A Graph-based Blocking Approach for Entity Matching Using Contrastively Learned Embeddings

John Bosco Mugeni
University of Tsukuba
Ibaraki prefecture
Tsukuba, Japan
bosco@kde.tsukuba.ac.jp

Toshiyuki Amagasa
University of Tsukuba
Ibaraki prefecture
Tsukuba, Japan
amagasa@cs.tsukuba.ac.jp

ABSTRACT

Data integration is considered a crucial task in the entity matching process. In this process, redundant and cunning entries must be identified and eliminated to improve the data quality. To archive this, a comparison between all entities is performed. However, this has quadratic computational complexity. To avoid this, 'blocking' limits comparisons to probable matches. This paper presents a k-nearest neighbor graph-based blocking approach utilizing state-of-the-art context-aware sentence embeddings from pre-trained transformers. Our approach maps each database tuple to a node and generates a graph where nodes are connected by edges if they are related. We then invoke unsupervised community detection techniques over this graph and treat blocking as a graph clustering problem. Our work is motivated by the scarcity of training data for entity matching in real-world scenarios and the limited scalability of blocking schemes in the presence of proliferating data. Additionally, we investigate the impact of contrastively trained embeddings on the above system and test its capabilities on four data sets exhibiting more than 6 million comparisons. We show that our block processing times on the target benchmarks vary owing to the efficient data structure of the k-nearest neighbor graph. Our results also show that our method achieves better performance in terms of F1 score when compared to current deep learning-based blocking solutions.

CCS Concepts

•**Data Integration** → **Entity matching;** *transformer embeddings;* k-nearest neighbor (k-NN) graph algorithm; Graph based-blocking;

Keywords

Entity matching, Graph based-blocking, contrastive learning

1. INTRODUCTION

Entity matching (EM) is the task of finding co-referring entities among heterogeneous data sources (e.g., e-commerce websites and databases) [11]. For instance, the first and last tuples of Table 1 and 2 are said to co-refer. Normally, entities to be considered can be mentions of people, places, publications, citations, or even customer products as in our example. However, most of the problems of EM present themselves during integration scenarios, where data must be merged to obtain a holistic view. At the least, records to be merged may follow an identical schema but no unique global identifier. Since EM algorithms are sensitive to data quality, the task becomes even more challenging when we have noisy data, spelling variations or errors, missing values, etc. At worst, a full pairwise comparison must be performed between entries to be matched, i.e., comparing entries of one database against that of another. This comparison exhibits a quadratic growth $O(n^2)$ or complexity in the presence of proliferating data and can be time-consuming [21]

To alleviate the alluded quadratic complexity, blocking is introduced for the efficient execution of EM [21]. Blocking aims to limit the comparison of entities to those that are likely to match. Figure 1 highlights this concept. In this paper, we draw attention to blocking, which has been a hot topic among many research disciplines.

Typically, there are three approaches for blocking; rule-based, learning-based, and cluster-based blocking. Let us assume relational tables A and B with an identical schema. We say tuple $a \in A$ co-refers if it identifies with tuple $b \in B$. In the rule-based blocking methods, we assume a set of handcrafted rules defined over different attributes and measure the similarity between tuples a and b as the weighted sum of the similarity scores. Thus, we group highly similar groups of tuples as a block. However useful, rule-based methods can be labour-intensive and require lots

Table 1: e-commerce product data source A.

category	brand	model no.	price
garden - general	d-link	dcs-1100	99.82
furniture	3m	fr530cb	67.88
stationery & machinery	brother	dk2113	64.88

Table 2: e-commerce product data source B.

category	brand	model no.	price
footrests	3m #	fr530cb #	67.34
file folder labels	avery	5029	14.2
surveillance cameras	d-link	dcs-1100	99.82

of domain knowledge when sophisticated rules need to be encoded.

In the next category, the learning-based methods, we exploit a supervised machine learning approach in which a matching model is learned from training data consisting of pairs of matching tuples along with corresponding labels. By training the model based on the training data, the model learns the features (relational attributes) that effectively match the tuples, thereby allowing us to classify new tuples into matching blocks. The downsides of this approach are that labels are not always available, and learning methods require lots of training data, which may be hard to obtain in some domains. Moreover, it implicitly assumes that the tendency of the training data remains the same for the future incoming data to be classified, but, in real applications, such tendency may vary according to different reasons in the real world.

The last category, the unsupervised methods, rely on unsupervised classification methods such as clustering. They partition a set of tuples into a distinct set of clusters according to the user-specified similarity such that tuples within the same cluster are likely to refer to the same real-world entities. This approach exhibits one major advantage: no labelled training data is required, addressing the drawback of the learning-based method. However, one tricky aspect of cluster-based blocking techniques is that they suffer from long execution times when dealing with large databases and are memory-intensive depending on the clustering algorithm.

