SC-Block: Supervised Contrastive Blocking Within Entity Resolution Pipelines

Alexander Brinkmann[1](✉), Roee Shraga[2], and Christina Bizer[1]

[1] University of Mannheim, 68131 Mannheim, Germany
{alexander.brinkmann,christian.bizer}@uni-mannheim.de
[2] Worcester Polytechnic Institute, Worcester, USA
rshraga@wpi.edu

Abstract. Millions of websites use the schema.org vocabulary to annotate structured data describing products, local businesses, or events within their HTML pages. Integrating schema.org data from the Semantic Web poses distinct requirements to entity resolution methods: (1) the methods must scale to millions of entity descriptions and (2) the methods must be able to deal with the heterogeneity that results from a large number of data sources. In order to scale to numerous entity descriptions, entity resolution methods combine a blocker for candidate pair selection and a matcher for the fine-grained comparison of the pairs in the candidate set. This paper introduces SC-Block, a blocking method that uses supervised contrastive learning to cluster entity descriptions in an embedding space. The embedding enables SC-Block to generate small candidate sets even for use cases that involve a large number of unique tokens within entity descriptions. To measure the effectiveness of blocking methods for Semantic Web use cases, we present a new benchmark, WDC-Block. WDC-Block requires blocking product offers from 3,259 e-shops that use the schema.org vocabulary. The benchmark has a maximum Cartesian product of 200 billion pairs of offers and a vocabulary size of 7 million unique tokens. Our experiments using WDC-Block and other blocking benchmarks demonstrate that SC-Block produces candidate sets that are on average 50% smaller than the candidate sets generated by competing blocking methods. Entity resolution pipelines that combine SC-Block with state-of-the-art matchers finish 1.5 to 4 times faster than pipelines using other blockers, without any loss in F1 score.

Keywords: Identity Resolution · Blocking · schema.org · Benchmarking · Supervised Contrastive Learning

A. Meroño Peñuela et al. (Eds.): ESWC 2024, LNCS 14664, pp. 121–142, 2024.
https://doi.org/10.1007/978-3-031-60626-7_7

1 Introduction

The Web Data Commons (WDC) project regularly extracts schema.org data from the Common Crawl[1] [5]. The extraction[2] from the October 2022 version of the Common Crawl has shown that 2.5 million websites (hosts) use the schema.org vocabulary to annotate product offers, 1.2 million websites annotate information about local businesses such as addresses and opening hours, and 50,000 websites annotate job postings. In total, 502 million records describing products and 55 million records describing local businesses were extracted from the Common Crawl. Due to the shallow coverage of the Common Crawl, the extracted data only represents a fraction of the schema.org data available on the Semantic Web. Applications seeking to integrate schema.org data from the Semantic Web for use cases such as product recommendation, price comparison, or complementing knowledge graphs face two challenges in the link discovery [2,11,35] step of their data integration and cleansing pipeline: (1) they require entity resolution methods that can scale to millions of entity descriptions and (2) these methods must be able to handle the heterogeneity that arises from using data from multiple sources. A scalable blocking method is crucial for these use cases as the pairwise comparison of all records is infeasible. Blocking [9,40,49] employs a computationally inexpensive method to generate a set of candidate pairs which likely contains most matches while being as small as possible. Afterwards, a matcher derives the final set of matching pairs using a computationally more expensive and more precise matching method [30,31,33,36].

In this paper, we propose SC-Block, a blocking method that applies supervised contrastive learning to cluster records that likely describe the same real-world entity in an embedding space. In this blocking-only scenario, SC-Block is compared to different state-of-the-art blocking methods [34,39,49,51]. Additionally, SC-Block and the two most competitive blockers are combined with different state-of-the-art matching methods [6,30,31,44] to evaluate the F1 performance and runtime of complete entity resolution pipelines. The experiments are conducted using existing entity resolution benchmarks as well as WDC-Block, a large new benchmark which is introduced in this paper. Existing benchmarks that are used for evaluating blocking methods are either rather small [33], mostly Wikipedia-related [17] or rely on synthetic data [41]. None of the benchmarks uses large amounts of real-world e-commerce data originating from numerous data sources. WDC-Block fills this gap by requiring the blocking of product offers from 3,259 e-shops. In summary, this paper makes the following contributions:

1. We propose SC-Block a blocking method which applies supervised contrastive learning to position records likely describing the same real-world entity close to each other in an embedding space.

[1] https://commoncrawl.org/.

[2] https://webdatacommons.org/structureddata/2022-12/stats/schema_org_subsets.html.

2. We introduce WDC-Block a new large blocking benchmark that requires blocking schema.org product offers from the Semantic Web and features a maximal Cartesian product of 200 billion record pairs and almost 7 million unique tokens within the entity descriptions.
3. We show that SC-Block creates smaller candidate sets than state-of-the-art blocking methods leading to pipelines that execute 1.5 to 2 times faster on the smaller benchmark datasets and 4 times faster on the largest product matching task of WDC-block without negatively affecting F1 scores.
4. We relate the training time of SC-Block to the overall runtime reduction of entity resolution pipelines and show that the runtime reduction overcompensates the training time of SC-Block for larger datasets.

The paper is structured as follows. Section 2 presents the WDC-Block benchmark. Section 3 introduces SC-Block and discusses its supervised contrastive training. SC-Block and the impact of SC-Block on complete entity resolution pipelines are evaluated in Sect. 4 and in Sect. 5, respectively. Related work is discussed in Sect. 6. The new benchmark[3] and the code[4] for replicating all experiments are available online.

2 WDC-Block: A Large Product Blocking Benchmark

This section introduces WDC-Block and compares it to blocking benchmarks from the related work. WDC-Block is built by extending the WDC Products entity matching benchmark[5] [45] with additional product offers from the WDC Product Data Corpus V2020[6]. WDC Products is a multi-dimensional benchmark which supports the evaluation of matching systems along combinations of three dimensions: (1) amount of corner cases, (2) training set size, and (3) amount of unseen entities in the test set. The product offers in WDC Products and the WDC Product Data Corpus have been extracted from the Common Crawl using schema.org annotations. The product offers are clustered by product identifiers such as MPN and GTIN [45]. All offers in the same cluster describe the same real-world entity. Record pairs from the clusters are classified as corner cases if they are difficult to match for a range of baseline matchers [45]. 500 clusters are selected as *seen*. The record pairs in the *seen* clusters are split into train, validation and test sets. Afterwards, record pairs in the test set are replaced with record pairs from *unseen* clusters until the desired percentage of unseen record pairs in the test set is reached. For WDC-Block, we select the dataset containing 80% corner cases, because corner cases are challenging for entity resolution pipelines. We chose the largest training set of WDC Products as our seed training set. We choose the test set with 50% unseen record pairs. This ensures a balance between products that were part of the training set and those that were not.

[3] https://webdatacommons.org/largescaleproductcorpus/wdc-block/.

[4] https://github.com/wbsg-uni-mannheim/SC-Block/.

[5] https://webdatacommons.org/largescaleproductcorpus/wdc-products/.

[6] https://webdatacommons.org/largescaleproductcorpus/v2/index.html.

