A Textual Entity-Matching Implementation for Zalando–About You Product Offerings Using Deepmatcher

Patrick Travers 21267100
Dublin City University
CA684 Machine Learning 2022
Patrick.travers3@mail.dcu.ie

ABSTRACT

Entity matching is the process of identifying entities in different data sources which refer to the same real-world object. This research examines an entity-matching problem instance where entities are matched across product offerings from two e-commerce companies: Zalando and About You. Within entity matching-related literature, deep-learning solutions drawing on pre-trained language models represent the state-of-the-art. This is particularly so with unclean data, which is why deep learning is necessary in this case. The solution offered takes three stages: preprocessing, blocking and training. The final model is an ensemble of three Hybrid models from the Deepmatcher package drawing on a fastText language pre-trained on German Wikipedia. Each model specialises in matching products of brands where the cartesian product of inter-shop same-brand offerings is within certain bounds: M_{small} for small brands, M_{medium} for medium-sized brands, and M_{large} for larger brands. Positive matches are those returned with P(match) > 0.5 for each of these three models. On a small post-prepreprocessing, post-blocking test dataset, the ensemble model achieves an F_1 of 0.43. This result is contextualised within the larger picture of model performance, followed by a brief discussion of the many unexplored avenues remaining open for further investigations.

KEYWORDS

Entity matching, Record linkage, Data matching, Entity resolution, Deepmatcher, Language model, fastText, Zalando, About You

1 Introduction

Entity matching (also sometimes referred to as record linkage, data matching, entity resolution and other terms) is the process of identifying entities from different data sources which refer to the same real-world object or object-class. As an example, consider two companies undergoing a merge. Each company has a table listing all their customers. If these two tables are to be integrated, it should be done in such a way to maintain data integrity; a single customer should be represented by no more than one record. This means if someone happens to be a customer of both companies, they need to be identified in order to avoid post-merge duplications. Typically, the difficulty in this task is caused by different data sources representing the same entity in different ways. For example, it may be difficult to recognise that 'J Doe' with address '15 Pudding Lane, Southwark', represents the same individual as 'Janet D' of '15 P Lane, South London'. Entity resolution is the task of resolving such discrepancies to find true matches. In some cases, it is non-trivial task even at the level of human judgement.

While the above example of record linkage for the purposes of data deduplication is a common application area, entity matching problems take many guises in many different application areas. This paper examines the case of textual data scraped from the websites of two competing e-commerce companies: Zalando and About You. The offerings of these companies have many items in common. Looking at the problem from the perspective of Zalando, the benefit of entity matching with About You is to identify and reduce the number of instances where About You is beating Zalando's price for the same product. This paper will introduce a solution implementation to this particular problem instance.

1.1 Dataset

The entity matching problem in focus relates to product offering data for Zalando and About You—two German fashion e-commerce companies. The training dataset comprises 41k Zalando product offerings and 62k offerings from About You. As normal with entity matching problems, each set offerings contains both matches and non-matches. This dataset contains one-to-one matches only, of which there are 15,170

There are 10 data fields available for all products—Zalando and About You alike (referred to in this report with *italics*). Table 1 gives an overview of available attributes. The most informative textual attributes of *brand, color, title*, and *description* are each inconsistently formatted in various respects, giving rise to the difficulties of this task. For *brand, color*, and *title*, the content and formatting are inconsistent. *Description* is dollar-sign delimited key-value pairs for Zalando, and a mixture of JSON and html for About You. For both shops, keys available in *description* vary across offerings.

nformation regarding which records are involved in one-to-one ground-truth matches is contained in a separate dataset. The *offer_id*s of each matching pair are provided, along with an integer denoting ground-truth brand. No textual representation of ground-truth brand is given. Comparing text attributes *brand, color, title*, and *description* across brand fields reveals the full extent of

Field Name	Datatype	Comments
offer_id	String	Unique offering identifier
shop	String	'zalando' or 'aboutyou'
lang	String	'de' as dataset is in German
brand	String	Inconsistently formatted
color	String	Inconsistently formatted
title	String	Inconsistently formatted
description	String	Semi-structured information
price	Float	Price of offering.
url	String	URL of offering's webpage
Image_urls	String	URL of offering images

Table 1. Overview of available data

inconsistencies across brands for the same product. Even in the case of *brand* which one might expect to be relatively clean there are some significant differences. For example, there are 160 matches where Zalando gives *brand* as 'LASCANA' and About You gives nothing remotely resembling that value. This is but one example of many inter-brand inconsistencies.