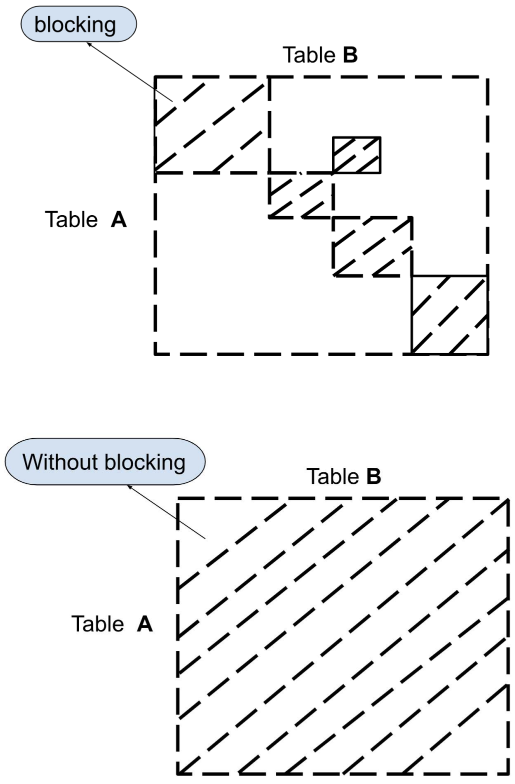

Figure 1: An illustration of matching required for EM. Blocking limits the number of comparisons (left), while exhaustive comparisons are required if we perform EM naively (right).

In the meantime, embedding techniques for text representation have been intensively studied since the emergence of word2vec [24]. More recently, modern works on contextual embedding technologies have been proposed by exploiting transformer architectures [27] that implement self-attention mechanisms to produce powerful contextual sentence representations. These sentence representations take into account the context of surrounding words, unlike traditional static word embeddings, showing better performance in various downstream tasks. One such application in data integration is to exploit sentence embeddings for EM tasks, and there have been many works in this context [2]. Basically, tuple and sentence embedding methods map tuples into their vector representations, followed by clustering over the vectors for blocking the matching candidates.

The work by Azzalini [1] is one of the state-of-the-art methods in cluster-based blocking, in which the semantic aspects of a database are exploited using traditional deep learning solutions coupled with traditional clustering methods. Nevertheless, it suffers from a long execution time when dealing with large databases. For this reason, improving the efficiency of blocking while maintaining accuracy is a major challenge.

Recently, another line of work has emerged involving contrastive learning as a strategy to tune the representation space (i.e., the embedding space) [10] of pre-trained language models. The key ingredient behind such techniques is to alter the embedding space such that positive pairs appear closer in the representation space and further from the negative pairs. In the context of EM, pairs may contain words with lexico-semantic variants which may result in a similar grouping of non-matching entities, which in turn, may affect the quality of blocking.

To this end, we propose a novel unsupervised graph-based blocking method for EM. Unlike [1], we employ pre-trained language models based on the transformer architecture [27] to generate context-aware sentence representations. Further, we exploit representation learning via contrastive learning to leverage a more uniform representation space. Instead of applying clustering to the original or reduced vector representations of tuples, we generate a k-nearest neighbour (k-NN) graph by 1) mapping each tuple to a node; and 2) generating an edge if a node is within the k-nearest neighbour from another node in terms of the similarity in the vectors space. Having constructed a k-NN graph, we apply clustering (or community detection) to it for generating blocks. Thus, we avoid expensive clustering over massive vectors, achieving faster execution of blocking.

The main contributions of our paper are as follows:

- We propose a graph-based blocking technique predicated on the k-nearest neighbour (k-NN) graph algorithm for EM.
- We leverage readily available context-aware sentence embeddings from four pre-trained language models,

namely BART [18], ReviewBERT [30], RoBERTa [31], along with DeBERTa [15] and exploit these embeddings for unsupervised blocking on EM tasks for both structured and unstructured datasets.

- We show that by leveraging recent state-of-the-art advances in NLP such as contrastive learning, the blocking performance can improve significantly.
- We conduct intensive experiments using four real-world datasets. The experimental results reveal that matching entities over a k-NN graph transcends the existing deep learning-based cluster blocking solution in terms of time required to perform EM on large datasets and show significant improvement on the unstructured versions of the datasets, especially iTunes-Amazon, where we record $\Delta F - score$ of 21.7 %.

The rest of the paper is organised as follows. Section 2 presents an overview of related works. Section 3 introduces our proposed approach. Section 4 introduces the methodology for feature extraction. Section 5 introduces the evaluation datasets and experimental setup. We then compare embeddings from four pre-trained transformers and present experimental results on the selected datasets in section 6. Finally, in section 7 we conclude the paper and offer insights for future works.

1.1 Problem Statement

Let $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b1, b2, \ldots, b_m\}$ be relational tables (or relations) with an identical schema. We also assume that the tuples in both relations refer to their corresponding real-world entities. The *entity matching* (EM) is to find all matching pairs of tuples:

$$\{(a_i, b_j) \mid a_i \in A \wedge b_j \in B\}$$

such that a_i and b_j refer to the same real-world entity. A naive execution of EM requires $O(|A| \cdot |B|)$ of comparisons, which grows quadratically according to the size of relations.

To alleviate the excessive comparisons required to perform EM, a blocking scheme has been studied. Blocking is to partition the candidate pairs into k non-overlapping partitions called *blocks* so that the tuples in the same block are likely to match. In other words, a blocking scheme BS is a mapping from the pairs of tuples to the *blocking keys* K.

$$BS : A \times B \rightarrow K$$

Having generated the blocks, ordinary matching is performed within each block, thereby avoiding unnecessary comparisons.