Three Sizes of WDC-Block. To match the setup of previous blocking benchmarks, we divided the WDC Products data into two separate datasets, A and B. The maximum cardinality of matching records across datasets A and B is 15. We then extended these datasets with additional randomly selected offers from the WDC Product Data Corpus V2020 to create three versions of WDC-Block: a small version (WB_{small}), a medium version (WB_{medium}), and a large version (WB_{large}). We ensure that the randomly selected records do not match existing records in the datasets to avoid introducing additional matching pairs. Through these additional records and identical train, validation and test pairs in WB_{small}, WB_{medium} and WB_{large}, we can measure the effect of significantly increased vocabulary sizes (67K to 6.9M tokens) and large Cartesian products ($A \times B$ between $2.5 \cdot 10^7$ and $2.0 \cdot 10^{11}$ pairs) on the performance of blockers. Table 1 provides statistics about the different versions of the WDC-Block benchmark and other blocking benchmarks from related work. The vocabulary size is defined as the number of unique tokens in all datasets that belong to the benchmark task. Tokens are derived from the records by serializing them into entity descriptions (see Sect. 3) and splitting the entity descriptions by whitespace.

Table 1. Statistics of the benchmark tasks.

Benchmark	Dataset	Dataset	Pos	Neg	Pos	Neg	Pos	Neg	Vocab	Cartesian
	A	B	Train	Train	Val.	Val.	Test	Test	Size	Product
WDC-Block$_{small}$	5.0k	5.0k	6.5k	10.5k	3.2k	5.3k	0.5k	4.0k	67k	$2.5 \cdot 10^7$
WDC-Block$_{medium}$	5.0k	0.2M	6.5k	10.5k	3.2k	5.3k	0.5k	4.0k	1.2M	$1.0 \cdot 10^9$
WDC-Block$_{large}$	0.1M	2.0M	6.5k	10.5k	3.2k	5.3k	0.5k	4.0k	6.9M	$2.0 \cdot 10^{11}$
Abt-Buy	1.1k	1.1k	0.6k	5.1k	0.2k	1.7k	0.2k	1.7k	7.9k	$1.2 \cdot 10^6$
Amazon-Google	1.4k	3.3k	0.7k	6.2k	0.2k	2.1k	0.2k	2.1k	7.4k	$4.5 \cdot 10^6$
Walmart-Amazon	2.6k	22.1k	0.6k	5.6k	0.2k	0.9k	0.2k	0.9k	49.2k	$5.6 \cdot 10^6$
DM$_{2M}$	2.0M	–	–	–	–	–	1.7M	–	3.8M	$2.0 \cdot 10^{12}$
BTC12-Infoboxes	1.6M	8.9M	–	–	–	–	1.5M	–	4.9M	$1.5 \cdot 10^{13}$

Comparison to other Blocking Benchmarks. Similar to WDC-Block, the Abt-Buy (A-B), Amazon-Google (A-G) and Walmart-Amazon (W-A)[7] benchmarks, which are widely used in the related work [31,33,37,43,44,49], cover e-commerce use cases. The statistics in Table 1 show that these benchmarks have much smaller Cartesian products $A \times B$ and vocabulary sizes compared to WDC-Block. Each of these benchmarks only requires blocking data from two e-shops while WDC-Block contains data from 3,259 sources. The DM$_{2M}$ benchmark [37] offers a large Cartesian product but consists of synthetic data. The benchmark is thus biased by its data generation process. Similar to WDC-Block, the blocking benchmarks introduced in [17] are based on data from the Semantic

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

Web. The largest benchmark from this paper, BTC12-Infoboxes, requires blocking data from the Billion Triples Challenge 2012 dataset which has been crawled from the LOD Cloud and infobox data from DBpedia [17]. Although the dataset BTC12-Infoboxes is larger than WDC-Block, the matches in BTC12-Infoboxes all involve entities that appear in Wikipedia. In contrast, WDC-Block contains data from a different topical domain: e-commerce data from many e-shops. In addition, the e-commerce data in WDC-Block is more recent than the Wikipedia data in BTC12-Infoboxes (2022 versus 2012).

3 SC-Block: Supervised Contrastive Blocking

There are unsupervised [16,42], self-supervised [34,49,51,53], and supervised blocking methods [4,18,39]. The supervised methods use a training set containing matching and non-matching record pairs. Most state-of-the-art matching methods require training sets, which are assembled by many entity resolution projects. The motivation behind supervised blocking is to use the available training data not only for matching but also for blocking. In the context of the Semantic Web, the necessary training data can be derived from schema.org annotations that contain identifiers like GTINs, MPNs or ISBNs. SC-Block utilizes supervised contrastive learning to position record embeddings likely describing the same real-world entity close to each other in an embedding space. Figure 1 gives an overview of SC-Block while the different steps of the method are described below.

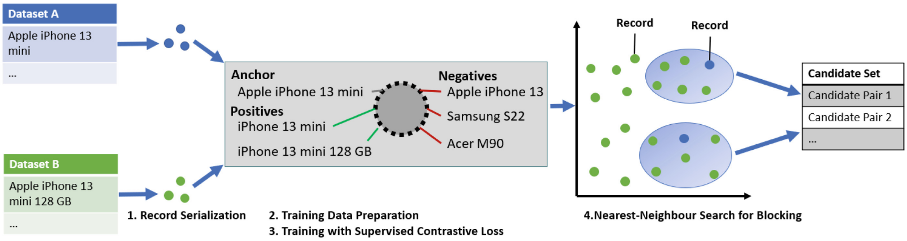

Fig. 1. Overview of the SC-Block method

(1) Record Serialization. Following recent works [31,49,51], records in datasets A and B are serialized into textual entity descriptions. Each attribute of a record is serialized as follows, "[Col] attribute_name [Val] actual_attribute_value". For the entity descriptions, all attribute serializations are concatenated. In the appendix, we evaluate the impact of adding the attribute names to the serialization.

(2) Training Data Preparation. This section explains how the training data is prepared for supervised contrastive learning and how source-aware sampling is used to reduce inter-label noise. The supervised contrastive loss requires records to share the same label if they describe the same real-world entity. The benchmark datasets provide pairs of records instead of clusters of matching records that share the same label. To identify records that refer to the same real-world entity, a correspondence graph is constructed, following the approach of Peeters and Bizer [44]. The records serve as vertices in the graph, and an edge between two vertices indicates a match between the corresponding records. A unique label is assigned to each connected component in the graph, ensuring that records describing the same real-world entity receive the same label. The labelling procedure may introduce inter-label noise as only a subset of all matches is known. Consequently, some matching records may not share the same label. During training, these matching records are treated as non-matches and are not embedded in a nearby location in the embedding space. This inter-label noise reduces the effectiveness of the embedding [12]. We apply source-aware sampling to reduce inter-label noise [44]. To achieve source-aware sampling, we create two training sets: A and B. Training set A contains all records from dataset A and the records from dataset B that share a label with a record from dataset A. Training set B contains all records from dataset B and the records from dataset A that share a unique label with a record from dataset B. During training, batches of offers are sampled from either training dataset A or B. This sampling strategy reduces the noise resulting from missing matches [44].

