Beyond Fuzzy Matching: Effective Ways To Transfer Learning in NLP

Sruthi Poddutur
Stanford University, CA
poddutur@stanford.org

Abstract

This paper attempts to solve Fuzzy Matching for Named Entities like Organisation names which is a long-standing Natural Language Processing(NLP) problem useful for Data Integration particularly when data coming from multiple external sources. Interestingly, there seems to be not much work published on this very task making use of transfer learning (TL) from state-of-the-art Transformer architectures such as BERT. The work presented here takes BERT as pre-trained model from which TL is evaluated. Work explores 1. Feature-Based TL and 2. Fine-Tune Based TL from BERT. Novelty presented here dwells into comparing the nature of knowledge that Image-TL (vs) NLP-TL offers, how lack of hierarchical structure in NLP-TL's knowledge poses problem and proposes a work-around solution to it that achieves 90% test accuracy. Work also demonstrates how NLP-TL made it easy to achieve beyond simple fuzzy matching for organization names. Work concludes by establishing that 1. It is very sensitive to InputFormat and Encoding Schemes, 2. There is huge scope to trim the model without hurting the performance and 3. NLP-TL still has some journey ahead to attain the maturity at the levels of Image-TL.

1. Introduction

With the advent of big-data and Internet of Things, different types of structured and unstructured large amounts of data got mined. Data Integrating via Fuzzy Matching became even more significant means to process all of that data in order for us to make use of it in decision support. With fuzzy matching, we do not look for exact matches but we look for similar values. Despite being a long-standing problem, we are still witnessing research publications trying to address this very problem of approximate matching. Problem investigated in this work explores effective ways to make use of transfer learning(TL) in NLP for the task of Fuzzy matching using transformer architectures like BERT. In particular, this work attempts fuzzy matching for Named Entities especially organisation names. Given two organisation names as input, the model should be able to classify them as match or not. Few examples of organisation fuzzy matches are: *(IBM, International Business Machines), (Seven Eleven, 7 Eleven), (Make n Mold, Make and Mold)* etc. Training data was obtained from Wiki and DBPedia. For baseline, SVM model with edit-distance features is trained. Work also trains multiple DNNs to study the inherent nature of Bert embeddings with 1. Feature-Based NLP TL from Bert (having different input formats and encoding schemes) and 2. FineTune-Based NLP TL from Bert (to learn embeddings that are either true representation of entities or help in fuzzy matching).

[0]Code link: https://github.com/spoddutur/cs229-project

Beyond Simple Fuzzy Maching Positive Predictions Due to Wanted Knowledge from NLP TL		
OrgName1	OrgName2	Relation
Quaker Oats	PepsiCo	Aquired in 2001
US Oncology	McKesson	Aquired in 2010
Cimpor	Cecisa Comercio Internacional	Same Family
Tamoil	Libyan Nation Oil	Same Family
TSG Consulting	Transportes Sousa Gomes	Abbreviation
Excalibur Resources	Metalla Royalty Streaming	Org Rename
CMP INFO	UBM Intermediate	Org Rename

Table 1: Interesting Positive Samples labelled correctly by the work's final model (in Fig2) making use of the knowledge of wanted relationships in pre-trained BERT such as acquisitions (as old as 2001), same family tree and organization rename

Novel contributions of this work:
1. *Beyond Fuzzy Match* - A boon from knowledge reuse in NLP TL is that it has the knowledge of complex relations like $Acquisition$, $FamilyTree$, $OrgRename$. Some interesting positive predictions done by the model that refelect this knowledge are listed in Table1.

