3) attempts to determine if differing contexts that arise from rearranging spans within text sequences can result in aggregator token output embeddings that are sufficiently varied for ensemble construction to be viable.

2.5.3. Task-specific heads and aggregator tokens

Most BERT derivatives use an aggregator/beginning-of-sentence token (Section 2.5.1) like BERT. Furthermore, most transformers have adopted the format whereby the "core" self-attention layers are considered separable from the task-specific heads and text tokenization, with new transformer derivatives mostly confining their architectural improvements to the core whilst ignoring the task-specific heads and tokenization. Our framework takes advantage of these conventions: if we restrict our modifications to only altering parts of the architecture outside the core, then regardless of whatever enhancements are made to the core, e.g., the innovations introduced by ConvBERT and DeBERTa, so long as during tokenization the sequences are prepended with aggregator tokens whose output embeddings are used for downstream classification, the classification performance of these transformer variants will still be influenced by our modifications.

Increasing the clarity of aggregated data through the use of multiple aggregator tokens (Section 3.4) was one of the modifications we investigated since tokenization lies outside the core and aggregator tokens are a common design feature of BERT derivatives. Finding ways to better utilize the outputs from the core was the other avenue explored, which we achieved through ensembling the output embeddings from text that has undergone different preprocessing and the addition of a decoder layer to the classifier head (Sections 3.3 and 3.5).

3. AttendEM: a Framework for Entity Matching

This section introduces the problem setting (Section 3.1), gives high-level descriptions of the *AttendEM* framework (Section 3.2) and transformers' architectures (Section 2.5) before delving deeper into the framework's key components: the text preprocessing step (Section 3.3), the information aggregator token design (Section 3.4), and the ensembling of individual models (Section 3.5).

3.1. Problem definition

In the problem of entity resolution (ER), an *entity* refers to a unique real-world object, e.g., product, person, etc. A *record* refers to a mention of an entity found within a collection, i.e., *datasets* that records the attributes of said objects. Given two such datasets D and D' with symmetrical attributes (i.e., all the attributes found in one dataset have their respective equivalents in the other

dataset) that contains values which can be represented as strings (for instance, if the values are integers or floats, they can be converted to text), the goal of ER is to find all pairs of records between D and D' that point to the same real-world entity. These record pairs that are found are referred to as *matches*. Pairs of records that point to two distinct entities are *non-matches*.

A standard ER pipeline consists of two stages, entity blocking (EB) and entity matching (EM). The *blocker* in EB aims to filter out the the non-matches found in the cross product $D \times D'$, which typically vastly outnumbers the matches, into C, a subset of $D \times D'$ containing candidate matches. Blockers are assumed to not produce any false negatives. A *matcher* in EM then determines which of the record pairs in C are matches and which are non-matches.

The focus of this paper is on the design of an effective matcher. Specifically, given labeled examples — record pairs which indicate whether the pairs are matches or non-matches — for every dataset pair split into three disjoint subsets — training, validation, and testing — we aim to train the matcher using training data, determine the optimal hyperparameter for the matcher with the validation data, and evaluate the matcher's performance using testing data.

3.2. Process overview

When tasked with determining if a pair of textual entities, A from dataset D and B from dataset D', are matches, we first obtain different representations of the entities by rearranging the text. The rearranged texts are then tokenized. For each set of tokens, we insert aggregator tokens in them. The number of aggregator tokens as well as the insertion location differs based on how the texts were rearranged. Each set of tokens is fed into the transformer, one set at a time. Once the transformer has processed the last set, the embeddings for all the aggregator tokens found in all the different sets are concatenated. The concatenated embeddings are then fed into a classifier block consisting of self-attention, linear, and softmax layers that outputs a probability of the pair being duplicates. The model is fine-tuned by learning from classification errors. Figure 2 shows the EM process within our framework.

3.3. Text preprocessing

There are three ways that the raw text can be rearranged within our framework. Two of these are text summarization. The third way is column/attribute alignment between entities of two datasets.