(3) Training with Supervised Contrastive Loss. In this section, we introduce the supervised contrastive loss and how it is applied to train effective embeddings to cluster matching records in an embedding space. The training procedure starts with a batch of N entity descriptions sampled from the prepared training data as discussed in Sect. 3. We duplicate all records so that for each record there is at least one matching record in the batch. An *encoder network* $Enc(\cdot)$ maps each entity description t to an embedding, $z = Enc(t) \in R^D$. In our experiments, $Enc(\cdot)$ is a pre-trained RoBERTa-base model [32] with $D = 768$. The record embedding z is mean pooled and normalized using the L_2 normalization. During training, we apply the supervised contrastive loss to update the parameters of the RoBERTa model. The supervised contrastive loss exploits label information of matching records by maximizing the agreement of records with the same label (Positives) and minimizing the agreement of records from different labels (Negatives) to train effective embeddings for SC-Block. Formally, supervised contrastive loss is defined as follows [29]: Given a batch of $2N$ embedded records $z = Enc(t) \in R^D$:

$$\mathcal{L} = \sum_{i \in I} \mathcal{L}_i = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{exp(z_i \cdot z_p / \mathcal{T})}{\sum_{a \in A(i)} exp(z_i \cdot z_a / \mathcal{T})} \tag{1}$$

Here, $i \in I \equiv 1...N$ is the index of an embedded record z. The index i is called an anchor embedding z_i, $P(i) \equiv \{p \in A(i) : y_p = y_i\}$ is the set of indices of all records with the same label in the batch distinct from i, and $|P(i)|$ is its cardinality. Also recall that the dot $(\cdot)$ symbol denotes the inner product,

$A(i) \equiv I \backslash i$, and $\mathcal{T} \in R^{+}$ is a scalar temperature parameter. Within a batch, each embedding becomes an anchor embedding z_i that is pulled close to all record embeddings from the same unique label while it is pushed away from all record embeddings with different labels. By setting the batch size to 1024 we ensure that each embedding is compared to many other embeddings, which is beneficial for the supervised contrastive loss [29]. Additionally, in each training epoch, different records are sampled into a batch leading to a variation in the record comparison during training. The large number of comparisons and the variation of the batches enable supervised contrastive training to learn better embeddings than pair-wise training which only uses the record pairs from the training set. In our experiments, $\mathcal{T}$ is set to 0.07 and the encoder network is trained for 20 epochs with a learning rate of $5e-5$. After training the encoder network embeds the entity descriptions from the input datasets A and B.

(4) Nearest-Neighbour Search for Blocking. SC-Block uses nearest neighbour search to create blocks of similar records, which is a common approach for blocking [49,51,53]. The nearest neighbour search exploits that similar records are close in the embedding space due to supervised contrastive training. For the search, we define dataset A, which contains fewer records than dataset B, to serve as the query table Q_A and dataset B to serve as an index table I_B. In the appendix, we evaluate the impact of switching the query and the index table. We index the embeddings z_B of I_B using FAISS [26] to allow an efficient search. For each query record $z_A \in Q_A$, the search ranks candidate records $z_B \in I_B$ based on their cosine similarity with the query record $z_A \in Q_A$. A hyperparameter k determines how many nearest neighbours are retrieved for each query record. A record $r_A \in A$ and its k corresponding neighbouring records $r_1, r_2 \ldots, r_k, r_i \in B$ build the set of pairs $(r_A, r_1), (z_A, r_2) \ldots (r_A, r_k)$ and are added to the candidate set C. For the nearest neighbour search, FAISS performs a merge-sort on I_B, which has a time complexity of $O(n \log_2 n)$ [26].

4 Blocking-Only Evaluation

This section evaluates the candidate sets generated by SC-Block concerning recall and candidate set size and compares them to the sets generated by seven other blocking methods from related work. The evaluation of SC-Block's candidate sets is conducted in two scenarios: (1) with a fixed value of k=5 to draw general conclusions about the recall and precision of the candidate sets, and (2) with k tuned to minimize the number of missed matching pairs while keeping the candidate set size as small as possible.

4.1 Baseline Blocking Methods

This section describes the blocking methods from related work that we compare to SC-Block. Further details on the baseline blocker configurations are available online[8]. Table 2 provides an overview of the methods based on the criteria

[8] https://webdatacommons.org/largescaleproductcorpus/wdc-block/.

blocking technique, training procedure, supervision, and encoder network. If a criterion is not applicable, it is marked with a '–'.

Table 2. Blocking techniques.

	Learning	Blocking technique	Training	Encoder
JedAI [38]	unsupervised	key-based blocking	–	–
BM25 [42]	unsupervised	nearest-neighbour search	–	–
Auto [49]	self-supervised	nearest-neighbour search	autoencoder	fasttext
CTT [49]	self-supervised	nearest-neighbour search	pair-wise	fasttext
BT [51]	self-supervised	nearest-neighbour search	contrastive	RoBERTa
SimCLR [51]	self-supervised	nearest-neighbour search	contrastive	RoBERTa
SBERT [46]	supervised	nearest-neighbour search	pair-wise	RoBERTa
SC-Block	supervised	nearest-neighbour search	contrastive	RoBERTa

JedAI. The JedAI framework implements symbolic techniques for entity resolution pipelines. For our experiments, we run the default configuration of the block building, block cleaning & filling steps of the linked tutorial[9] [16].

BM25. BM25 is an unsupervised blocker [42] that uses a vector space model [48] and the BM25 term weighting scheme [47] to compute a similarity score for the nearest neighbour search. BM25 is evaluated using whitespace tokenization (referred to as BM25) and tri-grams (referred to as BM25$_3$).

Autoencoder (Auto) and Cross Tuple Training (CTT). Auto and CTT use fasttext to embed the tokens of the entity descriptions and average the token embeddings to obtain a record embedding [49]. For Auto, the record embeddings are sent through an autoencoder. Auto is self-supervised and thus requires no labelled training data. For CTT, the record embeddings are sent through a Siamese summarizer and a classifier learns to detect matches based on the element-wise difference of the created embeddings. CTT is trained on synthetically produced training data derived from the two blocked datasets.

Barlow Twins (BT) and SimCLR. BT and SimCLR, two self-supervised blockers, duplicate and augment entity descriptions in training batches by dropping random tokens [51]. The entity descriptions are embedded using a RoBERTa model to be comparable to SC-Block. During training, BT aligns the cross-correlation of augmented and original entity descriptions with the identity matrix [52], while SimCLR aims to maximize the similarity between embeddings of the same records and minimize it for different records within a batch [8].

Sentence-Bert (SBERT). SBERT requires labelled pairs of matching and non-matching records for training [46]. The entity descriptions of a labelled pair

[9] https://github.com/AI-team-UoA/pyJedAI/blob/main/docs/tutorials/CleanCleanER.ipynb.

are embedded using a pre-trained RoBERTa language model to ensure that the encoder network of SBERT and SC-Block is the same. During training, the cosine similarity of both embeddings is calculated, and the weights of the language model are updated using mean-squared error loss.

4.2 Implementation

We used a shared server with 96 × 3.6 GHz CPU cores, 1024 GB RAM and an NVIDIA RTX A6000 GPU for the experiments. We use Elasticsearch[10] to implement BM25 and BM25$_3$. The Elasticsearch instance runs on a virtual machine with 4 × 2.1 GHz CPU cores, 32 GB RAM and 512 GB storage. If an execution exceeds 48 h or if the shared server's memory is insufficient, the experiment is labelled as timed-out or as an out-of-memory error, respectively.

