# Experimental Protocol for XCS224u: D2EM Deep Blocking & Deep Matching for Entity Resolution

## Hypothesis

Traditionally Record Linkages is the standard method was either rule based or machine learning models to match entities utilizing supervised approaches. In order to reduce this second order complexity for pair wise comparison, blocking has been done to cluster similar entities together. However, unlike matching, not a lot of sophisticated deep neural network model has been implemented in blocking.

In order to measure the performance of the new blocking approach in entity deduplication, we want to leverage a few of the deep learning architectures. The aim here is 2 folds of high-performance optimization with reduced potential matches for matching and high accuracy of matching.

## Data

Primarily we are aiming to build this with product dataset of Amazon – Google (https://pages.cs.wisc.edu/~anhai/data1/deepmatcher_data/Structured/Amazon-Google/exp_data/) and Walmart – Amazon (https://pages.cs.wisc.edu/~anhai/data1/deepmatcher_data/Structured/Walmart-Amazon/exp_data/). The major fields in the dataset are title, categories, brand, model no and price. The title attribute is a short sentence which will have many spelling mistakes, needs to look similar meaning (synonymous) and re-arranged words.

## Metrics

The metrics are from the python / custom packages and mainly include:
1. Average F1: The Average score of F1 for both the labels (match and no-match).
2. Average Train Time: Since optimization is also part of hypothesis, its important to measure the time taken for training.

## Models

This research proposes Deep Blocker framework that significantly advances the state of the art in applying Deep Learning to blocking for Entity Matching. These solutions do not require labeled training data and exploit recent advances in DL (e.g., sequence modeling, transformer, self-supervision). Also, it empirically determines which solutions perform best on what kind of datasets (structured, textual, or dirty).

There were primarily 2 limitation this research aims to solve:

- First, they consider only a relatively simple way to convert each tuple into a vector, namely by combining the vectors of the words in the tuple using unweighted or weighted averaging. This method is called aggregation. Such methods suffer from low recall accuracy.
- Another limitation of the existing works is that some of them require labeled data, Autoblock [here] uses this data to learn the weights for combining the word vectors based on semantics, position, and the surrounding context. Such labeled data is difficult to generate for blocking in many real-world Entity Matching scenarios.

Some of the important pointers from research was:

- It develops self-supervised solutions for blocking. The training data is generated as part of the process.
- Defining a Rich Space of DL Solutions as an architecture or framework where various blocking algorithm can be tested and verified.

The research digs deeper in various blocking techniques:

- Attribute equivalence, hash, and sorted neighborhood:
  Attribute outputs a pair of tuples if they share the same values of a set of attributes. Hash blocking (also called key-based blocking) is a generalization of AE, which outputs a pair of tuples if they share the same hash value, using a pre-specified hash function. Sorted neighborhood outputs a pair of tuples if their hash values (also called key values) are within a pre-defined distance.
- Other blockers include similarity, rule-based, and composite blocking: Similarity blocking is like AE, except that it accounts for dirty values, misspellings, abbreviations, and natural variations by using a predicate involving string similarity measures, such as edit distance and Jaccard distances. Rule-based blocking employs multiple blocking rules, where each rule can employ multiple predicates.
  Composite blocking (e.g., canopy blocking) generalizes rule-based blocking and can combine arbitrary blocking methods.
- Schema agnostic tokenization a.k.a. token-based blocking
- Meta-blocking techniques improves upon this process by constructing a blocking graph and using it to discard redundant comparisons.
- DeepER: This computes tuple vector via unweighted aggregation of the vectors of the individual words.
- AutoBlock: It improves upon DeepER as by using a set of tuple pairs labeled as match/no-match to learn the weights for the aggregation.

Architecture Template:

| **Template Module** | **Choices** | |
|---|---|---|
| Word Embedding | Granularity<br>(1) Word<br>(2) Character | Training<br>(1) Pre-trained<br>(2) Learned |
| Tuple Embedding | (1) Aggregation based<br>(a) Simple average<br>(b) Weighted average (e.g., SIF)<br>(2) Self-supervised<br>*Many choices exist for each type of auxiliary tasks:*<br>(a) Self-reproduction<br>(b) Cross-tuple training<br>(c) Triplet loss minimization<br>(d) Hybrid | |
| Vector Pairing | (1) Hash (e.g., LSH)<br>(2) Similarity based<br>(a) Similarity measure: Cosine, Euclidean<br>(b) Criteria: Threshold, KNN<br>(3) Composite | |

The template provides various options like word vs character level embeddings or leveraging pre-trained vs learned embeddings. Also, when the embedding is generated for each word or characters, the tuple embedding choice is either aggregation or self-supervised methods. The aggregator then is feed to encoder-decoder architecture.

| Auxiliary Tasks | Solution Architectures | Instantiation Examples |
|---|---|---|
| Self-Reproduction | Aggregator + Encoder + Decoder | • Aggregator: SIF, LSTM, . . .<br>• Encoder/Decoder: Feed-Forward NN, LSTM, Transformer |
| Cross-Tuple Training | Aggregator + Summarizer + Classifier | • Aggregator: SIF, LSTM, . . .<br>• Classifier: Feed-Forward NN, LSTM, Cosine |
| Triplet Loss Minimization | Aggregator + Loss Minimizer | • Aggregator: BERT, . . . |
| Hybrid | Many possible architectures exist, e.g.:<br>• encoder + summarizer + classifier<br>• SBERT aggregator + summarizer + classifier | • Each architecture has many components<br>• Each component has many instantiation choices as listed above |

There are primarily two tuple embedding choices and these are aggregation methods and self-supervised methods. The paper delves in detail to discuss weighted and unweighted aggregation strategies. Also, it considers four types of auxiliary tasks in self supervised methods like self-reproduction, cross-tuple training, triplet loss minimization, and hybrid.

## General Reasoning

*{Entity* Recognition*} = {Blocking} + {Matching}*

Entity Recognition from a variety of fields, including databases, machine learning, natural language processing and information retrieval. Ironically, different subdisciplines refer to it by a variety of names, including record linkage, deduplication, co-reference resolution, reference reconciliation, object consolidation, identity uncertainty and database hardening.

Despite the long history of work on ER there is still a surprising diversity of approaches – including rule-based methods, pair-wise classification, clustering approaches, and richer forms of probabilistic inference – and a lack of guiding theory.

Naive ER algorithms that compare every pair of mentions is O(n^2). The paper reviews efficient indexing, blocking, and message passing techniques, that can reduce the complexity to near linear time. In addition, distributed computation can also significantly improve scalability of ER algorithms, and the research review recent work on distributed ER.

## Summary of Progress

So far, I have done following:

- Data Acquisition
- Data Processing (Remove unnecessary tokens and cleansing)
- Applied simple minLSH for hashing and cosine similarity as a mechanism for the baseline blocking algorithm.
- Yet to apply other DL based algorithms.

## References


1. GetoorL. et al. Entity resolution: Theory, practice & open challenges; 2012 (here)
2. R Baxter, P Christen, and T Churches. A Comparison of Fast Blocking Methods for Record Linkage; 2003 (here)
3. V Verroios, H Garcia-Molina. Top-K Entity Resolution with Adaptive Locality-Sensitive Hashing; 2019 (here)
4. LiY. et al. Deep entity matching with pre-trained language models; 2021 (here)
5. Deep Learning for Blocking in Entity Matching: A Design Space Exploration (here)
6. https://github.com/anhaidgroup/deepmatcher