2. RELATED WORK

Earlier studies adopted rule-based solutions and these include but are not limited to; *standard blocking* [7], *sorted neighbourhood blocking* [16], *Q-gram blocking* [3], *suffix blocking* [28], and *canopy blocking* [19]. Normally, these rule-based solutions employ a special function to map candidate pairs of tuples to a blocking key value (BKV), which encodes information about how a tuple will be assigned to a potential block. In those works, problems arise when datasets contain long, dirty, or noisy text, sometimes missing values or spelling errors. Additionally, some of the solutions such as *sorted neighbourhood blocking* are sensitive to parameters, i.e., the sliding window size within which to search for matching tuples.

To avoid these issues, previous works such as [8] and [20] employed distributed word representations to capture similarities between tuples instead. Both methods, whose basis is the recurrent neural network (RNN), require labelled data, which, in some domains, are difficult or costly to acquire. As shown by the experiments of [20], training an RNN to produce distributed representations or sentence embeddings requires significant computational efforts.

Later, [1] devised an unsupervised EM system based on the InferSent algorithm by Facebook, an RNN implementing a bi-LSTM encoder, to generate semantic and syntactic sentence representations of 4,096 dimensions. [1] built InferSent from scratch on the Stanford Natural Language Inference Corpus (SLNI) [6] and used the learned parameters of the model, whose knowledge could be transferred, to construct semantically similar sentence representations on out-of-sample data. Finally, they cluster these sentence representations into similar blocks based on similarity using conventional clustering algorithms such as DBSCAN, K-NN, birch, and hierarchical agglomerative clustering. To deal with high dimensionality, they apply dimensionality reduction to a low dimensional space using either PCA [12] or t-SNE [26], to ease the clustering.

Problems of existing approaches can be summarized as follows: (i) Clustering is resource-intensive, particularly when dealing with large datasets. Particularly, it uses more time and memory to block large datasets (ii) When reducing the dimensionality of vectors, t-SNE performed better than PCA, but it suffers scalability issues with large datasets. (iii) InferSent uses static word embeddings to produce sentence representations that do not consider the context of surrounding words.

Compared to the works mentioned above, our work is different as follows. First, we propose an unsupervised framework for blocking that relies on syntactic and semantic information and the context of surrounding words by exploiting the latest pre-trained language models. Second, instead of clustering vector representations of tuples, we map vectors to a k-nearest neighbour (k-NN) graph coupled with community detection algorithms for blocking, thereby addressing the bottlenecks of blocking large datasets. For further reading on the subject we refer readers to [14, 17, 9] and references therein.

3. THE PROPOSED METHOD

3.1 An Overview of The Pipeline

Our proposal consists of contextual embeddings from transformers, a k-nearest neighbour (k-NN) graph, and community detection algorithms which we reference in Figure 2.

The system takes as input a couple of relations

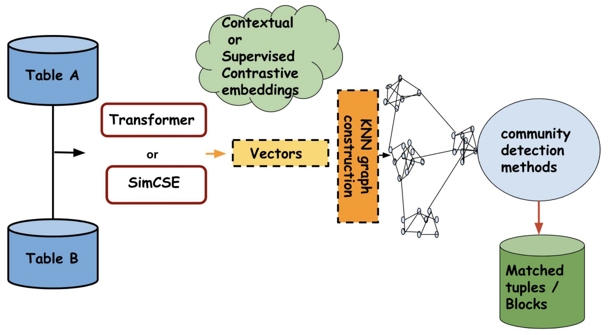

Figure 2: An EM pipeline coupled with a modern deep learning approach.

$A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ following an identical schema, where a_i or b_j represent tuples. First, for each tuple, we concatenate the attribute values into a string. This makes our approach schema-agnostic, i.e., we can also apply our method to those relations with different schemas. Next, the generated set of strings is input to a pre-trained language model based on the transformer architecture. Thus, we map each tuple to vectors in a contextual embedding space $S = \{s_1, s_2, \ldots, s_{m+n}\} \subseteq \mathbb{R}^n$. Thus, using this embedding space of choice, the embedding vectors can be input to the k-NN graph construction algorithm. Lastly, we invoke community detection algorithms such as the Louvain method and Leiden method on the k-NN graph. The resulting output is a set of matching tuples or blocks containing similar tuples.

In this work, we rely on off-the-shelf pre-trained language models and dimensionality reduction methods. So, we briefly discuss in the following sections pre-trained language models for embedding tuples, followed by k-NN graph construction and clustering over k-NN graph using community detection methods.

3.2 Pre-trained Language Models for Embedding Tuples

Our work relies on pre-trained embeddings from transformer models hosted by the HuggingFace transformers library [29]. More precisely, the models used in this work include BART [18], ReviewBERT [30], RoBERTa [31], and DeBERTa [15]. Additionally, we include the recent simCSE (i.e., simple contrastive learning of sentence embeddings) method [10] which incorporates the ideas of both supervised and unsupervised contrastive learning as an off-the-shelf solution. These methods are mostly trained on similar corpora such as English Wikipedia and BooksCorpus and have parameters concerning the training objective function (or loss function). ReviewBERT, however, has been post-trained on a review reading comprehension dataset covering domains from laptops and restaurants.