4.3 Results for Fixed K

We now analyze the candidate sets of the nearest neighbour blockers with a fixed number of nearest neighbours $k = 5$. By fixing the hyperparameter k, differences in recall and precision become visible that are not visible when k is tuned. $k = 5$ is chosen because it allows the blockers to score high recall, especially on the datasets A-B, A-G and W-A, which exhibit a maximum number of matching neighbours smaller than 5. At the same time, blockers miss matching pairs, because $k = 5$ is not sufficiently large, making it possible to see differences. In Sect. 4.4, we adjust the value of k to ensure that the candidate sets of the nearest neighbour blockers exceed a threshold of 99.5% on the validation set, to minimize the number of missed matches. Table 3 shows runtime, recall and precision of the candidate sets generated by the nearest neighbour blockers. The candidate sets are evaluated based on the record pairs of the respective test set. The lowest runtime and the highest score per column are marked in bold.

Table 3. Runtime (RT) in seconds, recall (R) in %, and precision (P) in % of the candidate sets generated by all nearest neighbour blockers with fixed $k = 5$

	A-B			A-G			W-A			WB$_{small}$			WB$_{medium}$			WB$_{large}$		
	RT	R	P	RT	R	P	RT	R	P	RT	R	P	RT	R	P	RT	R	P
SC-Block	175	**100**	36	290	**97**	36	588	96	**33**	402	**72**	**57**	0.7k	**66**	**64**	18.1k	**57**	**74**
BM25$_3$	18	98	27	36	94	31	290	**97**	21	300	53	39	1.0k	47	44	172.8k	timed-out	
BM25	**9**	92	27	**12**	94	32	**45**	96	22	**44**	59	40	**0.2k**	54	45	**4.1k**	42	54
Auto	20	75	31	27	83	36	353	82	23	127	43	40	0.7k	36	41	out-of-memory		
CTT	95	77	32	174	82	36	745	81	24	448	43	38	9.8k	35	41	out-of-memory		
SimCLR	178	90	**44**	41	92	**37**	534	91	32	728	35	39	1.1k	21	39	11.4k	3	33
BT	185	95	29	266	90	34	210	93	25	838	32	35	1.2k	21	36	9.6k	13	33
SBERT	498	74	75	307	29	41	441	31	17	1.9k	45	49	1.5k	35	55	8.4k	24	57

[10] https://www.elastic.co/what-is/elasticsearch.

On average, SC-Block has the highest recall and precision scores compared to the other blockers. It is also evident that the recall of all generated candidate sets decreases from WB_{small} to WB_{large}, indicating that a larger vocabulary and Cartesian product make the dataset more challenging. In the following paragraphs, we compare the performance of SC-Block to the performance of the other blockers.

Unsupervised Blockers. On datasets A-B, A-G, and W-A, BM25 and BM25_3 exhibit similar performance to SC-Block. However, on WDC-Block, BM25 and BM25_3 miss an average of 16.3% more pairs than SC-Block. This decrease in performance can be attributed to the larger token vocabulary of WDC-Block, which complicates the identification of matching pairs through token overlap by both BM25 blockers.

Self-supervised Blockers. Auto and CTT have the poorest performance among dense nearest neighbour-based blockers in datasets A-B, A-G, and W-A. This is due to the pre-trained fasttext embeddings used in Auto and CTT being less potent compared to the robust RoBERTa embeddings utilized in other blockers. SimCLR and BT lose between 3% and 9% more pairs than SC-Block on the datasets A-B, A-G and W-A with BT performing marginally better than SimCLR. On the W-A dataset, the slight difference between SC-Block, SimCLR, and BT can be attributed to the relatively low number of positive training pairs available. When supervision is absent, the supervised contrastive loss algorithm degrades to the loss of SimCLR [29]. On WDC-Block SimCLR and BT miss on average 45% and 43% more pairs than SC-Block. SC-Block holds an advantage as it leverages the guidance of matching and non-matching training pairs present in the datasets. Auto and CTT outperform BT and SimCLR on WDC-Block. During training, Auto and CTT require exposure to all tokens in a dataset, which necessitates training on all records from the initial datasets. However, BT and SimCLR are solely trained on the records mentioned in the training and validation set. This difference in training accounts for the performance gap and highlights the susceptibility of self-supervised blockers to unseen out-of-distribution records. SC-Block uses the same records for training as BT and SimCLR. However, its recall and precision scores demonstrate increased robustness against noise. The nearest neighbour search on WB_{large} of Auto and CTT results in an out-of-memory error, revealing a limitation of this implementation for large blocked datasets.

Supervised Blockers. The supervised blocker SBERT performs poorly. This indicates that SC-Block's supervised contrastive loss utilizes supervision more effectively than SBERT's pair-wise cosine similarity loss.

4.4 Results for 99.5% Recall on Validation Set

This section analyses the impact of tuning the hyperparameter k of the nearest neighbour blockers. Increasing k raises the likelihood of including all matches in the candidate set. However, higher k values generate larger candidate sets, which increase the runtime of the entity resolution pipeline since the matcher must compare more candidate pairs. To analyse how the blockers handle this trade-off, we evaluate each nearest neighbour blocker with increasing values of k, starting from $k = 1$. Once the recall of the candidate set exceeds 99.5% on the validation set, k is fixed and the recall is evaluated on the test sets. To limit the search space, we cap k at a maximum of 50 on A-B, A-G, W-A and WB_{small}, 100 on WB_{medium} and 200 on WB_{large}. Table 4 presents the values of k, the runtime, the recall achieved on the test set, and the size of the candidate sets. The highest recall and lowest k, runtime and candidate set size per dataset are highlighted.

Table 4. k, runtime (RT) in seconds, recall (R) in % and candidate set size ($|C|$) of the candidate sets on A-B, A-G and W-A. k is tuned on the validation set.

	A-B				A-G				W-A			
	k	RT	R	$\lvert C\rvert$	k	RT	R	$\lvert C\rvert$	k	RT	R	$\lvert C\rvert$
SC-Block	**5**	175	**100**	**5k**	**8**	293	**100**	**11k**	**12**	558	97	**31k**
BM25_3	13	18	**100**	14k	27	43	**100**	37k	**12**	290	**99**	**31k**
BM25	7	**9**	95	8k	29	**15**	99	40k	21	**45**	**99**	54k
JedAI	–	**9**	97	13k	–	19	98	20k	–	340	**99**	172k
Auto	50	20	97	54k	50	27	95	68k	50	353	93	128k
CTT	50	95	97	54k	50	174	95	68k	50	745	92	128k
BT	20	182	99	22k	30	49	98	41k	26	545	**99**	66k
SimCLR	29	186	96	31k	30	276	98	41k	23	221	96	59k
SBERT	26	502	86	28k	30	314	51	41k	50	441	49	128k
	WB_{small}				WB_{medium}				WB_{large}			
	k	RT	R	$\lvert C\rvert$	k	RT	R	$\lvert C\rvert$	k	RT	R	$\lvert C\rvert$
SC-Block	**14**	406	94	70k	**20**	426	92	**100k**	**50**	**8.9k**	90	**5M**
BM25_3	50	679	94	250k	100	779	94	500k	200	173.0k	timed-out	
BM25	50	86	**97**	250k	100	**186**	**98**	500k	200	14.2k	**96**	20M
JedAI	–	**50**	55	**51k**	–	1.6k	81	561k	–	173k	timed-out	
Auto	50	127	85	250k	100	450	80	500k	out-of-memory			
CTT	50	448	85	250k	100	672	80	500k	out-of-memory			
BT	50	770	67	250k	100	870	43	500k	200	10.7k	34	20M
SimCLR	50	923	70	250k	100	1.0k	46	500k	200	12.9k	36	20M
SBERT	50	1567	59	250k	100	2.1k	78	500k	200	13.5k	59	20M

Overall, the candidate sets of the blockers differ mainly in size, while most of the nearest neighbour search blocking techniques have a recall close to 1. SC-Block creates relatively small candidate sets. The candidate sets are now compared in detail.