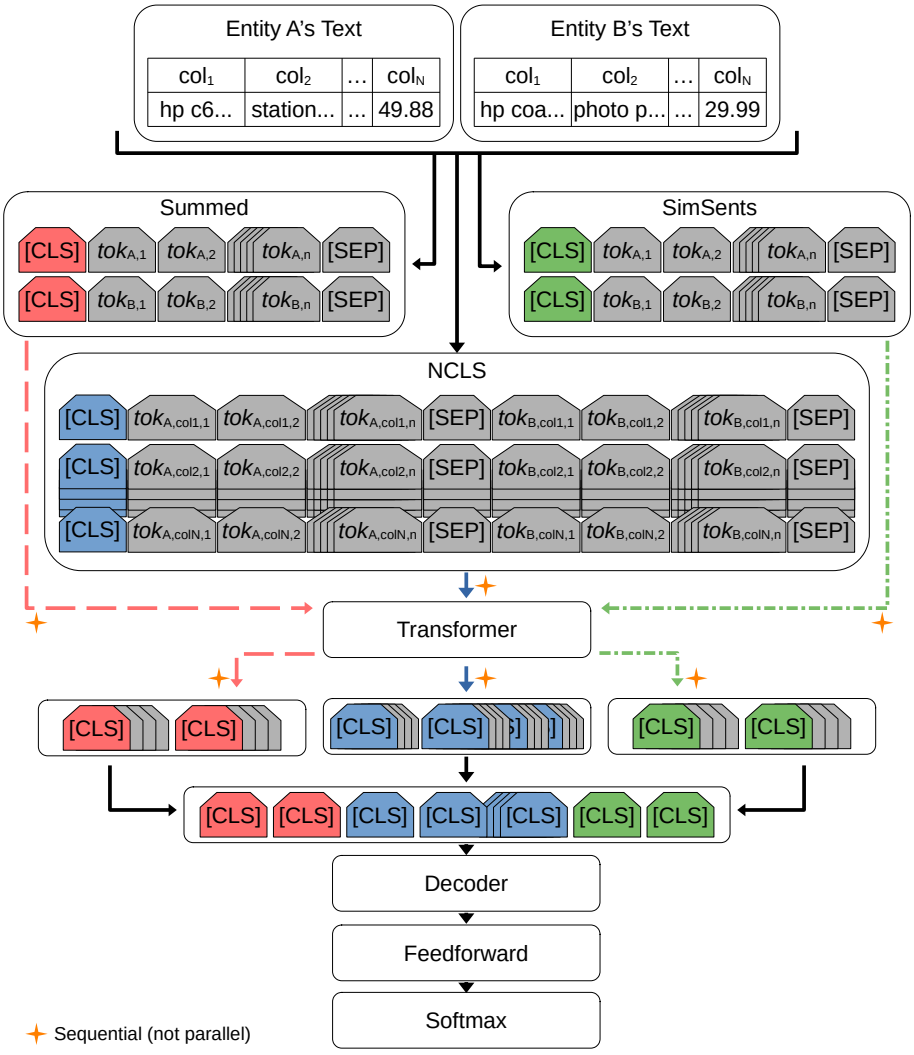

Fig. 2. *AttendEM*'s workflow showing text preprocessing (NCLS/Attribute Alignment, Summed, Simsents, detailed in Section 3.3), aggregator token insertion (dual-CLS, NCLS, detailed in Section 3.4), ensembling the outputs from the different representations of the text span as described in Section 3.5, and the classifier head containing a self-attention layer (decoder) along with feedforward layers processing the concatenated `[CLS]` embeddings as described in Sections 2.5.2 and 3.5. Transformer stands in for a substitutable architecture, e.g., DeBERTa or ConvBERT. As stated in the bottom left corner of the figure, a four-pointed star denotes that the process is sequential as opposed to parallel.

The first summarization method sorts the sentences found in a span of text based on the summed TFIDF (term frequency-inverse document frequency) values for all the words found in a sentence, with high-scoring sentences taking precedence over others. This approach is known as *Summed*. The second summarization method sorts sentences based on their similarity to a key sentence, with priority given to high-similarity sentences. Similarity is determined by the cosine distance between the TFIDF vectorized representations of a sentence and the key sentence. The key sentence used is

the title/name of an entity. Sentences from the rest of the selected attributes/features are the ones that will be compared against the key sentence and sorted. This approach is known as *SimSents*.

For attribute alignment (*Aligned*), the values from equivalent columns of an entity pair are placed next to each other. For instance, given an entity A within dataset D with the columns "title" and "brand" and an entity B within dataset D' with the columns "name" and "brand", an entity pair will be concatenated in the following order: A_{title}, B_{name}, A_{brand}, and B_{brand}.

3.4. Aggregator tokens: dual-CLS and NCLS

Transformers take tokenized text sequences as inputs. An input sequence of tokens consists of not just the original text but also additional tokens that perform special functions. Often, there are two special tokens, one meant to aggregate information for the text sequence, e.g., `[CLS]` and another meant to demarcate the end of a sequence, e.g., `[SEP]`. In pairwise tasks, the tokenization scheme of most transformers take the form of having two text sequences concatenated together with a single aggregator prepended to the first sequence and an end-of-sequence token appended to at the end of each sequence, i.e., `[CLS]` $tok_{seqA_1}, tok_{seqA_2}, \cdots$ `[SEP]` $tok_{seqB_1}, tok_{seqB_2}, \cdots$ `[SEP]`.

Having observed the recent successes in modifying BERT to use multiple CLS tokens in abstractive and extractive text summarization tasks (Liu and Lapata, 2019), we tested a similar modification for EM. For Summed and SimSents, we added an extra `[CLS]` to the start of the second sequence, making the final concatenated sequence `[CLS]` $tok_{seqA_1}, tok_{seqA_2}, \cdots$ `[SEP] [CLS]` $tok_{seqB_1}, tok_{seqB_2}, \cdots$ `[SEP]`. The maximum number of tokens for each entity in an entity pair is half of a typical transformer's 512-token limit minus the number of special tokens. This aggregator scheme is referred to as *dual-CLS* to contrast with the original formulation that uses only one token, single-CLS.

The *NCLS* aggregator scheme is only applicable for Aligned, i.e., entity pairs whose texts have undergone attribute alignment. This scheme places each entity-attribute pair between an aggregator token and an end-of-sequence token. The number of special tokens used depends on the number of attributes that the datasets have. For instance, datasets with three columns, "name", "brand", and "description", will have three sets of special tokens and the final concatenated sequence will take the form of `[CLS]` $tok_{nameA_1}, tok_{nameA_2}, \cdots, tok_{nameB_1}, tok_{nameB_2}, \cdots$ `[SEP] [CLS]` tok_{brandA_1}, $tok_{brandA_2}, \cdots, tok_{brandB_1}, tok_{brandB_2}, \cdots$, `[SEP] [CLS]` $tok_{descA_1}, tok_{descA_2}, \cdots, tok_{descB_1}, tok_{descB_2}$,

$\cdots$ `[SEP]`. The number of tokens allotted to each entity is the same as in the dual-CLS scheme.

3.5. Ensemble

In traditional machine learning, a process known as *ensembling* takes the intermediary predictions from multiple models to derive a final prediction, with Random Forest being a well known example of such ensemble classifiers. With neural language models, a common method for creating an ensemble is by averaging the outputs of models with different architectures. The intuition behind ensembles is that a group of classifiers can collectively compensate for the deficiencies of the individual models.

A barrier to the widespread use of ensembles of large language models is the size of the models as each of the GPUs that the models are trained on usually can only fit a single model in its memory. This memory limitation motivated us to devise a method for creating an ensemble while relying only on a single instance of a single transformer architecture. We accomplished this by varying the inputs given to the transformer so that it produces different outputs. Inputs are varied by using different methods for rearranging the text and inserting aggregator tokens as described in Sections 3.3 and 3.4.