On the other hand, simCSE leverages training signals from an NLI task (i.e., Natural Language Inference) merger between the SNLI [6] and MNLI [6] corpora (i.e., the Stanford Natural Language Inference and Multi-genre Natural Language Inference tasks). For this NLI task, "entailment" pairs are taken as positives, while "contradiction" pairs are taken as hard negatives. During training simCSE tunes the embedding space using a cosine similarity-based contrastive loss to allow for greater uniformity of the embedding space. In effect, this has the benefit of pushing similar embeddings closer to each other in the representation space and pushing the non-similar ones further apart. This unification of the embedding space might aid downstream clustering algorithms in finding optimal cluster boundaries. We will see how this compares with a traditional embedding space, i.e., a representation space whose embeddings are not well separated for the purpose of downstream clustering. In our experiments, we initialize simCSE with BERT.

Algorithm 1 k-NN graph construction.

```
Input: Embedding vectors $S$, similarity $sim(\cdot)$, $k$
Output: k-NN graph $G = (V(S), E, \omega)$
  $V(S) \leftarrow$ generate a set of node for each $s \in S$
  for each $s \in S$ do
    $T \leftarrow$ find $k$ most similar vectors in $S - \{s\}$ from $s$
    for each $t \in T$ do
      $E \leftarrow E \cup \{(V(s), V(t); sim(s,t))\}$
  end for
  return $(V(S), E, \omega)$
```

Figure 3: k-NN graph algorithm.

3.3 k-NN Graph Construction

In this step, we construct from a set of vectors a k-nearest neighbour (k-NN) graph for subsequent community detection steps. A k-NN graph $G = (V(S), E, \omega)$ is an edge-weighted graph where $V(S)$ is a set of nodes corresponding to a (reduced) embedding vector S; $E \subseteq V(S) \times V(S)$ is a set of edges; and $\omega : E \rightarrow \mathbb{R}$ is a weighting function for edges.

Specifically, we measure the similarity between two vectors $s_i, s_j \in S$ using the cosine similarity:

$$sim(s_i, s_j) = \frac{s_i \cdot s_j}{|s_i| \cdot |s_j|} \tag{1}$$

Then, we generate a k-NN graph by generating an edge between nodes s and t if t is one of the k most similar nodes from s. Algorithm 5 shows a detailed algorithm. For example, Figure 4 depicts an example of vectors, and we can generate a k-NN graph as shown in Figure 5.

3.4 k-NN Graph Clustering Using Community Detection

Having constructed a k-NN graph, we perform clustering on the graph instead of clustering vectors of tuples directly. Specifically, we employ two community detection algorithms, the Louvain method and the Leiden method.

3.4.1 Louvain method

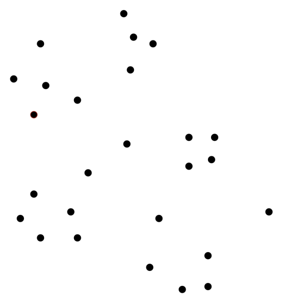

Figure 4: Scatter plot of embedding vectors.

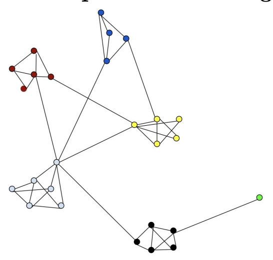

Figure 5: A k-NN graph.

The Louvain method [5] is one of the most well-known algorithms for community detection. It is an unsupervised community detection algorithm that does not require knowledge about the number or size of communities. The objective of the Louvain method is to maximize the modularity, which is calculated by the ratio of edges that connect nodes inside communities subtracted by the expected number of edges if the graph is random. The formula is as follows:

$$M = \frac{1}{2m}\sum_{ij}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j) \tag{2}$$

where m is the sum of edge weights in the graph; A_{ij} is the adjacency matrix of the weight of each edge connecting node i to j; C_i and C_j are the communities that nodes i and j belong, respectively; k_i and k_j are the sum of edge weights connected to nodes i and j, respectively; and δ is the Kronecker delta function, i.e., $(x, y) = 1$ if $x = y$ and 0 otherwise.

The algorithm of modularity-based clustering combines a local greedy optimization with a hierarchical refinement to maximize the modularity. In the local greedy optimization, each node is first assigned its own community. Then, each community is merged with one of its neighbouring communities in such a way that modularity is maximized. This process repeats until there is no further increase in modularity. In the second step, it conducts a hierarchical refinement.

Overall, the time complexity of the Louvain method is known to be $O(nlogn)$ where n is the number of nodes, which is reasonable for other clustering algorithms. It should be noted that the Louvain method exhibits some known problem called *resolution limit* when dealing with large graphs. We will see in the experimental evaluation whether it can derive practical clustering results in the context of EM.

Table 3: Dataset statistics.

Data	Domain	#Tuples	#Matches	Attr	|C|
DS	citation	2616-64263	5347	4	168
IA	music	6907-55923	132	8	386
WA	electronics	2554-22074	962	5	56
GD	citation	2616-64263	5347	4	168

3.4.2 Leiden Method

Leiden method [25] is another unsupervised community detection algorithm that extends the Louvain method for addressing some known problems. Particularly, the Louvain method can sometimes yield badly connected communities [25]. Leiden's method addresses the shortcomings of Louvain and yields well-connected communities. Unlike the Louvain method, the Leiden method does not follow a greedy approach and has three stages in its execution pipeline although it also works based on moving nodes to random neighbours. The three stages are as follows: (i) a fast local moving of nodes, (ii) a partition refinement, and (iii) aggregating the graph based on refined partitions. In this way, it guarantees well-connected components.