1.2 Literature Review

Given the benefits of high-quality entity matching solutions, it has naturally been the focus of many different machine learning attempts [1]. Upon its release in 2018, Deepmatch became the state-of-the-art on entity-matching problems, performing with an F_1 score upwards of 0.9 on many standard benchmark entity matching datasets [2]. As typical with deep-learning solutions for textual data, performance is particularly strong in Deepmatcher compared to other approaches in cases of 'dirty' data, where information relevant to one data field may appear in the other dataset in an entirely different field. This is a characteristic of the current dataset, which suggests Deepmatcher may be primed for strong performance on the task.

Deepmatcher formulates the entity matching task as a pairwise binary classification task with a logistic loss function. The solution architecture embodies a multi-layered perceptron using HighwayNet. followed by a softmax layer to complete the classification component. Firstly, the attribute embedding stage brings textual data from a sequence of words to a sequence of word embeddings. Secondly, the attribute similarity representation stage converts these embeddings to attribute similarity scores. In the final stages, attribute similarity is aggregated to overall entity similarity, before a final sofmax step to determine matching probability.

There are four different available types of Deepmatch models: smooth inverse frequency, recurrent neural network, attention, and hybrid (a sequence-aware variant of attention). Although different types of Deepmatch model have been found to perform better on different datasets, Hybrid models often perform best and where beaten are usually comparable in performance with the strongest model.

More recently, an alternative deep learning-based solution has been forwarded claiming to outperform Deepmatch in some instances. Ditto is a language-model entity matching solution whose innovations include novel data augmentation strategies, injection of domain knowledge on top of word embeddings, and the relaxation of one of Deepmatcher's more restrictive requirements—that each dataset under comparison should have like-for-like data fields available [3]. Ditto does not require matching columns, enhancing its applicability to datasets for which Deepmatcher is wholly unsuitable.

In Deepmatcher and Ditto alike, the driving force behind model performance is the use of pre-trained language models. Deepmatcher's default choice is fastText, whereas Ditto opts for BERT. Such models offer the ability to move incorporate vocabulary semantics into text similarity rather than more simplistic bag-of-words style approaches. It is this characteristic, along with the flexibility of matching words across data fields, that give Deepmatcher and Ditto alike an advantage over foregoing entity matching solutions.

2 Methodology

The solution to the problem as it has been outlined was implemented in three steps: preprocessing, blocking, and matching. The measures taken in each are outlined below.

2.1 Pre-processing

Deepmatcher and Ditto alike, the driving force behind model performance is the use of pre-trained language models. Deepmatcher's default choice is fastText, whereas Ditto opts for BERT. Such models offer the ability to move incorporate vocabulary semantics into text similarity rather than more simplistic bag-of-words style approaches. It is this characteristic, along with the flexibility of matching words across data fields, that give Deepmatcher and Ditto alike an advantage over foregoing entity matching solutions.

Of the available attributes listed above, only five are of interest for the purposes of text-based entity matching: *brand, color, title, description*, and *price*. The pre-processing steps taken for each attribute for both Zalando and About You product offerings are outlined in Table 2 and Table 3 respectively.

brand	Text deaccented and lowercased. Everything discarded except the first word.
color	Text deaccented and lowercased. Non-alphanumeric characters replaced by spaces. 'dunkel' and 'hell' removed.
title	Text deaccented and lowercased.
description	Keys discarded from the dollar-delimited key-value pairs. Values kept. Text deaccented and lowercased.
price	Rounded to 2 decimal places.

Table 2. Zalando training data pre-processing measures.

brand	Everything discarded except alphanumeric characters and whitespace. Text deaccented and lowercased. Everything discarded except for first word.
color	Text deaccented and lowercased. 'dunkel' and 'hell' removed.
title	Apostrophes removed. Text deaccented and lowercased.
description	Keys disregarded and values kept from the available JSON. Everything discarded from available HTML except text enclosed within <td> tags. Text deaccented and lowercased.
price	Rounded to 2 decimal places.

Table 3. About You training data pre-processing measures.

The reasoning behind text deaccenting and lowercasing is to allow exact matching with greater ease. The level of inconsistency in capitalisation in the dataset is such that this lowercasing is essential. 'Deaccenting' was carried out using Python's Unidecode package. This package permits translating the nuances of German text such as umlauts (ä, ë, ï, ö, ū) and the long 's' (ß) into Unicode equivalents. This measure also facilitates more exact word-for-word matching because these nuances are used inconsistently both intra- and inter-brand.

2.2 Blocking

Deepmatcher documentation outlines that a key step in its implementation is the 'blocking' stage, where 'obvious non-matches' are ruled out in order to reduce the matching shortlist to a 'reasonable' size [4]. The different blocking measures implemented are outlined in Table 4. Note: at the time of blocking, preprocessing as outlined above has already been completed.