The transformer takes as input the tokens from only one text variant at a time. From the transformer's output, only the embeddings for the CLS tokens are retained. These embeddings are concatenated and fed into the classification block.

While the typical classification block consists of linear layers, our framework adds an additional self-attention layer, specifically the decoder block from the original Transformer paper (Vaswani et al., 2017), before the linear layers. Since we are not using our framework for predicting words sequentially like the Transformer, we did not need to mask the inputs for the decoder. Inputs for both self-attention sub-layers of the decoder layer we repurposed are the same, which is the output embeddings of the aggregator tokens. Adding self-attention to the classification block is driven by the use of multiple CLS tokens in our framework. The standard classification blocks being made up of only linear layers may have been sufficient for accommodating the embeddings from a single CLS token, but in the presence of multiple CLS tokens, an attention mechanism may help each CLS token learn its relation to other CLS tokens, which in turn may improve classification performance.

4. Framework Assessment

4.1. Datasets

Our framework is benchmarked against datasets from Mudgal et al. (2018), which have been used to assess the performance of other EM solutions, e.g., Li et al. (2020); Huang et al. (2022). This collection of datasets, going by the name of ER-Magellan datasets, predate the Magellan ER solution and have been slowly built up over the years by the AnHai Doan research group. Earlier iterations that contain a subset of the datasets found in Mudgal et al. (2018) have also been used to assess the performance of EM solutions, e.g., Ebraheem et al. (2018). The training, validation, and test splits follow a 3:1:1 ratio for all the datasets and the splits we used are the exact same splits used in Mudgal et al. (2018); Li et al. (2020); Huang et al. (2022). The summary statistics for the datasets can be found in Table 1.

Table 1. Statistics of ER-Magellan benchmark datasets

Datasets	# Candidates (Pairs)	# Matches (Pairs)	Attributes
Amazon-Google (AG)	11,460	1,167	3
Beer	450	68	4
DBLP-ACM (DA)	12,363	2,220	4
DBLP-GoogleScholar (DS)	28,707	5,347	4
Fodors-Zagats (FZ)	946	110	6
iTunes-Amazon (IA)	539	132	8
Walmart-Amazon (WA)	10,242	962	5
Abt-Buy (AB)	9,575	1,028	3
Company	112,632	28,200	3

4.2. Experiment setup

Most recent publications have converged upon F1 as the metric of choice for evaluating EM classifiers; we will adhere to this convention as well.

As we have previously noted in Section 2.4, there have been conflicting scores reported for Ditto so we will be including in our comparison all versions of published pairwise EM scores (Li et al., 2020; Huang et al., 2022; Yao et al., 2022) for the purposes of transparency.

For EMTransformer, the original paper (Brunner and Stockinger, 2020) evaluated EMTransformer only on Dirty datasets and the Abt-Buy dataset while the DAEM paper (Huang et al.,

2022) evaluated EMTransformer only on Structured datasets so there are no overlaps in the reported scores, hence no conflicts.

Like Ditto (Li et al., 2020), the number of epochs (5, 10, or 20) used during training in our framework varied based on the dataset. The checkpoint used is the one with the lowest loss instead of the highest F1 score on the validation set. And like Ditto, we first determined which transformers performed best without the modifications found in our framework (text preprocessing, multiple aggregator tokens, and ensemble) before deciding on which architecture to use on which datasets. Note that for EMTransformer (Brunner and Stockinger, 2020; Huang et al., 2022), the best results from the four different transformer architectures tested (BERT, XLNet, RoBERTa, and DistilBERT) are the ones reported. With our framework, we tested only two architectures, ConvBERT (Jiang et al., 2020) and DeBERTa (He et al., 2021). Using two architectures is sufficient for demonstrating that the improvement resulting from *AttendEM*'s modifications is not restricted to a single architecture. The size discrepancy between the two architectures we chose also shows that *AttendEM*'s improvement is not dependent on the number of parameters found in an architecture. We settled on using ConvBERT for the Amazon-Google, iTunes-Amazon, Dirty iTunes-Amazon, and Dirty Walmart-Amazon datasets and DeBERTa for the rest. The pre-trained weights for ConvBERT are `YituTech/conv-bert-base` while DeBERTa used `microsoft/deberta-v3-base`. ConvBERT has 106M parameters. DeBERTa has 184M parameters. The mean of the F1 scores from 5 repeated runs are reported.

All experiments are conducted on a single Nvidia Titan Xp with 12GB of memory alongside an Intel Xeon E5-2697 v4 @ 2.30GHz. The *AttendEM* framework is implemented in PyTorch. Transformer models are from the HuggingFace `transformers` package. When using ConvBERT, the decoder block within the classifier block has 1 layer with 4 attention heads. DeBERTa uses 1 layer with 8 attention heads. The learning rates are 2e−5 with a weight decay of 1e−2 for the transformer and 1e−5 with a weight decay of 1e−3 for everything else. Learning rate decreases linearly with zero warmup steps. For the Python packages `random`, `numpy`, and `pytorch`, we used a random seed of 42.

4.3. Results

Table 2 compares the performance of *AttendEM* against several other SOTA EM frameworks.

Our framework is notably robust against Dirty datasets, where it outperforms all the EM classifiers that it was compared against in two instances and is on par in two other instances (after rounding based on the number of significant figures reported by the competing solution).