4. EXPERIMENT EVALUATION

To evaluate the proposed method, we conduct intensive experiments to evaluate it experimentally. Specifically, we check how does graph-based blocking compares with cluster-based blocking introduced in [1]?. We also check how do pre-trained general contextual embeddings compare with post-trained task-specific embeddings for EM?. How does the dimensionality reduction of the embedding vectors affects our graph-based blocking method?

4.1 Experimental Data

To evaluate our method, we adopt datasets from [20], which are now commonly used to evaluate entity-matching techniques. Azzalini [1] focused on datasets with a Cartesian-product size of fewer than 6 million comparisons, while we deal with in our experiments more challenging datasets with more than 6 million comparisons. We also categorize these datasets into structured or unstructured. The former means that the tuples contain attribute values that are short and atomic and include $iTunes - Amazon$, $GoogleScholar - DBLP_1$, $Walmart - Amazon$, and $GoogleScholar - DBLP_2$. On the other hand, the latter case contains tuples intelligently modified to challenge matching algorithms, containing missing values that sometimes moved to a wrong attribute, etc. These are modified versions of $iTunes - Amazon$, $GoogleScholar - DBLP$, and $Walmart - Amazon$. We summarise the characteristics of these datasets in Table 3.

4.2 Experimental Setup

In this section, we detail our workstation, instrument specifications as well the parameters of choice for the pre-trained models.

4.2.1 Experimental Environment

Our computing environment is a PC with Intel(R) Core(TM) i7-4820K quad-core CPU encompassing 48GB RAM, attached a GeForce GTX 1080 GPU (8GB memory) running Ubuntu 18.04. This accelerator is only used to construct sentence vectors beyond which it is disengaged. The pre-trained models are based on HuggingFace [29] and all associated blocking programs are implemented in Python version 3.7.6.

4.2.2 Pre-trained Model Parameters and Data Preprocessing

For each transformer-based model, we choose the attention spans to be 200 tokens in any case. We also use 32 for the batch size and employ mean-pooling as a strategy for summarising the token inputs of our text to produce contextual embeddings. As for each dataset, the missing value imputation follows the approach of [1] where, for each missing attribute value, we use a special “unk” token.

4.2.3 Entity Matching

Once the tables to be matched are passed to the EM framework, all tuples are mapped to an embedding space and a k-NN graph is built. As discussed in Section 3.3, k is an important parameter of the k-NN graph that determines the number of neighbouring vectors (nodes) to connect in the graph construction phase. We vary k between 2 and 30 and report the best performance of community detection algorithms using metrics defined in Section 4.3.

4.3 Evaluation Metrics

To assess the performance of our blocking method, we leverage commonly used metrics such as the *reduction ratio* (RR), and *pair completeness* (PC), which corresponds to recall. The RR computes how many potential matches are generated by the blocking scheme against the naive pair-wise comparison, in other words, the relative reduction of the comparison space regardless of quality, and then PC measures how many of the potential matches correspond to the known true matches and this are crucial for effectiveness. A good matching scheme should maximize both RR and PC, ranging between 0 and 1, where proximity to 1 is better. However, a trade-off between RR and PC exists. A high RR could come at the expense of a low PC and vice-versa. A high RR means that the blocking scheme removes many unnecessary comparisons while a high PC entails that generated matches contain a high proportion of true matches. In terms of formulation, RR and PC are defined as:

$$\begin{cases} RR = 1.0 - \frac{s_M + s_N}{n_M + n_N} \\ PC = \frac{s_M}{n_M} \end{cases}$$

where n_M and n_N are the number of matched and non-matched record pairs in the absence of a blocking scheme, respectively; and s_M and s_N are the number of matched and non-matched record pairs in the presence of a blocking scheme, respectively. Sometimes, it is possible to combine information about these metrics in what we term the F-score or the F1-score (the harmonic mean). The F1 score is given as:

$$\alpha = 2 \cdot \frac{RR \cdot PC}{RR + PC} \tag{3}$$

In addition to these metrics, we consider the blocking time (i.e., the time it takes to execute the blocking in seconds) of our scheme. This is especially crucial in order to elaborate on the practicality of blocking systems in real-world scenarios.

5. EXPERIMENTAL RESULTS

In this section, we are interested in comparing the performance of our k-NN graph-based blocking to the existing cluster-based blocking of [1], particularly, on the datasets defined in Table 3. They exhibit more than 6 million comparisons. Tables 4–7 show the performance of embeddings of choice coupled with the best community detection algorithm in each scenario. Note that ‘architecture’ is the pre-trained model used to construct sentence embeddings; ‘$algo_{best}$’ is the best blocking algorithm; ‘*emb*’ is the time taken by the architecture of choice to construct sentence embeddings; ‘bk’, short for blocking, is the time; ‘$algo_{best}$’ took to produce the blocks given the k-NN graph; ‘$total'$’ is the time taken by a scheme to complete EM; ‘$F1$’ is used to score the blocking performance, and ‘*dnf*’ is short for ‘did not finish computing’. Also, for each scenario, we include the previous best F1 score in bold (black font) and our current best F1 in red font.