2.3 Training

Training data was bagged into three groups, $Dataset_{small}$, $Dataset_{medium}$ and $Dataset_{large}$, depending on the size of the brand's inter-shop cartesian product. The positive-negative ratio of matches within these post-blocking bags was approximately 6, 12, and 16 respectively. These differences in data were reflected in Deepmatcher's pos_neg_ratio hyperparameter at training time. Each dataset was further separated into training, validation, and test sets. Three models M_{small}, M_{medium} and M_{large} were trained from training on each of these datasets respectively. Each model is an instance of Deepmatcher's hybrid model, with a pre-trained fastText German language model based on German Wikipedia. Each model was trained for 10 epochs. The iteration with the best F_1 was kept. At inference, the outputs of these models are combined by unanimous voting, where a predicted match must be predicted as such be each of the three models.

brand	Must be identical
color	About You color must contain a substring of any word from Zalando *color*
title	About You *title* must contain a substring of any word from Zalando *title* OR Zalando *title* must contain a substring of any word from About You *title*
description	Must have 6 or more words in common
price	Must have a difference no larger than 2.5

Table 4. An overview of blocking measures.

3 Results

A sample of results as obtained on the test set of $Dataset_{small}$ are given in Table 5.

With the aim of maximising F1 the main issue lies in an abundance of false-negatives. In each model recall is high at the cost of a large amount of non-matches being reported as matches. This was a clear trend even from very initial investigations with individual models, and it was hoped that ensembling with unanimous voting may offer an avenue around this. However, the ensemble's greater precision comes at such a large cost to recall that on this dataset, the ensemble does not score above the single model Msmall on F1 score. Msmall's strong performance on this test set, beating the ensemble model, should not be entirely unexpected, due to the fact that this test set is directly related to the training and validation sets which were used to train Msmall. However, the ensemble model is still the preferred overall solution rather than any single model. There are two reasons for this. Firstly, the ensemble performs stably across brands of various sizes, whereas Msmall, Mmedium and Mlarge exhibit strong performance in similarly-sized brands to their own training sets, and relatively weak performance elsewhere. The second reason is that this solution aims to predict matches among brands which is has never encountered before i.e. brands that were not included in any of Datasetsmall, Datasetmedium or Datasetlarge. With this in mind, it is hoped that the variance gained by combining predictions from three models trained on three different sets of brands should allow the resulting ensemble to perform better when encountering inputs from unfamiliar brands.

Some experimentations were carried out with various ensemble voting policies. None were found to beat unanimous voting. The performance of majority voting on the $Dataset_{small}$ test set were 0.28 precision, and 0.83 recall for an F_1 of 0.42. This trend held consistently across training sets: majority voting performs better than individual models, but when compared to unanimous

	M_{small}	M_{medium}	M_{large}	ensemble
Precision	0.34	0.27	0.2	0.39
Recall	0.69	0.74	0.8	0.48
F_1	0.45	0.39	0.32	0.43

Table 5. Results of inference on hold-out test set of $Dataset_{small}$

voting, the gain in recall is not worth the reduction in precision when it comes to calculating F_1 score.

4 Discussion

As is always the case with a dataset lacking in benchmark figures, it is difficult to fully contextualise the results achieved in this solution. Suffice to say the F_1 score achieved is far below Deepmatcher applied to other datasets [2]. Even the results reported are not an evaluation of the model across the entire labelled dataset; these figures relate only to the test set of $Dataset_{small}$. Furthermore, it must be pointed out that $Dataset_{small}$ during this inference had already gone through preprocessing and blocking, so the figures reported here do not account for true matches that were ignored and lost at this pre-training stage. The F_1 score on the entire labelled pre-blocking dataset will doubtless be considerably lower than those given here. Some general suggestions as to why the model's performance should be relatively poor, along with some points regarding other potential avenues left unexplored will be offered presently.

Firstly, the approach to brand-matching at the blocking stage is quite naïve. While the current approach captures approximately 97% of ground-truth training matches, there are some obvious examples which are missed by the current blocking method. For example, Zalando's 'Pieces' and About You's 'Little Pieces' are currently being ignored at the training stage. Given that *brand* forms the basis for the pre-pruned cartesian product, more care should be taken at this stage to ensure an optimal solution. Perhaps an alternative albeit more computationally intensive option may be to attempt text clustering. Such an effort may rely on *brand* and yet draw on information from other fields. A sophisticated solution may be able to detect subtle inter-brand differences in attributes other than *brand* which would probably go unnoticed even on human inspection.

Another area where improvements may be made through relatively simple processes of experimentation, and trial and error is in the chosen emsembling method. As described above, the solution takes the approach of using predictions of three different models: M_{small}, M_{medium} and M_{large}, and match is determined by a unanimous agreement between these models. Models are built on specific sets of brands based on the size of the inter-shop cartesian product of that brand (small, medium, or large). However, there may be a way around this, and to build flexible models which work in predicting brands of any size, even with various positive-negative ratios. There are doubtless innumerable other ways these models could combined. The current method of combination seems to perform relatively well empirically, though increasing the number of constituent models, adjusting the bagging approach, or otherwise adjusting the voting and/or thresholding systems may all be able to contribute to improved performance.