Results are mixed with Structured datasets, but *AttendEM* still made a strong showing. Against Ditto, DAEM's reported Ditto results, and HierGAT's reported Ditto results, *AttendEM* outperformed Ditto in all instances — unsurprising since DAEM and HierGAT themselves outperformed their Ditto replication attempts. Against DAEM, *AttendEM* achieved similar or better results in four out of seven datasets. Against HierGAT, *AttendEM* also achieved similar or better results in four out of seven datasets. DBLP-GoogleScholar and Walmart-Amazon are the two datasets in common where *AttendEM* performed worse compared to DAEM and HierGAT. Against EM-Transformer, *AttendEM* has the lead in all datasets save for DBLP-GoogleScholar. Evidently, *AttendEM* is weak against DBLP-GoogleScholar, although there are no immediately obvious reasons for its underperformance as the content and structure of DBLP-GoogleScholar is highly similar with DBLP-ACM, in which *AttendEM* was able to achieve performance comparable with other EM solutions.

With Textual data, out of the four solutions examined, our framework is outperformed by Ditto and EMTransformer, although *AttendEM* consistently placed second.

4.4. Implications

DAEM and Ditto both heavily emphasized the effort spent in cleaning up the text during the text preprocessing stage. DAEM uses inter-attribute completion to fill in missing values (this requires that the missing value for an attribute can be found within the values for other attributes). Ditto uses domain knowledge to identify text spans so that special tokens can be inserted to make it easy for the transformer to identify the span type (e.g., the phone number "(866) 246-6453" is tokenized as "(866) 246 - `[LAST]` 6453 `[/LAST]`" where `[LAST]` indicates the last 4 digits of the phone number). Domain knowledge is also used to normalize non-standard entries (e.g., rewriting "5 %" and "5.00 %" to "5.0 %"). Another aspect of DAEM and Ditto is their use of additional training data. DAEM uses active learning and selects examples high in informativeness and representativeness for the the classifier to learn from. Furthermore, DAEM uses adversarial learning to create fake matches as extra training examples. As for Ditto, data augmentation is used

Table 2. F1 scores on the ER-Magellan datasets. Scores for DAEM are from the DAEM paper (Huang et al., 2022). Scores for HierGAT are from the HierGAT paper (Yao et al., 2022). Scores for Ditto are from the Ditto (Li et al., 2020, 2023b), DAEM (Huang et al., 2022), and HierGAT (Yao et al., 2022) papers. Scores for EMTransformer are from the EMTransformer (Brunner and Stockinger, 2020) and DAEM (Huang et al., 2022) papers. The ΔF1 is a comparison between *AttendEM* and the best (or next best) performing model. If more than one ΔF1 are reported side-by-side, the leftmost ΔF1 represents a comparison with results published in the original paper while the remaining ΔF1 are with results from other papers' replication attempts. The highest scores are in bold, second best are underlined.

Datasets	DAEM (DAEM)	HierGAT (HierGAT)	Ditto (Ditto)	Ditto (DAEM)	Ditto (HierGAT)	EMTr (EMTr)	EMTr (DAEM)	*AttendEM*	ΔF1
Structured									
Amazon-Google	73.1	76.4	75.58	72.9	74.1	-	71.4	**77.67**	+1.27
Beer	90.3	93.3	**94.37**	84.4	84.6	-	87.5	91.04	-3.33/+6.64/+6.44
DBLP-ACM	98.9	99.1	98.99	98.7	99.0	-	98.9	**99.12**	+0.02
DBLP-GoogleScholar	**97.3**	96.3	95.60	95.6	95.8	-	96.0	95.85	-1.45
Fodors-Zagats	**100.0**	**100.0**	**100.00**	99.1	98.1	-	**100.0**	**100.00**	+0.00/+0.90/+1.90
iTunes-Amazon	**100.0**	96.3	97.06	92.4	92.3	-	93.1	98.90	-1.10
Walmart-Amazon	87.7	**88.2**	86.76	83.1	85.8	-	85.9	86.81	-1.39
Dirty									
DBLP-ACM	-	**99.1**	99.03	-	98.9	98.9	-	99.08	-0.02
DBLP-GoogleScholar	-	95.8	95.75	-	95.4	95.6	-	**95.89**	+0.09
iTunes-Amazon	-	94.7	95.65	-	92.9	94.2	-	**96.45**	+0.80
Walmart-Amazon	-	**86.3**	85.69	-	82.6	85.5	-	86.29	-0.01
Textual									
Abt-Buy	-	89.8	89.33	-	88.9	**90.9**	-	90.11	-0.79
Company	-	88.2	**93.85**	-	87.5	-	-	92.50	-1.35/+4.30

to create more training examples and the augmentation operators can distort the training examples so comprehensively that the labels become incorrect, necessitating the use of interpolation to rein in the changes.

Compared against DAEM and Ditto, our approach forgoes text cleaning and data augmentation. Our approach's text preprocessing only rearranges the text. In spite of that, our approach still manages to achieve higher F1 scores for many of the benchmark datasets.

The only requirement of our framework is that the two datasets of an EM problem has to have identical (or substitutable) attributes/columns. This is to facilitate attribute alignment where comparable columns from the two datasets are grouped together (see Section 3.3). All the benchmark dataset-pairs featured in Ditto, DAEM, HierGAT, and EMTransformer meet this criteria as the datasets in each pair have an identical number of attributes that mirror each other.

HierGAT is similar to our approach based on the absence of text cleaning and the use of additional training data in their approach. Another similarity between HierGAT and our approach is their use of multiple `[CLS]` tokens. Significant differences remain. HierGAT used multiple CLS tokens because each attribute embedding is represented as a CLS token and the concatenated attribute embeddings go on to form entity embeddings. Our use of multiple CLS tokens comes

from text preprocessing (Section 3.3) and also from building ensembles (Section 3.5), with only the Aligned text preprocessing scheme bearing some resemblance to HierGAT's attribute embeddings. Another difference is the core novel feature underlying each framework. In HierGAT, this is the hierarchical heterogeneous graph, which needs to be constructed and trained to obtain attribute and entity embeddings. Our approach invested in building intra-model (i.e., using only a single transformer architecture) ensembles.