Table 4: GoogleScholar-DBLP-1.

method	algo$_{best}$	emb’$_{sec}$	bk$_{sec}$	total$_{sec}$	F1
R-BERT	l’vian	111.4	203.8	315.2	**93.5**
DeBERTa	l’vian	439.0	215.3	654.3	89.5
RoBERTa	l’vian	370.9	226.2	597.0	90.7
BART	l’vian	451.2	189.0	640.2	92.4
RNN	birch	2563.0	dnf	dnf	dnf
SimCSE	l’vian	110.4	156.8	267.2	97.8
R-BERT$_d$	l’vian	131.6	210.3	342.0	**86.8**
DeBERTa$_d$	l’vian	463.2	211.0	674.2	80.2
RoBERTa$_d$	l’vain	380.6	463.7	844.3	73.8
BART$_d$	l’vian	501.0	194.5	695.4	83.5
SimCSE$_d$	l’vian	90.8	245.5	336.3	93.8

Note: R-BERT is a short form for ReviewBERT, l’vian for louvian and l’den for leiden.

Table 4 shows results on GoogleScholar-DBLP-1. We note that contrastively trained embeddings from SimCSE have a significant impact on the quality of blocking. In particular, we record Δ of +4.3 % on the structured version of the benchmark. On the unstructured version, SimCSE embeddings increase the F1 score by a 7.0 % margin, albeit at a slight degradation of 4.03 % from the previous 7.17 % (i.e., the difference between R-BERT and R-BERT$_d$).

On the structured iTunes-Amazon dataset (Table 5), SimCSE yields the best performance **92.8 %**. Other

Table 5: iTunes-Amazon.

method	algo_{best}	emb'_{sec}	bk_{sec}	total_{sec}	F1
R-BERT	l'vian	**91.8**	461.8	553.6	85.2
DeBERTa	l'vian	311.2	557.2	868.4	89.2
RoBERTa	l'vian	253.6	**58.1**	**311.7**	89.7
BART	l'vian	324.0	433.6	757.6	**91.7**
RNN	birch	2329.8	dnf	dnf	dnf
SimCSE	l'vian	64.5	160.9	225.4	92.8
R-BERT_d	l'den	127.7	**328.2**	**455.9**	56.2
DeBERTa_d	l'den	470.0	607.8	1077.8	56.4
RoBERTa_d	l'vian	391.5	368.2	759.7	64.0
BART_d	l'den	642.9	347.5	990.4	**68.0**
SimCSE_d	l'den	125.8	164.4	290.2	89.7

transformer embeddings yield acceptable performance with BART-Louvain coming in at second place with a 91.7 % F1 score. However, the performance gain on this benchmark is minimal perhaps due to the task being well-structured. Interestingly, the unstructured version reveals a different scenario for the traditional embeddings. We suspect that missing attribute values, typographical errors, punctuation symbols, etc, will yield inferior blocking performance. However, the SimCSE contrastively trained embeddings yield a suitable embedding space in which positive and negative embeddings are well separated. As a result, we witness better downstream clustering performance with Δ of **21.7 %**.

Table 6: Walmart-Amazon.

method	algo_{best}	emb'_{sec}	bk_{sec}	total_{sec}	F1
R-BERT	l'vian	**39.1**	58.9	**98.0**	**91.6**
DeBERTa	l'den	134.5	47.3	181.9	90.1
RoBERTa	l'vian	111.3	58.2	168.6	89.7
BART	l'vian	132.1	48s	180.1	90.2
RNN	birch	835.9	**12.6**	848.4	90.1
SimCSE	l'vian	42.1	36.46	78.5	92.5
R-BERT_d	l'vian	**52.9**	45.6	**98.51**	90.3
DeBERTa_d	l'vian	162.2	48.08	210.2	**90.5**
RoBERTa_d	l'den	283.2	56.28	339.4	87.6
BART_d	l'vian	160.5	**30.4**	190.9	89.4
SimCSE_d	l'vian	47.3	32.98	80.3	92.5

Table 6 shows results on the Walmart-Amazon. All transformer embeddings perform well on average when combined with community detection algorithms, including the existing InferSent based-blocking system, dubbed cluster-based blocking, implementing birch as the best clustering algorithm. However, we note that cluster-based blocking requires more time to yield embeddings even though this time it processes them faster. Overall, in both scenarios, i.e., structured and unstructured data, SimCSE obtains the best performance, i.e., **92.5 %** and **92.5 %** respectively.

Lastly on GoogleScholar-DBLP-2, Table 7, we observe again that SimCSE achieves the best F1 score of **95.6 %** representing a Δ of 4.0 %. Note that processing the blocks requires more time in the case of the InferSent-based blocking system. This is due to the bottlenecks introduced by t-SNE

Table 7: GoogleScholar-DBLP-2.

method	algo_{best}	emb'_{sec}	bk_{sec}	total_{sec}	F1
R-BERT	l'vian	**108.0**	283.0	**391.0**	**91.6**
DeBERTa	l'vian	293.0	283.3	576.3	89.1
RoBERTa	l'vian	328.6	**229.8**	558.4	89.6
BART	l'vian	289.8	261.4	551.2	91.4
RNN	birch	2787.1	dnf	dnf	dnf
SimCSE	Louvain	127.8	173.4	301.2	95.6