The preprocessing steps described were carried out to facilitate a greater number of textual inter-brand exact matches. However, it may be that the severity of this preprocessing may be hampering the ability of fastText to properly compute word embeddings. It may be worthwhile to investigate whether less intensive preprocessing may improve the language model's contributions. Furthermore, trials could be carried out to investigate whether other language models than fastText's German Wikipedia may be more suited to this problem instance.

Although the current solution aimed to implement a text-based solution, there is a significant untapped resource in the form of product offering images. It is rare in entity matching literature for images to be incorporated within the process. However, in this case they may be a boon, and various avenues should be explored to establish how matching scores could be achieved in a computationally feasible way, and incorporated with the textual component of the solution.

As already noted, the most significant barrier in maximising F_1 lies in the abundance of false-positives reported. A simple option towards reducing this issue may be to adopt even more aggressive strategies at the blocking stage. Although the blocking policies as outlined in Table 4 seem to stand to reason, from a results-maximising perspective it may be worthwhile to gauge the effects of different strategies on final F_1 score. Furthermore, an alternative means of blocking may facilitate the processing of the largest brands in the dataset—due to the scale of the cartesian product for 'vero' in the preprocessed labelled dataset, and 'tom' in the preprocessed unlabelled set, have been ignored for training and inference respectively to prevent system crashes. This problem is far from intractable and could be overcome with some adjustments.

For a full analysis of the Deepmatcher suite's performance on the Zalando dataset, the other available models should be trialled, namely the SIF, RNN and attention models. While the Deepmatcher hybrid model generally performs well in most scenarios, it is certainly worth investigating whether other variations may perform better in this instance. It must be pointed out that while *description* was used at the blocking stage, it was not used at training stage because it prevented model training to any effective degree. Since the hybrid Deepmatcher model takes account of word-orderings which is not a characteristic of *description*, perhaps a different Deepmatcher model would be better suited to make use of this attribute. After all, there is lots of information available in the *description* attribute, it certainly appears wasteful to completely discard it at training time.

It should be pointed out also there are many other options for this task beyond those available in Deepmatcher. As has been outlined in the foregoing literature review, there are other deep learning-based solutions which may be well disposed to strong performance on this dataset. Most notably, Ditto has exhibit high F_1 scores on unclean data, and has some distinct advantages over Deepmatcher in data augmentation, domain knowledge injection and asynchronous field facilitation [3].

4 Conclusions

This research has focused on an instance of entity matching across Zalando and About You product offerings. Following the necessary steps of preprocessing and blocking, three hybrid

Deepmatcher models drawing on fastText's pre-trained German Wikipedia language model were trained. Each model was built to specialise in brands of various inter-brand cartesian product sizes: M_{small}, M_{medium} and M_{large} for small, medium, and large brands respectively. While each model performs best within its own domain of training, the best generalisable solution is to take unanimous votes, where matches must be classified as such by each of the three models. On a small hold-out test set of $Dataset_{small}$ the ensemble model achieves an F_1 score of 0.43. Even though this score is elevated by the fact that it was calculated on a post-blocking dataset, the score is still well below typical Deepmatcher scores on comparable datasets. A number of potential adjustments to improve these results have been discussed: adjustments in relation to preprocessing, blocking, bagging, ensemble voting policy, inclusion of image data, incorporation of 'vero' and 'tom' brands as well as the *description* field, Deepmatcher configuration and hyperparameters, and choice of language model are all worthy of further investigations in improving the offered solution. Furthermore, given the strength of its performance on benchmarked datasets, it certainly must be checked as to how the Ditto entity matching solution should perform at this task. Given the comparably poor results achieved here, it is imminently possible that switching to Ditto-based models may offer an immediate improvement on the reported results.

REFERENCES

[1] Barlaug, N. and Gulla, J.A. 2021. Neural networks for entity matching: A survey. *ACM transactions on knowledge discovery from data*. 15, 3 (2021), 1–37. DOI:https://doi.org/10.1145/3442200.

[2] Mudgal, S. et al. 2018. Deep learning for entity matching: A design space exploration. *Proceedings of the 2018 International Conference on Management of Data* (New York, NY, USA, 2018).

[3] Li, Y. et al. 2020. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*. 14, 1 (2020), 50–60. DOI:https://doi.org/10.14778/3421424.3421431.

[4] Notebook on nbviewer: *https://nbviewer.org/github/anhaidgroup/deepmatcher/blob/master/examples/end_to_end_em.ipynb*. Accessed: 2022-03-30.