Unsupervised Blockers. The BM25 weighting schema is insufficient to achieve competitive candidate set sizes on the WDC-Block benchmark, as evident from the large candidate sets of unsupervised blockers. This is due to the considerable number of corner cases in the validation sets and the vast vocabulary of the datasets. SC-Block is fine-tuned and can exploit matching information from the training set to acquire knowledge about the corner cases. JedAi's pruning of blocks negatively affects the recall score on WB_{small} and WB_{medium} and causes a timeout on WB_{large}.

Self-supervised Blockers. The self-supervised blockers BT and SimCLR require two to six times higher values of k than SC-Block to produce candidate sets that meet the recall threshold of 99.5% on the validation set. Auto and CTT fail to meet the recall threshold for any of the datasets. On the WDC-Block benchmark, Auto and CTT again outperform BT and SimCLR on the datasets WB_{small} and WB_{medium} by finding on average 25% more pairs. The out-of-memory error on WB_{large} arises from the nearest neighbour search implementation of Auto and CTT.

Supervised Blockers. SBERT overfits the training and validation data because it reaches a high recall score on the validation set but reaches a much lower recall on the test set. SC-Block's supervised contrastive loss better utilizes the training data than SBERT's mean-squared error loss on the cosine similarity of the training pairs.

The best-performing blockers SC-Block, BM25_3 and BT produce candidate sets with a recall close to 1, yet vary in their sizes. In Sect. 5, we combine these blockers with different matchers and measure the impact of the blockers on the F1 score and runtime of the entire entity resolution pipeline.

5 Evaluation Within Entity Resolution Pipelines

To evaluate the impact of the SC-Block, BM25_3, and BT blockers on entity resolution pipelines, we assess (1) the F1 score and runtime of the pipelines with these blockers and (2) whether the reduced runtime of the pipeline with SC-Block compensates for the training time of the blocker. The candidate sets, generated by the blockers, are processed by the matchers Magellan [30], RoBERTa Cross Encoder (CE) [6], Ditto [31], and SupCon-Match [44] to produce final sets of matching pairs. Each blocker uses the tuned k from Sect. 4.4. The matchers are fine-tuned on the same training sets as the blockers.

5.1 F1 Score and Runtime

This section analyses the impact of SC-Block, BM25_3, and BT on the F1 score and runtime of entity resolution pipelines. The runtime refers to the execution time of blocking and matching, excluding training times, which is discussed in Sect. 5.2. Table 5 shows the F1 score and the runtime in seconds, with the highest F1 score and lowest runtime highlighted.

Table 5. F1 score in % and runtime in seconds (RT) of state-of-the-art entity resolution pipelines

Blocker	Matcher	A-B		A-G		W-A		WB$_{small}$		WB$_{medium}$		WB$_{large}$	
		F1	RT	F1	RT	F1	RT	F1	RT	F1	RT	F1	RT
SC-Block	SupCon	**93**	71	**80**	133	81	355	71	**383**	72	**742**	72	**30.7k**
	Ditto	91	185	76	336	**86**	754	**77**	918	**78**	1.5k	**78**	66.5k
	CE	80	**57**	64	101	**86**	303	**77**	351	77	606	76	27.9k
	Magellan	52	335	58	511	68	1.5k	59	2.1k	61	2198	61	46.2k
BM25$_3$	SupCon	**93**	88	**80**	467	82	490	69	2.0k	70	4.2k	timed-out	
	Ditto	90	228	76	941	**86**	1.0k	74	3.6k	75	7.9k	timed-out	
	CE	79	70	64	353	85	437	73	1.8k	74	4k	timed-out	
	Magellan	51	216	58	555	67	1.9k	44	2.4k	43	3.7k	timed-out	
BT	SupCon	**93**	125	**80**	157	81	548	58	1.5k	45	2.9k	40	119.6k
	Ditto	90	296	75	442	**86**	1.0k	64	3.0k	50	6.3k	44	246.1k
	CE	79	75	64	**93**	85	**188**	63	1.4k	49	2.6k	43	109.8k
	Magellan	51	212	57	346	66	1.2k	42	2.4k	36	2.7k	32	72.1k

F1 Score. Table 5 shows that the F1 scores of the pipeline depend on the matcher for candidate sets with high recall. Pipelines with BT on WDC-Block are an exception because of the low recall of the respective candidate set, which harms the F1 scores as known from Sect. 4.4. On WDC-Block, pipelines consisting of SC-Block and Ditto or CE yield better results than pipelines with SC-Block and SupCon. This is because Ditto and CE learn distinctive patterns that are not acquired by SC-Block and SupCon. By combining SC-Block with Ditto or CE, these varied patterns are effectively exploited.

Runtime. In general, it can be concluded that using an effective blocker such as SC-Block can reduce the runtime of a pipeline. When comparing SC-Block to BM25$_3$ and BT, we can observe that the smaller candidate sets of SC-Block result in pipelines that run 4 to 7 times faster. For example, the runtime of the pipeline consisting of BT and CE on the WB$_{large}$ dataset is 30 h, whereas SC-Block reduces the workload of the CE matcher and reduces the pipeline runtime to 8 h. On the WB$_{large}$, BM25$_3$ requires a significant amount of time to generate large candidate sets, leading to pipeline timeouts after a two-day runtime. However, on datasets A-B, A-G, and W-A, using the smaller candidate sets of SC-Block results in pipelines that finish 1.5 to 2 times faster compared to pipelines with BM25$_3$ and BT. To see if training SC-Block is reasonable, we set the reduction in runtime into context with the training time in Sect. 5.2.

5.2 Impact of Training Time

If online training of a blocker is necessary, it is only reasonable to do so if the runtime of the pipeline, including training time, is shorter than the runtime of a pipeline with a blocker that does not require further training. Therefore, we consider the training time of the SC-Block and BT blockers in relation to the overall runtime of the entity resolution pipelines. Table 6 displays the blocker training time (BTT) and the complete runtimes (CT), which include the blocker training time, blocking time, and matching time of all pipelines.

Table 6. Blocker Training Time (BTT) and complete runtime of entity resolution pipelines (CT) in seconds

Blocker	Matcher	A-B		A-G		W-A		WB$_{small}$		WB$_{medium}$		WB$_{large}$	
		BTT	CT	BTT	CT	BTT	CT	BTT	CT	BTT	RT	BTT	RT
SC-Block	SupCon	227	298	259	392	430	0.8k	**343**	**0.7k**	**343**	1.1k	**343**	31.0k
	Ditto	227	412	259	595	430	1.2k	**343**	1.3k	**343**	1.8k	**343**	66.9k
	CE	227	284	259	360	430	0.7k	**343**	**0.7k**	**343**	**0.9k**	**343**	**28.2k**
	Magellan	227	562	259	770	430	1.9k	**343**	2.4k	**343**	2.5k	**343**	46.6k
BM25$_3$	SupCon	–	88	–	467	–	0.5k	–	1.9k	–	4.2k	timed-out	
	Ditto	–	228	–	941	–	1.1k	–	3.6k	–	7.9k	timed-out	
	CE	–	**70**	–	353	–	0.4k	–	1.8k	–	4.0k	timed-out	
	Magellan	–	216	–	555	–	1.9k	–	2.4k	–	3.7k	timed-out	
BT	SupCon	**158**	283	**236**	393	**130**	0.7k	779	2.3k	779	3.7k	779	120.4k
	Ditto	**158**	454	**236**	678	**130**	1.1k	779	3.9k	779	7.1k	779	246.9k
	CE	**158**	233	**236**	**329**	**130**	**0.3k**	779	2.2k	779	3.4k	779	110.6k
	Magellan	**158**	370	**236**	582	**130**	1.3k	779	3.2k	779	3.5k	779	72.9k