2.*NLP-TL onboards both wanted and huge amount of unwanted Knowledge unlike Image-TL:* One core reason for the success of CNNs architecture is that they enabled Image-TL with the knowledge that has a nice hierarchical structure (where lower layers learn generic edges and curves and higher layers learn task-specific features). One can pick and choose among the generic low-level or high-level task-specific features to reuse in Image-TL. However, NLP-TL knowledge (from pre-trained Bert Model [1] in this case)

Incorrect Model Predictions Due to UnWanted Knowledge from NLP TL		
OrgName1	OrgName2	Relation
Penn State Univ	Kent State Univ	SI,SC
Univ of Iowa	Univ of Utah	SI,SC
Amazon	Microsoft	Competitors
Airtel	Vodafone	Competitors
Polytecnic Inst NYU	Worcester Polytecnic Inst	SI,SC
Towa Bank	United Western Bank	SI,DC

Table 2: Samples of incorrect predictions (as fuzzy match) by the work's final model(Fig2) making use of knowledge of unwanted relationships in pretrained-BERT like SI=SameIndustry,SC=SameCountry,DC=DifferentCountry,Competitors.

do not have such hierarchical structure to it. Therefore, NLP-TL on-boards both wanted knowledge as well as much bigger unwanted knowledge. User cannot pick and choose only the wanted knowledge to reuse in NLP-TL. For this task, this unwanted knowledge that led to wrong predictions constitute Competitors, Same Industry, Shared-location etc relationships as listed in Table 2.

3. *Cause for no hierarchy in NLP-TL Knowledge*: As discussed above, Knowledge from NLP-TL does not have hierarchical structure to it and the reason is the very design of the transformers which is based solely on attention mechanism. Hence, NLP-TL has barely scratched the surface and needs more cultivation in this direction. It has some journey to travel before it could attain the maturity at the levels of Image TL.

4.*Pre-process block:* Because NLP-TL has the lack of control on selecting specific knowledge to reuse, there is a forcing need to have a pre-blocking component to filter all of the unwanted knowledge.

Other Findings:

1.*Feature Based TL is sensitive to InputFormat and EncodingScheme:* $\langle CLS \rangle\, Orgname1\, \langle SEP \rangle\, Orgname2\, \langle SEP \rangle$ input format with Mean of the last 3 layers encoding scheme gave the best result with feature-based TL. However, even with this combination it could not out-beat a simple baseline model with barely 4 edit-distance features. Both gave same 85% accuracy.

2. FineTune TL has faster convergence and has more learning ability than Feature-based TL.

3. FineTune based model could be trimmed to half the size without hurting performance

4.Differential learning rate based fine tuning did not help much in performance for this task

5. Contrastive loss was used to try and learn true embeddings to entity. But, it is observed that without harnessing much harder triplets it might not be good at this task

2. Related Work

Early Work on fuzzy matching typically used Rule-based solutions [7, 16] which are interpretable but require the heavy involvement of a domain expert. Later came locality-sensitive hashing methods which in general provide entities with blocking keys/signatures in such a way that similar entities have identical signatures with high probability [11, 14]. The most recent work on this task focused on providing deep learning (DL) solutions [6] where it designed a pairwise binary classification task using logistic loss; Triplet Learning [9] which learns good embeddings for objects using triplet loss and Siemese networks [13] which tries to learn good entity embeddings in vector space using contrastive loss. Some other DL solutions include proposing a scoring technique for company names in [8], RNN based classifiers [12]. However, until NLP-TL moment arrived, all these solutions had to learn their layer's features from scratch. With the recent advancement of transformers [17], NLPs ImageNet moment has arrived i.e., they enabled transfer learning in NLP. New challengers such as Elmo [15], ULMFit [10], Bert [5] etc made headlines providing pertained models that achieved state-of-the-art results on a wide range of NLP tasks. However, there seems to be not much work published on Fuzzy matching for Named Entities using Transformer architectures.