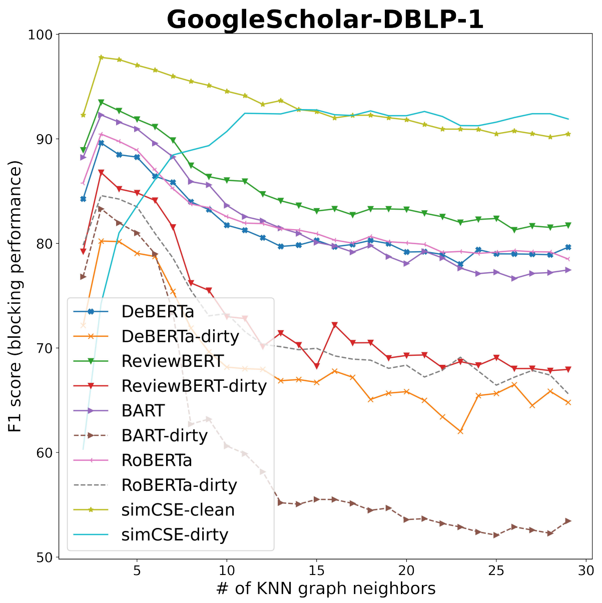

Figure 6: GoogleScholar-DBLP-1.

and the internal operation of the standard birch algorithm that comprises an expensive clustering step, which might be costly to run on large datasets as in this case.

5.1 Neck and Neck Transformer Embedding Performance

Next, we compare off-the-rack embeddings from the choice of transformers as a function of parameter k. We seek to establish the behaviour and performance of post-trained transformer embeddings versus general transformer embeddings for the EM task using our graphical approach. All other transformers except ReviewBERT are considered to produce general embeddings. Figures 6–8 compare the embedding performance as a function of k on the datasets outlined in Table 3.

We summarize our main findings as follows;

- Experiments show that contrastively learned embeddings perform much better than traditional embeddings across the board, i.e., for both structured and unstructured datasets. On iTunes-Amazon, the performance on the unstructured version improves by **21.7 %**.
- Across all datasets, we note that as k increases, the blocks generated by k-NN graph-based blocking start

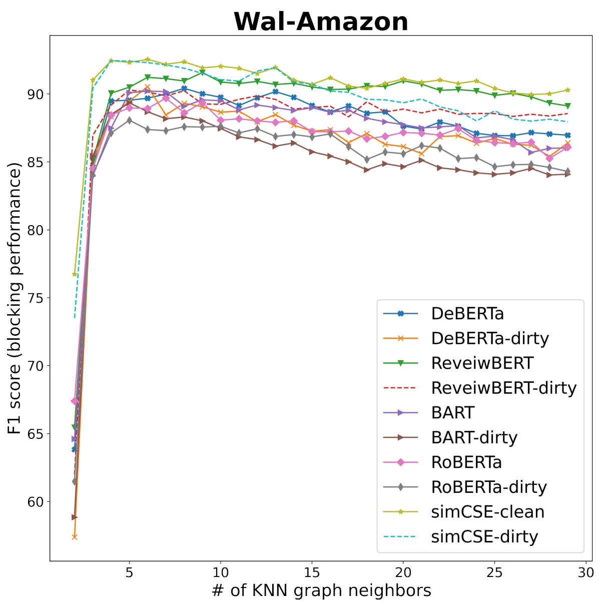

Figure 7: Walmart-Amazon.

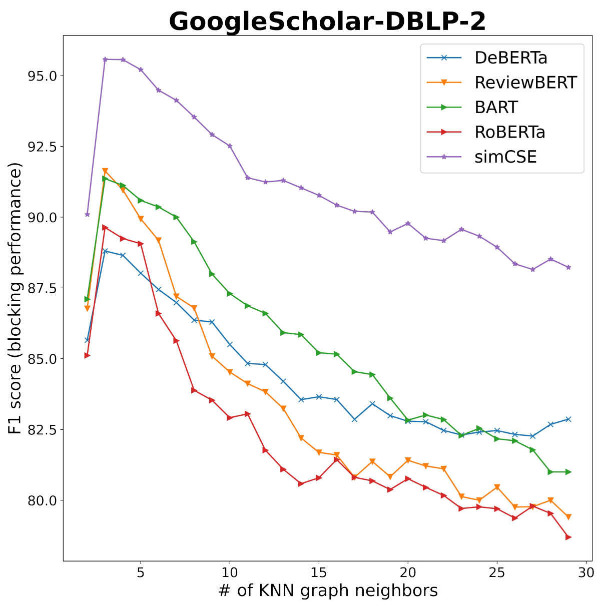

Figure 8: GoogleScholar-DBLP-2.

to incorporate more false positives, leading to a gradual decline in the F1 score for the traditional embeddings. This decline is interestingly stable for the contrastively trained embeddings across the board. In most settings, we find low values of k seemed to favour our blocking approach in general with simCSE and somewhat low to mid-range for the traditional embeddings.

- On all datasets, simCSE's contrastively trained embeddings lead performance, while ReviewBERT frequents second place. This demonstrates the importance of the contrastive objective in tuning the representation space for downstream clustering.

6. CONCLUSIONS

This paper has presented an unsupervised framework for blocking for the EM domain based on a k-NN graph and coins blocking as a graph clustering problem. Furthermore, we have developed our framework to utilize contextual embeddings from transformers and the simCSE framework for contrastively tuned embeddings, making it a competitive approach over existing unsupervised deep learning-based blocking approaches for EM. From a high-level perspective, our graph-based blocking framework improves the state-of-the-art in terms of execution time when processing large datasets. On the experimental datasets, block processing times vary between 59 seconds to 461 seconds. In comparison, the existing approach requires more time to perform EM.