Small Benchmark Datasets. For the small datasets A-B, A-G and W-A, the unsupervised BM25$_3$ blocker is as efficient as the supervised and self-supervised blocking methods. Although SC-Block and BT show a small improvement in runtime on A-B, A-G, and W-A, their advantage is counterpoised by the training time of the blockers. Training SC-Block for these small datasets is not practical since entity resolution pipelines with the unsupervised BM25$_3$ blocker and a tuned k value generate recall scores near 1, equivalent F1 scores, require no time for training and runtime is faster than SC-Block's runtime and training time.

WDC-Block. The runtime of pipelines using BM25$_3$ on WDC-Block increases with a larger vocabulary and a larger cartesian product of candidate pairs. In this scenario, training SC-Block is a reasonable option since the training data is available. The five minutes of training time reduce the pipeline's runtime compared to BM25$_3$ even on WB$_{small}$ by 20 min for the matchers SupCon and CE. Due to our computational restrictions, it was only possible to run competitive pipelines on WB$_{large}$ if SC-Block was applied for blocking.

6 Related Work

Entity resolution [11], also known as Link Discovery [35], is a crucial task in the process of integrating data from the Web. The entity resolution pipelines consist of two main steps: blocking and matching, which both have been studied for decades [2,11,35,36,40].

Blocking. Blocking is traditionally tackled as an unsupervised task by generating a blocking key value from each record [1,10,50]. Records with the same blocking key value are assigned to the same block. Instead of one blocking key value, some works use multiple blocking keys [13,14,22], or multi-dimensional blocking [3,14,24]. Meta-blocking, implemented in the JedAI toolkit [38], extends blocking by blocking key values with an additional pruning step that first weights candidate record pairs by their matching likelihood and discards pairs with the lowest scores [16]. This pruning step has also been implemented using supervision [18,39]. Other works add a clustering step to exploit the transitivity of candidate record pairs [3,50]. The blocking key value has also been used as a sorting key for sorted neighbourhood blockers [23,27]. Related works also utilized supervision [4,53] or self-supervision [28] to learn a blocking strategy. Recently, deep learning for blocking has become popular [25,53]. DeepBlocker [49] explored the use of deep self-supervised learning for blocking, introducing two blockers based on auto-encoding (Auto) and cross-tuple training (CTT). Sudowoodo [51] applies self-supervised learning in combination with a transformer model and the two loss functions SimCLR and Barlow Twins (BT). The supervised contrastive loss of SC-Block is an extension of the SimCLR loss [8,29]. We compare SC-Block to the blockers JedAI, Auto, CTT, SimCLR and BT from the related work. Except for JedAI all benchmarked blockers apply the nearest neighbour search for candidate set generation. Mugeni et al. use self-supervised contrastive learning to position embeddings in the embeddings space and apply an unsupervised community detection technique from graph structure mining to block record pairs [34]. Sparkly has recently demonstrated that TF-IDF and BM25 are strong baseline blockers [42].

Entity Matching. Most current entity matching methods rely on deep learning techniques [2]. This trend was initialized by Ebraheem et al. [15] and Mudgal et al. [33]. Recently, several transformer-based matching methods have achieved state-of-the-art performance [6,20,21,31,43]. Among them, Peeters and Bizer [44] and Wang, Li and Wang [51] use contrastive learning, similar to our work.

Contrastive Learning. SC-Block is inspired by ideas from computer vision [29], information retrieval [19], and entity matching [44] where contrastive learning has shown to be more effective than the traditional cross-entropy-based learning. Specifically, Gao, Yao and Chen [19] use contrastive learning to learn sentence embeddings without any supervision. Supervised contrastive learning still suffers from challenges, including the robustness of learned representations [12] and class collapse, i.e., all samples from a cluster are mapped to the same representation [7]. Similar to SupCon [44], SC-Block applies source-aware sampling to increase the robustness of the learned embeddings. The main difference to SupCon is that SC-Block's embeddings are optimized for blocking and not for matching, which requires a longer pre-training of the embeddings.

7 Conclusion

The paper introduced the WDC-Block blocking benchmark, which employs data from a large number of Web data sources. The benchmark offers a large maximal Cartesian product and vocabulary size. The paper further proposes the SC-Block blocking method, which employs supervised contrastive learning to position records in an embedding space. The evaluation of SC-Block using WDC-Block and three smaller benchmark datasets showed that SC-Block generates smaller candidate sets than other state-of-the-art blockers. When using SC-Block together with the best-performing matcher on WDC-Block, the runtime for pipelines decreased from 30 to 8 h, performing 1.5 to 4 times faster than pipelines utilizing other state-of-the-art blockers and the same matcher. The reduced runtime overcompensates the time that is required to train SC-Block.

A Appendix - Additional Experiments

This appendix presents additional experiments that empirically justify several low-level design decisions made for SC-Block. We evaluate (1) the impact of another record serialization and (2) the impact of switching query and index table. Furthermore, WDC-Block is available with three training set sizes. We evaluate (3) how the different training set sizes impact SC-Block's performance.

Serialization. The default serialization method adds the attribute name to entity descriptions when serializing records. It is unclear whether the attribute name affects how SC-Block places embeddings in the embedding space. In the 'No attribute name' experiment, we only concatenate attribute values to derive entity descriptions. A fixed value of k=5 is used in these experiments to observe differences in precision and recall. We use a fixed $k = 5$ for these experiments to see differences in precision and recall. The results in Table 7 show that for the benchmark datasets A-B, A-G and W-A the serialization has only a marginal impact on SC-Block's recall and precision. On WDC-Block we see that the serialization with attribute names achieves better recall and precision scores than the serialization without attribute names. We can conclude that the attribute names are not harmful and structure the entity descriptions, which makes the entity descriptions better readable.

Table 7. Recall (R) in % and Precision (P) in % for different Record Serialization

Dataset	A-B		A-G		W-A		WB$_{small}$		WB$_{medium}$		WB$_{large}$	
Serialization	R	P	R	P	R	P	R	P	R	P	R	P
Attribute Names	99	36	97	36	96	33	72	57	66	64	57	74
No attribute name	100	39	98	34	96	33	69	54	62	62	51	69

Switch Query Table and Index Table. By default, the table with fewer records is selected as the query table and the larger table is selected as the index table. In this study, we examine the effects of swapping the query and index tables. The baseline for these experiments is the corresponding SC-Block run with an optimized k from Sect. 4.4. The value of k for the experiments involving switched tables is selected to ensure that the candidate sets have the same size, denoted by $|C|$, as the candidate sets of the baseline experiment where tables are not switched. If this is not possible, the smallest possible value of k is chosen that generates a candidate set larger than the candidate set of the baseline experiment where the query table and the index table are not switched.