3. Data Collection

A total of 20,000 training data and 5,000 test data was collected for this task. Positive samples are collected from WikiData (Example Entity: https://www.wikidata.org/wiki/Q37156) and DBPedia with custom sparql queries [3] to extract Names and Aliases of organisations. 10,000 strategic negative cases were curated using (a) Random combinations of organizations For example, (Microsoft, Apple) is one such negative case of this category and (b) Interpolating different name parts and designation parts. For example: (AIG Insurance, Oriental Insurance) and (Atlantic Traders, Atlantic RealEstates) are two such negative cases of this category. Also, to curate such negative cases, data was pre-processed to extract designation part and name part given an organization name. Table1 and Table2 lists sample data.

4. Method - ML Techniques tried

The main intent behind the experiments conducted in this work is to *compare and analyze the nature of various forms of transfer learning in the latest transformer-based models* and Bert is the chosen transformer-based model for this analysis in this work. Experiments attempted in this work can be categorized broadly into three: *Baseline, Feature-Based Transfer Learning and Fine-Tuning based Transfer Learning*.

Consider a neural network function $\mathcal{N}$ with parameters w: $\mathcal{N} = \phi_w(x)$. Transfer learning from neural network $\mathcal{N}$ to learn a new task is to compose it with a new function, ψ_v, to yield $\psi_v(\phi_w(x))$. Here v constitutes the new task-specific parameters and w constitutes the original parameters of $\mathcal{N}$. Feature-based transfer learning is an approach

FeatureName	Mean of positive samples	Mean of negative samples
LevenshteinDsitance	-0.114339	0.092912
Jaccard	0.211604	-0.171949
Jaro	0.449772	-0.365484
EditDistance Ratio	0.327430	-0.266069

Table 3: Mean Differences of features between positive and negative samples illustrate presence of nice patterns captured by them for the model to learn.

where we freeze w and train only v *vs* Fine-tuning transfer learning is an approach where both w and v form the trainable parameters.

For Baseline, this work will start off with the most common solution for this problem where a simple statistical model such as SVM is trained using simple string edit distance features. Later, this work will move to current advanced attention-based transformer architectures such as BERT *et al*. [5] and attempt Feature-Based and Fine-Tuning based transfer learning methods on top of BERT to evaluate how its pre-trained embeddings [1] will perform for this task. It was also observed that the model was over-fitting data and so, Early Stopping and Dropout techniques were used.

5. Experiments

In all these experiments, OrganizationName1 and OrganizationName2 strings are inputs for this task of fuzzy matching and output is a confidence score indicating how close(fuzzy match) are the two given input organization names. *Binary CrossEntropy* loss and *Adam* optimiser are used in all DNN's attempted in this work. Also, over-fitting was observed in these DNN models. So, *Dropout and Early Stopping techniques* were employed to stop the network if there is no improvement in validation accuracy within 20 epochs.

5.1. Experiment1

BaseLine Model (Linear and RBF SVM): For this experiment, four edit distance based features are used: Jaccard, Levenshtein, Jaro, EditDistance. To compute Levenshtein, Jaro, EditDistance features, a git resource [2] was leveraged and code was written to compute Jaccard distance feature. Mean differences in features between positive and negative samples was computed (as shown in Table 3) to cross check that the generated features have some meaningful patterns for the model to learn and classify. Grid search was performed to find the best hyper parameters.

Results: Table4 lists the results of Linear and RBF SVM models. Clearly, RBF SVM performed better than Linear SVM in Accuracy, Precision and Recall metrics. However, the recall of RBF SVM is same as Linear SVM.

Analysis: Upon further analysis, it was observed that edit-distance features could not label cases like abbreviations (ex: IBM and International Business Machines), popular aliases (ex: big blue and IBM) and partial popular organisation names (Ex:Disney and Walt Disney)

Name	Train Acc	Test Acc	Test Precision	Test Recall
Linear SVM with C=1	78.86	79.49	76.40	76.85
RBF SVM with C=10	85.93	85.50	88.19	77.44

Table 4: Results of RBF vs Linear SVM model Based on Edit Distance Features.