From a high-level perspective, our graph-based blocking framework improves the state-of-the-art in terms of blocking times when dealing with large datasets. On the experimental datasets, block processing times vary between 30 seconds to 557 seconds. In comparison, the existing approach requires more time to perform EM (i.e., time for block processing). We also note that fine-tuned embeddings from simCSE have competitive performance in most scenarios over general embeddings, especially on the dirt data sets.

In the future, we will study how to combine our blocking system with advances in contrastive learning of supervised embeddings and incorporate other major works [13, 4, 23, 22]. Also, we plan to explore avenues such as merging our blocking system with a supervised entity matching system.

7. REFERENCES

[1] F. Azzalini, S. Jin, M. Renzi, and L. Tanca. Blocking techniques for entity linkage: A semantics-based approach. *Data Science and Engineering*, 6(1):20–38, 2020.

[2] N. Barlaug and J. A. Gulla. Neural networks for entity matching: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(3):1–37, 2021.

[3] R. Baxter, P. Christen, and C. Epidemiology. A comparison of fast blocking methods for record linkage. *Proc. of ACM SIGKDD'03 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 25–27, 2003.

[4] M. Belcaid, C. Arisdakessian, and Y. Kravchenko. Taming dna clustering in massive datasets with slymfast. *ACM SIGAPP Appl. Comput. Rev.*, 22(1):15–23, 2022.

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 10:P10008, 2008.

[6] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom. e-snli: Natural language inference with

natural language explanations. In *Advances in Neural Information Processing Systems*, pages 9560–9572, 2018.

[7] P. Christen. The data matching process. In *Data matching*, pages 23–35, 2012.

[8] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018.

[9] C. Fu, X. Han, L. Sun, B. Chen, W. Zhang, S. Wu, and H. Kong. End-to-end multi-perspective matching for entity resolution. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 4961–4967, 2019.

[10] T. Gao, X. Yao, and D. Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, 2021.

[11] H. Garcia-Molina. Entity resolution: Overview and challenges. In P. Atzeni, W. Chu, H. Lu, S. Zhou, and T.-W. Ling, editors, *Conceptual Modeling*, pages 1–2, 2004.

[12] F. L. Gewers, G. R. Ferreira, H. F. D. Arruda, F. N. Silva, C. H. Comin, D. R. Amancio, and L. D. F. Costa. Principal component analysis: A natural approach to data exploration. *ACM Comput. Surv.*, 54(4):1–34, 2021.

[13] S. Ghosh, S. Maji, and M. S. Desarkar. Effective utilization of labeled data from related tasks using graph contrastive pretraining: Application to disaster related text classification. In *Proceedings of the 37th ACM SIGAPP Symposium on Applied Computing*, pages 875–878, 2022.

[14] R. D. Gottapu, C. Dagli, and B. Ali. Entity resolution using convolutional neural network. *Procedia Computer Science*, 95:153–158, 2016.

[15] P. He, X. Liu, J. Gao, and W. Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[16] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 127–138, 1995.

[17] N. Kooli, R. Allesiardo, and E. Pigneul. Deep learning based approach for entity resolution in databases. In *Asian conference on intelligent information and database systems*, pages 3–12, 2018.

[18] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[19] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 169–178, 2000.

[20] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34, 2018.

[21] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(2):1–42, 2020.

[22] B. A. Pijani, A. Imine, and M. Rusinowitch. Inferring attributes with picture metadata embeddings. *ACM SIGAPP Appl. Comput. Rev.*, 20(2):36–45, 2020.

[23] M. M. Rahman and A. Takasu. Exploiting knowledge graph and text for ranking entity types. *ACM SIGAPP Appl. Comput. Rev.*, 20(3):35–46, 2020.

[24] T. Shi and Z. Liu. Linking glove with word2vec. *arXiv preprint arXiv:1411.5595*, 2014.

[25] V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):1–12, 2019.

[26] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11):2579–2605, 2008.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.

[28] Y. Warke. Suffix array blocking for efficient record linkage and de-duplication in sliding window fashion. In S. C. Satapathy, V. Bhateja, and A. Joshi, editors, *Proceedings of the International Conference on Data Engineering and Communication Technology*, pages 57–65, 2017.

[29] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

[30] H. Xu, B. Liu, L. Shu, and P. Yu. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2324–2335, 2019.

[31] L. Zhuang, L. Wayne, S. Ya, and Z. Jun. A robustly optimized bert pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, 2021.

ABOUT THE AUTHORS:

John Bosco Mugeni is a first-year doctoral student in systems and information engineering at the University of Tsukuba, studying under professor Toshiyuki Amagasa. Before embarking on the doctoral program, he received a master's degree in data science from the University of Rwanda (2020), and another in systems and information engineering at the University of Tsukuba (2022). Currently, he is focusing on database research covering the application of machine learning in what has presently come to be called data integration.

Toshiyuki Amagasa received B. E., M. E., and Ph. D from the Department of Computer Science, Gunma University in 1994, 1996, and 1999, respectively. He is currently a full professor at the Center for Computational Sciences, University of Tsukuba. His research interests cover database systems, data mining, and database application in scientific domains. He is a board member of DBSJ, a senior member of IEICE and IEEE, and a member of IPSJ and ACM.