Table 8. Recall (R) in %, Query Table (QT) Size and k for switching query and index table

QT	DS	R	QT Size	k	$\|C\|$
A	A-B	99	1,081	5	5,405
B	A-B	99.0	1,092	5	5,460
A	A-G	99.6	1,363	8	10,904
B	A-G	99.1	3,266	4	13,064
A	W-A	96.9	2,554	12	30,648
B	W-A	96.9	22,074	2	44,148
A	WB_{small}	93.5	5,000	14	70,000
B	WB_{small}	91.7	5,000	14	70,000
A	WB_{medium}	91.9	5,000	20	100,000
B	WB_{medium}	0.0	200,000	1	200,000
A	WB_{large}	89.5	100,000	50	5,000,000
B	WB_{large}	51.6	2,000,000	3	6,000,000

The results in Table 8 show that for the benchmarks A-B, A-G, W-A and WD_{small} switching query and index table while keeping the candidate set size stable results in a comparable recall. The advantage for A-G and W-A is that k is much smaller, meaning fewer k values need to be tested during the hyperparameter search. On WB_{medium} and WB_{large} table B is 20 to 40 times larger. If the query table and the index table are switched and k is chosen such that the candidate set size is similar, the recall is much lower.

Different Training Set Sizes. WDC-Block is available with three training set sizes. To analyze how the different training set sizes impact the performance of SC-Block, we train SC-Block on the different training sets (SC-Block_{small}, SC-Block_{medium} and SC-Block_{large}) and search for k as described in Sect. 4.4. We compared SC-Block to SimCLR, which was trained on records from the same three training sets. Unlike SC-Block, SimCLR does not consider the information about matching records in the training set. If only datasets without matching information are available, SC-Block's loss is equivalent to SimCLR's loss.

Table 9. k and Recall (R) in % for different Training Set Sizes

	Pos.	Neg.	WDC-B$_{small}$			WDC-B$_{medium}$			WDC-B$_{large}$		
	Dev.	Dev.	k	R	$\|C\|$	k	R	$\|C\|$	k	R	$\|C\|$
SC-Block$_{large}$	9,680	15,752	14	93.5	70k	20	91.9	100k	50	89.5	5M
SC-Block$_{medium}$	2,338	2,819	50	92.6	250k	100	86.6	500k	200	77.8	20M
SC-Block$_{small}$	399	611	50	71.8	250k	100	52.0	500k	200	42.2	20M
SimCLR	0	0	50	69.5	250k	100	46.0	500k	200	36.1	20M

The results in Table 9 demonstrate that more training data improves recall and decreases the candidate set size of SC-Block. Furthermore, the small training set alone is adequate to enhance the results beyond those of the SimCLR baseline. These findings confirm the usefulness of training data for SC-Block.

References

1. Aizawa, A., Oyama, K.: A fast linkage detection scheme for multi-source information integration. In: Proceedings of the Sixth IEEE International Conference on Data Mining, pp. 30–39. IEEE, Tokyo, Japan (2005). https://doi.org/10.1109/WIRI.2005.2
2. Barlaug, N., Gulla, J.A.: Neural networks for entity matching: a survey. ACM Trans. Knowl. Discov. Data **15**(3), 1–37 (2021). https://doi.org/10.1145/3442200
3. van Bezu, R., Borst, S., Rijkse, R., Verhagen, J., Vandic, D., Frasincar, F.: Multi-component similarity method for web product duplicate detection. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp. 761–768. Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2695664.2695818
4. Bilenko, M., Kamath, B., Mooney, R.J.: Adaptive blocking: learning to scale up record linkage. In: Proceedings of the Sixth IEEE International Conference on Data Mining, pp. 87–96. IEEE, Hong Kong, China (2006). https://doi.org/10.1109/ICDM.2006.13
5. Brinkmann, A., Primpeli, A., Bizer, C.: The web data commons Schema.org data set series. In: Companion Proceedings of the ACM Web Conference 2023, pp. 136–139. Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3543873.3587331
6. Brunner, U., Stockinger, K.: Entity matching with transformer architectures - a step forward in data integration. In: Proceedings of the 23rd International Conference on Extending Database Technology, pp. 463–473. OpenProceedings.org, Copenhagen, Denmark (2020). https://doi.org/10.5441/002/edbt.2020.58
7. Chen, M., Fu, D.Y., Narayan, A., Zhang, M., Song, Z., Fatahalian, K., et al.: Perfectly balanced: improving transfer and robustness of supervised contrastive learning. In: Proceedings of the 39th International Conference on Machine Learning, vol. 162, pp. 3090–3122. PMLR, Baltimore, Maryland, USA (2022)
8. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: Proceedings of the 37th International Conference on Machine Learning, vol. 119, pp. 1597–1607. PMLR, Virtual (2020)

9. Christen, P.: Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection, 1st edn. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31164-2
10. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. Knowl. Data Eng. **24**(9), 1537–1555 (2012). https://doi.org/10.1109/TKDE.2011.127
11. Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., Stefanidis, K.: An overview of end-to-end entity resolution for big data. ACM Comput. Surv. **53**(6), 1–42 (2021). https://doi.org/10.1145/3418896
12. Chuang, C.Y., Robinson, J., Lin, Y.C., Torralba, A., Jegelka, S.: Debiased contrastive learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Proceedings of the 33rd Annual Conference on Neural Information Processing Systems, vol. 33, pp. 8765–8775. Curran Associates, Inc., Virtual (2020)
13. van Dam, I., van Ginkel, G., Kuipers, W., Nijenhuis, N., Vandic, D., Frasincar, F.: Duplicate detection in web shops using LSH to reduce the number of computations. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing, pp. 772–779. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2851613.2851861
14. De Assis Costa, G., Parente De Oliveira, J.M.: A blocking scheme for entity resolution in the semantic web. In: Proceedings of the 30th International Conference on Advanced Information Networking and Applications, pp. 1138–1145. IEEE, Crans-Montana (2016). https://doi.org/10.1109/AINA.2016.23
15. Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., Tang, N.: Distributed representations of tuples for entity resolution. Proc. VLDB Endow. **11**(11), 1454–1467 (2018). https://doi.org/10.14778/3236187.3236198
16. Efthymiou, V., Papadakis, G., Papastefanatos, G., Stefanidis, K., Palpanas, T.: Parallel meta-blocking for scaling entity resolution over big heterogeneous data. Inf. Syst. **65**, 137–157 (2017). https://doi.org/10.1016/j.is.2016.12.001
17. Efthymiou, V., Stefanidis, K., Christophides, V.: Benchmarking blocking algorithms for web entities. IEEE Trans. Big Data **6**(2), 382–395 (2020). https://doi.org/10.1109/TBDATA.2016.2576463
18. Gagliardelli, L., Papadakis, G., Simonini, G., Bergamaschi, S., Palpanas, T.: Generalized supervised meta-blocking. Proc. VLDB Endow. **15**(9), 1902–1910 (2022). https://doi.org/10.14778/3538598.3538611
19. Gao, T., Yao, X., Chen, D.: 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, pp. 6894–6910. Association for Computational Linguistics, Virtual and Punta Cana, Dominican Republic (2021). https://doi.org/10.18653/v1/2021.emnlp-main.552
20. Genossar, B., Gal, A., Shraga, R.: The battleship approach to the low resource entity matching problem. Proc. ACM Manag. Data **1**(4), 1–25 (2023). https://doi.org/10.1145/3626711
21. Genossar, B., Shraga, R., Gal, A.: FlexER: flexible entity resolution for multiple intents. Proc. ACM Manag. Data **1**(1), 1–27 (2023). https://doi.org/10.1145/3588722
22. Hartveld, A., et al.: An LSH-based model-words-driven product duplicate detection method. In: Krogstie, J., Reijers, H.A. (eds.) CAiSE 2018. LNCS, vol. 10816, pp. 409–423. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91563-0_25
23. Hernández, M.A., Stolfo, S.J.: The merge/purge problem for large databases. ACM SIGMOD Rec. **24**(2), 127–138 (1995). https://doi.org/10.1145/568271.223807