DNN1 Architecture	**DNN2 Architecture**	**DNN3 Architecture**
F1, E1	F1, E1	F1, E1
Dense(10)	Dense(256),Dropout(0.1)	Dense(256),Dropout(0.2)
Dense(1)	Dense(10),Dropout(0.1)	Dense(10)
	Dense(1)	Dense(1)
DNN4 Architecture	**DNN5 Architecture**	
F2, E1	F2, E2	
Dense(256),Dropout(0.2)	Dense(256),Dropout(0.2)	
Dense(10)	Dense(10)	
Dense(1)	Dense(1)	

Table 5: Architecture of DNN1 to DNN5. F1(Format1)=CONCAT(Bert-Embedding(OrgName1), Bert-Embedding(OrgName2)). F2(Format2) = $\langle CLS\rangle\, OrgName1\, \langle SEP\rangle\, OrgName2\, \langle SEP\rangle$. E1(Encoding Scheme1) = Berts last Encoder Layer embedding. E2(Encoding Scheme2) = Mean of Berts last 3 Encoder Layers embedding

5.2. Experiment2

Feature-Based Transfer learning DNN using Bert's Pretrained Model: In this category, Bert is our neural network $\mathcal{N}$ from which transfer learning is attempted. As we are attempting feature-based learning Bert's weights are frozen in this case. The experiments tried in this category mainly differ in the input data format and what encoder layers output from Bert are consumed. The two input data formats tried in these experiments are:

1. *Format1:* Get BertEmbeddings for *OrgName1* and *OrgName2* input strings; Concatenate these embeddings and pass it to Dense layers down the lane.
2. *Format2:* Get BertEmbedding for input string $\langle CLS\rangle\, OrgName1\, \langle SEP\rangle\, OrgName2\, \langle SEP\rangle$ and pass it to Dense layers down the lane.

The two ways in which encoder outputs are consumed are:

1. *EncOutput1:* Take Last Encoder Layer Output
2. *EncOutput2:* Take Mean of Last 3 Encoder Layers

DNN1, DNN2 and DNN3 are the three different DNNs tried with fixed input format and encoder output i.e., *Format1* and *EncOutput1* but different architectures as briefed in Table5. Based on validation accuracy, the idea was to pick the best out of these three architectures for further analysis where input format and encoder output are changed to compare and analyse their effects on model performance. DNN3 performed marginally better among the three with 80% validation accuracy. So, DNN3's architecture was adapted for further analysis in DNN4 and DNN5 experiments i.e., DNN4 and DNN5 use the same network architecture as DNN3 but the input format and encoder output are changed to *Format2* and *EncOutput2*. Table5 shows the architecture of DNN4 and DNN5 respectively.

Results: Figure 1 illustrates the Train and Validation Accuracy curves of DNN1, DNN2, DNN3, DNN4 and DNN5. Table6 lists the accuracy numbers for these 5 mod-

Model	Epochs to Converge	Train Acc	Val Acc
DNN1	6	80.4%	78.9%
DNN2	32	92.5%	79.9%
DNN3	19	85.3%	80.2%
DNN4	18	70.05%	68.30%
DNN5	11	91.15%	85.04%

Table 6: Feature-Based Transfer Learning: Accuracies and Num Iteration took to converge for DNN1, DNN2, DNN3, DNN4 and DNN5. As EarlyStopping was employed, epochs to converge reported are different for each model. DNN5 gave the best results (but performed same as RBF SVM baseline model).

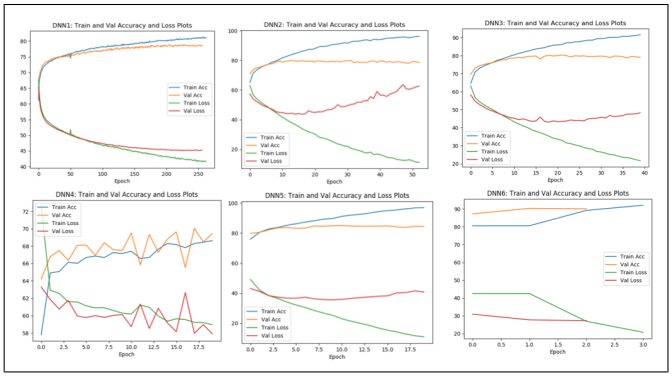

Figure 1: Left to right: Train and Test Accusracy and Loss curves for DNN1, DNN2, DNN3, DNN4, DNN5 and DNN6 respectively