4.5. Ablation study

Here, we investigate the influence each of our enhancements on the base transformer architecture exert on EM classification results as well as runtimes. There are too many possible configurations for ensembles (e.g., Summed and Simsents can used with single-CLS instead of dual-CLS, default with single-CLS can be ensembled together with Summed with dual-CLS etc.) so we only presented a small informative selection of configurations. All ensembles use ConvBERT and are run for 10 epochs and the F1 scores are means from five runs. The training runtimes are means from the five runs as well. The text preprocessing runtimes are averages from the five-run averages of all ensembles.

The baseline is "default, sans attention", which uses the transformer as originally designed with single-CLS and is comparable to the EMTransformer approach. *AttendEM* is represented by "Aligned + SimSents + Summed". To investigate the impact of the additional self-attention layer in the classification block on ensembles that contain multiple models, we also present results for *AttendEM* with the attention in the classifier head removed.

Runtimes, which can be found in Table 3, appear fairly stable across different datasets. Each additional model in the ensembles increases the training runtime almost linearly by 0.7×, i.e., two-model ensembles takes 1.7× the default's time to run while three-model is roughly $1.7\times + 0.7\times = 2.4\times$ the default's. During evaluation/inference (no backpropagation), the burden imposed by additional models in an ensemble is greater than during training, with each extra model adding 1.0× to the runtime, thus making two-model ensembles' runtimes around 2.0× the default's and three-model ensembles at about 3.0× the default's.

In general, each additional model in the ensemble imposes an additional linear increase in runtime. Therefore, the full *AttendEM* is unsuited for time-sensitive applications such as real-time

deduplication of database entries. However, for periodic (e.g., weekly) purging of duplicates from databases, *AttendEM* can be a promising solution. Expending more time for better performance without violating computing resource constraints (e.g., GPU memory limit) can be a worthwhile tradeoff, especially for organizations without easy, affordable access to computing resources that are optimal for running transformers and other types of neural network yet still want to obtain the highest quality results when performing entity matching.

From the F1 scores obtained by the ensembles/models in Table 4, we can see that there is no single text preprocessing method that works well across different datasets when using additional aggregator tokens and self-attention during classification. For instance, Aligned (97.06) performs better than the default (95.42) in Dirty iTunes-Amazon but falters when facing against the Dirty Walmart-Amazon dataset (83.27, worse than the default's 85.48). Meanwhile, the reverse applies to Summed, where it outperforms the default on Dirty Walmart-Amazon (85.52 vs 85.48) but not on Dirty iTunes-Amazon (95.06 vs 95.42).

Ensembles with two models feature noticeable variability across datasets as well. The "Aligned + Summed" ensemble performed better than the default on Dirty iTunes-Amazon (95.74 vs 95.42) but is worse when it comes to Dirty Walmart-Amazon (85.05 vs 85.48).

The performance of two-model ensembles is not necessarily the average of their constituents. For Dirty iTunes-Amazon, the SimSents (94.37) and the Summed (95.06) models individually are worse than the default (95.42) but perform almost as good as the default in the "SimSents + Summed" ensemble (95.41). A similar situation can be found in Dirty Walmart-Amazon when comparing the individual Aligned (83.27) and SimSents (84.19) models against the ensemble (85.79) and default (85.48).

The full three-model ensemble we used to represent *AttendEM* seems to avoid wild swings in F1 scores when moving between datasets. While the full *AttendEM* may not necessarily be the best performing ensemble (e.g., the individual Aligned model with F1 of 97.06 outscores the full *AttendEM*'s 96.45 in Dirty iTunes-Amazon), it manages to maintain its performance lead over the default across different datasets unlike the smaller ensembles.

Self-attention in the classifier head appears necessary for models with multiple aggregator CLS tokens and for large ensembles featuring such models. For both Dirty iTunes-Amazon and Dirty Walmart-Amazon, the use of dual-CLS tokens resulted in worse performance (94.75 Dirty IA,

85.39 Dirty WA) than the default sans self-attention model (95.10, 85.85); the introduction of self-attention in the classifier helped regain the lost performance (96.10, 85.61). In the case of ensembles, without self-attention, *AttendEM* (96.45) would have scored lower (94.76) than the default Dirty iTunes-Amazon (95.42). When there is only a single model, such as the default, self-attention can potentially degrade performance, as can be seen in the case of Dirty iTunes-Amazon (95.10 vs 95.42).

5. Conclusion and Future Work

In this paper, we introduced *AttendEM*, a transformer-based framework for entity matching. Our evaluation showed that ensembled models built under this framework achieve state-of-the-art results in a majority of EM benchmark datasets. For the datasets where *AttendEM* performed better, the observed improvement in F1 ranges from 0.13 to 2.09 (the Beer and Company datasets being the two exceptions). When compared against Ditto's own results over all 13 datasets, i.e., including datasets where *AttendEM* performed worse, the average improvement was +0.16. If we use DAEM's reported results for Ditto, the average is +3.31 for all seven evaluated datasets. And if we use HierGAT's reported results for Ditto, the average is +2.60 for all 13 evaluated datasets. When compared against DAEM, the range of improvement is from 0.22 to 4.57 (the DBLP-GoogleScholar, iTunes-Amazon, and Walmart-Amazon datasets being the three exceptions) with a mean of +0.30 over all seven datasets that DAEM was tested against. And when compared against HierGAT, the range is from 0.02 to 4.30 (the five exceptions being Beer, DBLP-GoogleScholar, Walmart-Amazon, Dirty DBLP-ACM, and Dirty Walmart-Amazon) with a mean of +0.48 over all 13 datasets that HierGAT was tested against.

Although the improvement may seem slight, *AttendEM*'s improvements are comparable to two other solutions that claimed to have outperformed Ditto, HierGAT (Yao et al., 2022) and DAEM (Huang et al., 2022). The mean of their percentage improvements over Ditto, calculated using their respective replications of Ditto results, were 2.46% for HierGAT (compared to *AttendEM*'s 2.99%) and 3.42% for DAEM (compared to *AttendEM*'s 3.93%).