24. Isele, R., Jentzsch, A., Bizer, C.: Efficient multidimensional blocking for link discovery without losing recall. In: Marian, A., Vassalos, V. (eds.) Proceedings of the 14th International Workshop on the Web and Databases, pp. 1–6. Association for Computing Machinery, New York, NY, USA (2011)
25. Javdani, D., Rahmani, H., Allahgholi, M., Karimkhani, F.: DeepBlock: a novel blocking approach for entity resolution using deep learning. In: Proceedings of the 5th International Conference on Web Research, pp. 41–44. IEEE, Tehran, Iran (2019). https://doi.org/10.1109/ICWR.2019.8765267
26. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data **7**(3), 535–547 (2021). https://doi.org/10.1109/TBDATA.2019.2921572
27. Kejriwal, M., Miranker, D.P.: Sorted neighborhood for schema-free RDF data. In: Gandon, F., Guéret, C., Villata, S., Breslin, J., Faron-Zucker, C., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9341, pp. 217–229. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25639-9_38
28. Kejriwal, M., Miranker, D.P.: An unsupervised instance matcher for schema-free RDF data. J. Web Semant. **35**(2), 102–123 (2015). https://doi.org/10.1016/j.websem.2015.07.002
29. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., et al.: Supervised contrastive learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Proceedings of the 33rd Annual Conference on Neural Information Processing Systems, vol. 33, pp. 18661–18673. Curran Associates, Inc. (2020)
30. Konda, P., Das, S., Doan, A., Ardalan, A., Ballard, J.R., et al.: Magellan: toward building entity matching management systems over data science stacks. Proc. VLDB Endow. **9**(13), 1581–1584 (2016). https://doi.org/10.14778/3007263.3007314
31. Li, Y., Li, J., Suhara, Y., Doan, A., Tan, W.C.: Deep entity matching with pre-trained language models. Proc. VLDB Endow. **14**(1), 50–60 (2020). https://doi.org/10.14778/3421424.3421431
32. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., et al.: RoBERTa: a robustly optimized bert pretraining approach (2019). arXiv:1907.11692 [cs]
33. Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., et al.: Deep learning for entity matching: a design space exploration. In: Proceedings of the 2018 International Conference on Management of Data, pp. 19–34. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3183713.3196926
34. Mugeni, J.B., Amagasa, T.: A graph-based blocking approach for entity matching using contrastively learned embeddings. ACM SIGAPP Appl. Comput. Rev. **22**(4), 37–46 (2023). https://doi.org/10.1145/3584014.3584017
35. Nentwig, M., Hartung, M., Ngonga Ngomo, A.C., Rahm, E.: A survey of current link discovery frameworks. Semant. Web **8**(3), 419–436 (2016). https://doi.org/10.3233/SW-150210
36. Ngomo, A.C.N., Auer, S.: LIMES: a time-efficient approach for large-scale link discovery on the web of data. In: Proceedings of the 22nd international joint conference on Artificial Intelligence, vol. 3, pp. 2312–2317. AAAI Press, Barcelona, Catalonia, Spain (2011)

37. Papadakis, G., Fisichella, M., Schoger, F., Mandilaras, G., Augsten, N., Nejdl, W.: Benchmarking filtering techniques for entity resolution. In: Proceedings of the IEEE 39th International Conference on Data Engineering, pp. 653–666. IEEE, Anaheim, CA, USA (2023). https://doi.org/10.1109/ICDE55515.2023.00389
38. Papadakis, G., Mandilaras, G., Gagliardelli, L., Simonini, G., Thanos, E., Giannakopoulos, G., et al.: Three-dimensional entity resolution with JedAI. Inf. Syst. **93**, 101565 (2020). https://doi.org/10.1016/j.is.2020.101565
39. Papadakis, G., Papastefanatos, G., Koutrika, G.: Supervised meta-blocking. Proc. VLDB Endow. **7**(14), 1929–1940 (2014). https://doi.org/10.14778/2733085.2733098
40. Papadakis, G., Skoutas, D., Thanos, E., Palpanas, T.: Blocking and filtering techniques for entity resolution: a survey. ACM Comput. Surv. **53**(2), 1–42 (2020). https://doi.org/10.1145/3377455
41. Papadakis, G., Svirsky, J., Gal, A., Palpanas, T.: Comparative analysis of approximate blocking techniques for entity resolution. Proc. VLDB Endow. **9**(9), 684–695 (2016). https://doi.org/10.14778/2947618.2947624
42. Paulsen, D., Govind, Y., Doan, A.: Sparkly: a simple yet surprisingly strong TF/IDF blocker for entity matching. Proc. VLDB Endow. **16**(6), 1507–1519 (2023). https://doi.org/10.14778/3583140.3583163
43. Peeters, R., Bizer, C.: Dual-objective fine-tuning of BERT for entity matching. Proc. VLDB Endow. **14**(10), 1913–1921 (2021). https://doi.org/10.14778/3467861.3467878
44. Peeters, R., Bizer, C.: Supervised contrastive learning for product matching. In: Companion Proceedings of the Web Conference 2022, pp. 248–251. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3487553.3524254
45. Peeters, R., Der, R.C., Bizer, C.: WDC products: a multi-dimensional entity matching benchmark. In: Proceedings of the 27th International Conference on Extending Database Technology, vol. 27, pp. 22–33. OpenProceedings.org, Konstanz (2023). https://doi.org/10.48786/edbt.2024.03
46. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (2019). https://doi.org/10.18653/v1/D19-1410
47. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Found. Trends® Inf. Retr. **3**(4), 333–389 (2009). https://doi.org/10.1561/1500000019
48. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975). https://doi.org/10.1145/361219.361220
49. Thirumuruganathan, S., et al.: Deep learning for blocking in entity matching: a design space exploration. Proc. VLDB Endow. **14**(11), 2459–2472 (2021). https://doi.org/10.14778/3476249.3476294
50. Vandic, D., Frasincar, F., Kaymak, U., Riezebos, M.: Scalable entity resolution for web product descriptions. Inf. Fusion **53**, 103–111 (2020). https://doi.org/10.1016/j.inffus.2019.06.002

51. Wang, R., Li, Y., Wang, J.: Sudowoodo: contrastive self-supervised learning for multi-purpose data integration and preparation. In: Proceedings of the IEEE 39th International Conference on Data Engineering, pp. 1502–1515. IEEE, Anaheim, CA, USA (2023). https://doi.org/10.1109/ICDE55515.2023.00391
52. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: self-supervised learning via redundancy reduction. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning, vol. 139, pp. 12310–12320. PMLR, Virtual (2021)
53. Zhang, W., Wei, H., Sisman, B., Dong, X.L., Faloutsos, C., Page, D.: AutoBlock: a hands-off blocking framework for entity matching. In: Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 744–752. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3336191.3371813