els and because Early stopping was employed, the number of epochs it took for them to converge is also presented. Clearly DNN1 was simplest network with least number of trainable paramters which converged the fastest, but it could not attain more than 80% accuracy in both train and test. So, DNN2 was trained with more parameters and with equal 10% dropout at every layer and it turned out to be over-fitting, where training accuracy could go beyond 90% but validation accuracy stayed put at 80%. This is also evident from the gap between its loss and accuracy curves in Figure 1. So, the third model i.e., DNN3 was tried with more dropout percentage (= 20%) in Dense 256 Layer. This kind of put a check on the over-fitting where the gap between train and validation accuracies reduced. As all the three models gave same validation accuracy (around 80%), one of the three models i.e., DNN3's architecture was chosen for further analysis to evaluate the effect of input format and encoder outputs.

Comparing DNN3 (*Format1*, *EncOutput1*), DNN4 (*Format2*, *EncOutput1*) and DNN5 (*Format2*, *EncOutput2*) models; it is found that DNN5 with 85% validation accuracy stood first. Therefore, *Format2* and *EncOutput2*, turned out to be the best combination to use for this task with feature-based transfer learning.

Analysis - common misclassification cases: All the models trained in Baseline (Experiment1) and Feature-Based transfer learning (Experiment2) gave less than 85% validation accuracy. Therefore, validation results were further analysed to understand the common cases where they failed to predict correctly. Some such error cases

Model	Num Bert's Encoder Layers Used	Architecture Same as DNN5 With unfrozen layers	Epochs to Converge	Train Acc	Test Acc
DNN6	12	DNN5+12th LU	3	92.1%	90.06%
DNN7	1	DNN5+1st LU	3	91.39%	81.42%
DNN8	2	DNN5+2nd LU	1	91.88%	80.72%
DNN9	3	DNN5+3rd LU	3	94.83%	86.47%
DNN10	4	DNN5+4th LU	3	95.61%	86.67%
DNN11	5	DNN5+5th LU	3	94.91%	88.53%
DNN12	6	DNN5+6th LU	11	91.06%	89.53%

Table 7: (LU = Layer Unfrozen). Experiment3 DNNs attempted to trim the finetuned model. It is noticed that the model can be easily trimmed to half the size by using only 6 encoder layers of Bert which matched the performace of the model that used all 12 of Bert's encoder layers

were: *(Smiths Sheds Fencing, Simon Sheds Fencing), (Advanced Forming Research Ctr,Oxford Advanced Research Ctr), (Young Rubicam Barcelona,Vinizius Young Rubicam), (Studio 89 Prod,Studio Prod 89) etc*. Clearly, the nature of the mistakes did not seem to be complex for model to not understand. There seems to be some more scope for the model to be further improved to learn the structured patterns in these error cases.