AttendEM demonstrates that ensembles combined with additional aggregator tokens and self-attention can improve the performance of a transformer in the domain of EM, albeit at the cost of greater runtimes as the transformer needs to cycle through each model's preprocessed text

Table 3. Runtimes for different ensembles and text preprocessing methods. Default uses single-CLS, SimSents and Summed use dual-CLS while Aligned uses NCLS. *AttendEM* is "Aligned + SimSents + Summed".

Ensemble	Dirty IA	Dirty WA
Training (per epoch)		
Default	31s	9m52s
Default, w/o attention	0.969×	0.971×
Default, 2CLS	1.02×	0.998×
Default, 2CLS, w/o attention	0.969×	0.958×
Aligned	1.00×	1.04×
SimSents	1.00×	1.03×
Summed	1.00×	1.04×
Aligned + SimSents	1.64×	1.66×
Aligned + Summed	1.65×	1.65×
SimSents + Summed	1.59×	1.69×
AttendEM	2.34×	2.41×
AttendEM, w/o attention	2.57×	2.43×
Evaluation (per epoch)		
Default	3s	56s
Default, w/o attention	1.05×	0.987×
Default, 2CLS	1.03×	0.978×
Default, 2CLS, w/o attention	1.05×	0.984×
Aligned	1.02×	0.982×
SimSents	0.979×	0.977×
Summed	1.02×	0.991×
Aligned + SimSents	1.95×	1.93×
Aligned + Summed	1.96×	1.85×
SimSents + Summed	1.94×	1.95×
AttendEM	2.92×	2.98×
AttendEM, w/o attention	3.40×	2.96×
Text preprocessing		
Default	0.414s	0.162s
Aligned	0.875×	0.814×
SimSents	378×	416×
Summed	257×	261×

sequentially. To alleviate the issue with runtime, future work can explore the use of processing units, e.g., GPUs and TPUs, with sufficient memory capable of fitting multiple models at once so that running the ensembles can be parallelized. Alternatively, future work can proceed further down the path of designing a smaller and more efficient *AttendEM* framework, such as reducing the number of computations required at inference time using techniques such as dynamical inference.

Table 4. F1 scores for different ensembles. Default uses single-CLS, SimSents and Summed use dual-CLS while Aligned uses NCLS. *AttendEM* is "Aligned + SimSents + Summed".

Ensembles/models	Dirty IA	Dirty WA
Default	95.42	85.48
Default, w/o attention	95.10	85.85
Default, 2CLS	96.10	85.61
Default, 2CLS, w/o attention	94.75	85.39
Aligned	97.06	83.27
SimSents	94.37	84.19
Summed	95.06	85.52
Aligned + SimSents	96.08	85.79
Aligned + Summed	95.74	85.05
SimSents + Summed	95.41	85.13
AttendEM	96.45	86.29
AttendEM, w/o attention	94.76	86.23

Dynamic inference has been demonstrated to improve the efficiency of transformers, albeit in the domain of computer vision (Li et al., 2023a) and not NLP that is typical of EM.

Filling in missing values and utilizing domain knowledge to normalize text spans and identify attribute types may improve *AttendEM* performance as well. After all, both DAEM and Ditto credited their careful engineering of text cleaning solutions for their frameworks' improved performance over existing solutions.

Pre-training transformers used in *AttendEM* using text that more closely matches the problem domain of the downstream classification task (e.g., pre-training on scholarly publication dataset prior to classification fine-tuning for DBLP-ACM EM task) is a potential avenue for improving EM performance, as the transformers used in our work and in previous works were pre-trained on "generic" text corpora. There are known examples of transformers benefiting from unsupervised pre-training on datasets that bears greater resemblance to the datasets found in downstream tasks. SciBERT outperfomed BERT on NLP tasks in scientific domains such as biomedicine and computer science simply through pre-training on a large multi-domain corpus of scientific texts as opposed to the Wikipedia and literary books corpora used by BERT (Beltagy et al., 2019).

Incorporation of unsupervised (self-supervised) contrastive learning to transformer pre-training can also potentially improve the current *AttendEM* framework. Contrastive learning as applied to transformers resulted in demonstrable improvements in SentEval, a suite of sentence embedding

quality benchmark tasks (Giorgi et al., 2021) and in semantic textual similarity (STS) tasks (Gao et al., 2021). Contrastive learning has also shown success in zero-shot object detection in the field of computer vision (Yan et al., 2022). Note that the unsupervised contrastive learning being referred to here is different from the raw string-level contrastive learning used in the EM framework CorDEL, as the authors of CorDEL themselves have noted (Wang et al., 2020).

Despite the great variety of unique sources in the datasets found within the ER-Magellan benchmark, the two domains of commerce (product listings and company listings) and bibliography make up the entirety of the benchmark. While *AttendEM* has proven itself against the ER-Magellan benchmark, the generalizability of *AttendEM* to other, more specialized, problems such as the merging/de-duplicating of entries from different crime databases (Ahmed et al., 2022) should be studied.

Some of the modifications introduced by *AttendEM* is not specific to the task of EM, namely the use of ensembles and the addition of an attention block in the classification head. Extending the use of these modifications to problems outside of EM warrants investigation.

Acknowledgements

The research is supported in part by the Discovery Grants (RGPIN-2018-03872) from the Natural Sciences and Engineering Research Council of Canada (NSERC) and Canada Research Chairs Program (950-232791).

Competing Interests Statement

The authors have no competing interests to declare.

References

Ahmadi, N., Sand, H., Papotti, P., 2022. Unsupervised Matching of Data and Text, in: 2022 IEEE 38th International Conference on Data Engineering (ICDE), pp. 1058–1070. doi:10.1109/ICDE53745.2022.00084.