5.3. Experiment3

FineTuning Based Transfer learning DNN using Bert: To overcome the structured failures discussed above, DNN5 - the best model from Experiment2 with *Format2* and *EncOutput2* was taken and deeper coupling with Bert was enabled i.e., Bert's last encoding layer was unfrozen and its weights were fine-tuned during training. This is our DNN6 i.e., *DNN6=DNN5 with Bert's 12th encoding layer unfrozen*. DNN6 performed very well on test data. Therefore, further experiments were conducted to trim the model and see if its possible to cut down its size without hurting its performance. Because Bert's 12 encoding layers contributed most to the model size, naturally when it came to trimming the model, the focus was on those layers. Table 7 lists the architectures of the trimmed DNN's attempted here.
Results: As shown in Table7, all the fine-tuned DNN's converged to roughly about 90% train and test accuracy in just 3 epochs. It is also found that the model could be trimmed to half in size without effecting the performance i.e., DNN12(which used only 6 encoding layers of Bert) achieved same numbers as DNN6 which used all 12 encoding layers of Bert. Lastly, clearly fine-tuning based transfer learning did out-beat feature-based transfer learning easily not only in faster convergence but also in getting higher numbers.
Analysis: Table1 and Table2 lists some interesting positive and negative predictions by the model. Bert remembered a whole lot of information about acquisitions, family tree, organization renames, etc relationships between organisations from the knowledge it obtained while training on millions of wiki data and other sources. These constitute useful patterns that Bert knows which is helping DNN6 to go beyond simple fuzzy matching. However, there are some unwanted patterns that Bert knows which led to wrong predictions such as the ones listed in Table 2. One way to fix

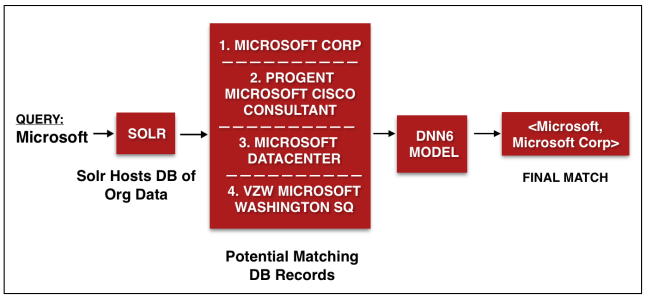

Figure 2: Final Proposed Model to FuzzyMatch a query orgName against a DB of orgs: 1. Host org DB in Solr. 2. Hit Solr with query and get candidates matches according to our requirement. 3. Pass these candidates to our DNN6 Model to find the best pair. A Simple yet very Effective solution!!

this is to understand these unwanted patterns and collate the training data to help DNN6 unlearn these patterns. However, it could be a deamon of a task to accomplish this as Bert might have lots of such unwanted patterns. Hence, a simpler alternative would be to have a pre-blocking module to counteract these unwanted learnings. Figure2 shows the *FINAL proposed MODEL* for this task which uses Solr as pre-blocking unit to filter out unwanted learnings.

5.4. Experiment4

Differential learning rates based Finetune Transfer learning using Bert: Differential Learning Rates is about using different learning rates(LR) per group of relavent layers. Rationale for this is that, lower layers of the network learn more generic features. Hence, these layers need not be disturbed much and so their LR can be lower. Also, last layers of the model (which learn more task-specific features) needs more flexibility to change and hence LR needs to be higher for these layers. In the case of transfer learning, tweaking LRs for generic-lower and task-specific higher layers depends on the data correlation between the pre-trained model and our required model. For example, if the task is to create a dog/cat classifier and our pre-trained model is already good at recognizing cats, then we can use learning rates of less magnitude. But if the task is to create a model on satellite/medical imagery then we will need slightly higher LR [4]. In this experiment, DNN6 architecture was used with Adam optimizer having default learning rate of 0.001 for final dense layers and a 100x slower learning rate of 0.00001 is applied to Bert. This is our DNN13.
Results: In just 3 epochs, DNN13 obtained 92.09% train and 89.49% test accuracy. Clearly, DNN13 matched the numbers of DNN6 in Experiment2. Hence, differential learning rate did not matter much in this case.
Analysis: Higher the data correlation between pre-trained model and our required model, lesser can be the gap in learning rate between these layers. So, this experiment confirms in a way that the fuzzy matching task of this work has good correlation with pretrained Bert's training tasks.
Contrastive Loss based Finetune TL using Bert: Contrastive Loss was used to try and learn high-quality embedding vectors that are true representation for the input (i.e., organization names). DNN14 listed in Table8 is trained for this and it could only scale upto 73% F1-Score. Upon error analysis, it was found this model failed on harder pairs such as (Ice-o-Matic, Mile High Equipment LLC) - where one organization is alias of the other; (South Georgia Cotton Gin LLC, SO GA Cotton Wholesale) - where harder abbreviations are involved. To fix this, we will need to collect more such harder data samples and is deferring for future work.

Contrastive Loss DNN14 (Layers)
emb1=Bert-Last-Enc-Layer-Embedding(OrgName1)
emb2=Bert-Last-Enc-Layer-Embedding(OrgName2)
ContrastiveLoss(Euclidean-Distance(emb1, emb2))

Table 8: Contrastive Loss based Architecture of DNN14

6. Conclusion and Future Work

Many studies are being published on a wide range of NLP tasks using transfer learning(TL) after the ImageNet moment arrived in NLP. However, little has been studied about TL for the task of fuzzy-matching on named entities. This work fills this void choosing BERT as pre-trained model to transfer learn from. To study the inherent nature of Bert embeddings, variants of Feature-Based TL DNNs were trained with different input formats and encoding schemes. Also, Bert embeddings were fine-tuned in multiple ways to learn the embeddings that are either 1. True representations of organization or 2. Help in the task of fuzzy matching organisations better. Work done so far found that the advanced attention-based architectures such as BERT needs fine-tuning for the specific problem we are trying to use. Using it as mere embeddings did not outbeat much simpler SVM model trained with right set of features for this task. Furthermore, interesting results surfaced upon analysis of the nature of knowledge NLP-TL offers. In that, the core reason for the success of Image-TL is that its knowledge bank has nice hierarchical structure inherently and one can pick and choose if he wants to reuse more generic low-level features or higher-level features according to the requirement. Because of the very design of attention mechanism, transformer architectures that enabled NLP-TL do not posses such nice hierarchical structure within its knowledge. This makes it tough to extract only the wanted knowledge from it. Hence, this work sees a necessity to cultivate NLP-TL more in that direction. Also, this very possession of unwanted knowledge in NLP-TL mandates some pre-blocking mechanism to filter out the unwanted stuff. Additionally, Differential Learning Rate did not had any impact for this task and an unsuccessful attempt was made to learn true-representation embeddings using contrastive loss.
Future work entails Contrastive-Loss based Fine-Tuning with more harder pairs and will also study the effect of Multi-task Learning for this task.

References

[1] Bert model https://tfhub.dev/google/bert_uncased_l-12_h-768_a-12/1.

[2] Ptython library for edit distances https://github.com/ztane/python-levenshtein/tree/master/levenshtein.

[3] Sparql http://dbpedia.org/sparql.

[4] Transfer learning using differential learning rates - https://towardsdatascience.com/transfer-learning-using-differential-learning-rates-638455797f00.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[6] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. Deeper – deep entity resolution. 2017.

[7] W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. *Proc. VLDB Endow.*, 2(1):407–418, Aug. 2009.

[8] T. Gschwind, C. Miksovic, J. Minder, K. Mirylenka, and P. Scotton. Fast record linkage for company entities, 2019.

[9] E. Hoffer and N. Ailon. Deep metric learning using triplet network, 2014.

[10] J. Howard and S. Ruder. Universal language model fine-tuning for text classification, 2018.

[11] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[12] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, pages 19–34, New York, NY, USA, 2018. ACM.

[13] P. Neculoiu, M. Versteegh, and M. Rotaru. Learning text similarity with Siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[14] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas. Comparative analysis of approximate blocking techniques for entity resolution. *Proc. VLDB Endow.*, 9(9):684–695, May 2016.

[15] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations, 2018.

[16] R. Singh, V. Meduri, A. Elmagarmid, S. Madden, P. Papotti, J.-A. Quiané-Ruiz, A. Solar-Lezama, and N. Tang. Generating concise entity matching rules. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 1635–1638, New York, NY, USA, 2017. ACM.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.