Ahmed, S., Gentili, M., Sierra-Sosa, D., Elmaghraby, A.S., 2022. Multi-layer data integration technique for combining heterogeneous crime data. Information Processing & Management 59,

102879. URL: https://www.sciencedirect.com/science/article/pii/S0306457322000115, doi:10.1016/j.ipm.2022.102879.

Amazon, 2022. 2022 Amazon Brand Protection Report. Technical Report. Amazon. URL: https://brandservices.amazon.com/progressreport.

Arnold, P., Wartner, C., Rahm, E., 2016. Semi-Automatic Identification of Counterfeit Offers in Online Shopping Platforms. Journal of Internet Commerce 15, 59–75. URL: http://www.tandfonline.com/doi/full/10.1080/15332861.2015.1121459, doi:10.1080/15332861.2015.1121459.

Beltagy, I., Lo, K., Cohan, A., 2019. SciBERT: A Pretrained Language Model for Scientific Text, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China. pp. 3615–3620. doi:10.18653/v1/D19-1371.

Besedo, 2016. Duplicate Real-Estate Listings. Why Should You Care? URL: https://besedo.com/resources/blog/duplicate-real-estate-listings-why-should-you-care/.

Brewster, M., 2022. Annual Retail Trade Survey Shows Impact of Online Shopping on Retail Sales During COVID-19 Pandemic. Technical Report. United States Census Bureau. URL: https://www.census.gov/library/stories/2022/04/ecommerce-sales-surged-during-pandemic.html.

Brunner, U., Stockinger, K., 2020. Entity matching with transformer architectures - a step forward in data integration, in: Proceedings of the 23rd International Conference on Extending Database Technology (EDBT), OpenProceedings. pp. 463–473. URL: https://digitalcollection.zhaw.ch/handle/11475/19637, doi:10.21256/zhaw-19637.

Christen, P., 2008. Febrl -: An open source data cleaning, deduplication and record linkage system with a graphical user interface, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery,

New York, NY, USA. pp. 1065–1068. URL: https://doi.org/10.1145/1401890.1402020, doi:10.1145/1401890.1402020.

Clark, K., Luong, M.T., Le, Q.V., Manning, C.D., 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators, in: International Conference on Learning Representations (ICLR) 2020.

Davies, R., Mudge, L., 2018. Is it all downhill from here for NZ's e-commerce giant? Hell no, says Trade Me. The Spinoff URL: https://thespinoff.co.nz/business/29-09-2018/is-it-all-downhill-from-here-for-trademe/.

Dou, C., Cui, Y., Sun, D., Wong, R., Atif, M., Li, G., Ranjan, R., 2019. Unsupervised blocking and probabilistic parallelisation for record matching of distributed big data. The Journal of Supercomputing 75, 623–645. URL: https://doi.org/10.1007/s11227-017-2008-8, doi:10.1007/s11227-017-2008-8.

Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., Tang, N., 2018. Distributed Representations of Tuples for Entity Resolution. Proc. VLDB Endow. 11, 1454–1467. URL: https://doi.org/10.14778/3236187.3269461, doi:10.14778/3236187.3269461.

Gao, T., Yao, X., Chen, D., 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings, in: Moens, M.F., Huang, X., Specia, L., Yih, S.W.t. (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic. pp. 6894–6910. URL: https://aclanthology.org/2021.emnlp-main.552, doi:10.18653/v1/2021.emnlp-main.552.

Giorgi, J., Nitski, O., Wang, B., Bader, G., 2021. DeCLUTR: Deep Contrastive Learning for Unsupervised Textual Representations, in: Zong, C., Xia, F., Li, W., Navigli, R. (Eds.), Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online. pp. 879–895. URL: https://aclanthology.org/2021.acl-long.72, doi:10.18653/v1/2021.acl-long.72.

Gokhale, C., Das, S., Doan, A., Naughton, J.F., Rampalli, N., Shavlik, J., Zhu, X., 2014. Cor-

leone: Hands-off crowdsourcing for entity matching, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery, Snowbird, Utah, USA. pp. 601–612. URL: https://doi.org/10.1145/2588555.2588576, doi:10.1145/2588555.2588576.

Hall, R., Sutton, C., McCallum, A., 2008. Unsupervised Deduplication Using Cross-field Dependencies, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA. pp. 310–317. URL: http://doi.acm.org/10.1145/1401890.1401931, doi:10.1145/1401890.1401931.

Haruna, C.R., Hou, M., Eghan, M.J., Kpiebaareh, M.Y., Tandoh, L., 2019. An Effective and Cost-Based Framework for a Qualitative Hybrid Data Deduplication, in: Bhatia, S.K., Tiwari, S., Mishra, K.K., Trivedi, M.C. (Eds.), Advances in Computer Communication and Computational Sciences. Springer Singapore, Singapore. volume 924, pp. 511–520. URL: http://link.springer.com/10.1007/978-981-13-6861-5_44, doi:10.1007/978-981-13-6861-5_44.

He, P., Gao, J., Chen, W., 2021. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. URL: http://arxiv.org/abs/2111.09543, doi:10.48550/arXiv.2111.09543, arXiv:2111.09543.

Huang, J., Hu, W., Bao, Z., Chen, Q., Qu, Y., 2022. Deep entity matching with adversarial active learning. The VLDB Journal URL: https://doi.org/10.1007/s00778-022-00745-1, doi:10.1007/s00778-022-00745-1.

Jiang, Z.H., Yu, W., Zhou, D., Chen, Y., Feng, J., Yan, S., 2020. ConvBERT: Improving BERT with Span-based Dynamic Convolution, in: Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 12837–12848. URL: https://proceedings.neurips.cc/paper/2020/hash/96da2f590cd7246bbde0051047b0d6f7-Abstract.html.

Kejriwal, M., Miranker, D.P., 2013. An Unsupervised Algorithm for Learning Blocking Schemes, in: 2013 IEEE 13th International Conference on Data Mining, pp. 340–349. doi:10.1109/ICDM.2013.60.

Kirielle, N., Christen, P., Ranbaduge, T., 2023. Unsupervised Graph-Based Entity Resolution

for Complex Entities. ACM Transactions on Knowledge Discovery from Data 17, 1–30. URL: https://dl.acm.org/doi/10.1145/3533016, doi:10.1145/3533016.

Leone, M., Huber, S., Arora, A., García-Durán, A., West, R., 2022. A critical re-evaluation of neural methods for entity alignment. Proceedings of the VLDB Endowment 15, 1712–1725. URL: https://doi.org/10.14778/3529337.3529355, doi:10.14778/3529337.3529355.

Li, C., Wang, G., Wang, B., Liang, X., Li, Z., Chang, X., 2023a. DS-Net++: Dynamic Weight Slicing for Efficient Inference in CNNs and Vision Transformers. IEEE Transactions on Pattern Analysis and Machine Intelligence 45, 4430–4446. URL: https://ieeexplore.ieee.org/abstract/document/9842348/authors#authors, doi:10.1109/TPAMI.2022.3194044.

Li, Y., Li, J., Suhara, Y., Doan, A., Tan, W.C., 2020. Deep entity matching with pre-trained language models. Proceedings of the VLDB Endowment 14, 50–60. URL: http://doi.org/10.14778/3421424.3421431, doi:10.14778/3421424.3421431.

Li, Y., Li, J., Suhara, Y., Doan, A., Tan, W.C., 2023b. Effective entity matching with transformers. The VLDB Journal URL: https://link.springer.com/10.1007/s00778-023-00779-z, doi:10.1007/s00778-023-00779-z.

Liu, Y., Lapata, M., 2019. Text Summarization with Pretrained Encoders, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China. pp. 3730–3740. doi:10.18653/v1/D19-1387.

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., 2018. Deep Learning for Entity Matching: A Design Space Exploration, in: Proceedings of the 2018 International Conference on Management of Data, Houston, TX, USA. p. 16. doi:10.1145/3183713.3196926.

Nie, H., Han, X., He, B., Sun, L., Chen, B., Zhang, W., Wu, S., Kong, H., 2019. Deep Sequence-to-Sequence Entity Matching for Heterogeneous Entity Resolution, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, ACM, Beijing China. pp. 629–638. URL: https://dl.acm.org/doi/10.1145/3357384.3358018, doi:10.1145/3357384.3358018.

Paganelli, M., Buono, F.D., Baraldi, A., Guerra, F., 2022. Analyzing how BERT performs entity matching. Proceedings of the VLDB Endowment 15, 1726–1738. URL: https://dl.acm.org/doi/10.14778/3529337.3529356, doi:10.14778/3529337.3529356.

Shao, J., Wang, Q., Lin, Y., 2019. Skyblocking for entity resolution. Information Systems 85, 30–43. URL: https://linkinghub.elsevier.com/retrieve/pii/S0306437918304770, doi:10.1016/j.is.2019.06.003.

Teofili, T., Firmani, D., Koudas, N., Martello, V., Merialdo, P., Srivastava, D., 2022. Effective Explanations for Entity Resolution Models, in: 2022 IEEE 38th International Conference on Data Engineering (ICDE), pp. 2709–2721. doi:10.1109/ICDE53745.2022.00248.

Tu, J., Fan, J., Tang, N., Wang, P., Chai, C., Li, G., Fan, R., Du, X., 2022. Domain Adaptation for Deep Entity Resolution, in: Proceedings of the 2022 International Conference on Management of Data, Association for Computing Machinery, New York, NY, USA. pp. 443–457. URL: https://doi.org/10.1145/3514221.3517870, doi:10.1145/3514221.3517870.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is All you Need, in: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp. 5998–6008.

Wang, Q., Cui, M., Liang, H., 2016. Semantic-Aware Blocking for Entity Resolution. IEEE Transactions on Knowledge and Data Engineering 28, 166–180. URL: http://ieeexplore.ieee.org/document/7202898/, doi:10.1109/TKDE.2015.2468711.

Wang, Z., Sisman, B., Wei, H., Dong, X.L., Ji, S., 2020. CorDEL: A Contrastive Deep Learning Approach for Entity Linkage, in: 2020 IEEE International Conference on Data Mining (ICDM), pp. 1322–1327. URL: https://ieeexplore.ieee.org/abstract/document/9338287, doi:10.1109/ICDM50108.2020.00171.

Yan, C., Chang, X., Luo, M., Liu, H., Zhang, X., Zheng, Q., 2022. Semantics-Guided Contrastive Network for Zero-Shot Object detection. IEEE Transactions on Pattern Analysis and Machine In-

telligence , 1–1URL: https://ieeexplore.ieee.org/abstract/document/9669022/authors#authors, doi:10.1109/TPAMI.2021.3140070.

Yao, D., Gu, Y., Cong, G., Jin, H., Lv, X., 2022. Entity Resolution with Hierarchical Graph Attention Networks, in: Proceedings of the 2022 International Conference on Management of Data, Association for Computing Machinery, New York, NY, USA. pp. 429–442. URL: https://doi.org/10.1145/3514221.3517872, doi:10.1145/3514221.3517872.

Zhang, D., Li, D., Guo, L., Tan, K.L., 2020. Unsupervised Entity Resolution with Blocking and Graph Algorithms. IEEE Transactions on Knowledge and Data Engineering , 1–1doi:10.1109/TKDE.2020.2991063.

Zhu, L., Ghasemi-Gol, M., Szekely, P., Galstyan, A., Knoblock, C.A., 2016. Unsupervised Entity Resolution on Multi-type Graphs, in: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (Eds.), The Semantic Web – ISWC 2016, Springer International Publishing, Cham. pp. 649–667. doi:10.1007/978-3-319-46523-4_39.