Find your Match: Towards Simulating User Relevance Feedback

Abstract

This paper proposes a deep learning-based approach that simulates the user-feedback loop in entity matching and relevance modeling without the need for costly and time-consuming data collection. Our method formulates the relevancy modeling problem as a text/entity matching task and leverages NLP techniques to learn matching patterns from logged interactions. We evaluate our approach on datasets from three popular domains and show its effectiveness, achieving close human performance. Furthermore, our approach demonstrates robust performance on complex negative samples, indicating its potential to benefit any domain that relies on user feedback.

1. Introduction

Online Information Retrieval (IR) and Recommender Systems face the challenge of relevance matching, which involves ranking documents or items based on their relevance to a user's query [22]. A critical aspect of relevance matching is entity matching.

Entity matching refers to the task of determining whether two different representations refer to the same real-world entity. The term "entity matching" also loosely refers to the broader problem of determining whether two heterogeneous representations of different entities should be associated together. [16]. An example of entity matching is illustrated in Figure 1, where $Water$ and H_2O refer to the same entity, but have different representations.

Grounding this problem relies on user or expert feedback. However, gathering enough user data or ensuring continued long-term engagement from users can pose significant financial and cognitive challenges. In such scenarios, a learned model for user feedback can be beneficial for recommendation or retrieval problems.

Deep learning has made significant progress in recent years in matching through deep semantic models for search and neural collaborative filtering models for recommendation, primarily due to their ability to learn representations and generalize matching patterns from raw data like queries, documents, users, and items. [10] The successes of Deep learning in semantic matching can be extended to entity matching and relevance modeling, which is of great importance to various domains that depend on user feedback. The bottleneck of many online systems depends on user feedback to continue improving their models. The application of our approach goes beyond the IR domain and broadly impacts any domain that leverages user feedback, including Reinforcement Learning and numerous applications in Medical Sciences.

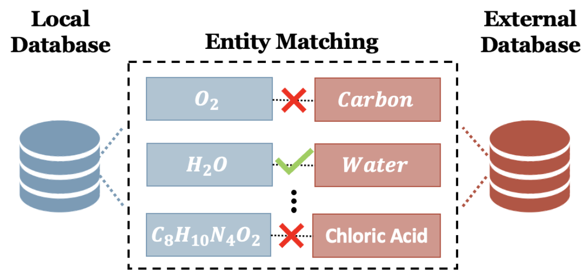

Figure 1. Entity matching for tuples from different databases: determine whether the pairs refer to the same entity.

Modeling user feedback in the context of entity matching requires more than simply learning a similarity function between entities. Because any two entities can be represented in an infinite number of ways, measuring similarity is more complex than finding common features between the two. Rather, we must determine which characteristics are relevant to the task at hand in order to more accurately measure whether two objects refer to the same entity. However, determining relevance can be difficult since what is relevant is highly dependent on the context, interests, and goals of a specific user. Simply deriving similarity between two entities is not sufficient for modeling user feedback, as it is not a unique relation. This is due to the fact that relevant characteristics often vary with respect to the user. Hence, modeling user feedback in the context of entity matching requires a thorough understanding of the user's needs and goals, as well as the ability to determine which characteristics are most relevant to the task at hand.

This paper aims to address the challenges of entity matching and relevance modeling by introducing an approach that replaces user feedback. Our work relates to modeling user feedback in an entity-matching scenario [28] [23] [11], where the user determines whether a tuple from the local database is a relevant match with a tuple from the external database (Figure 2). Given a local and external database, our approach eliminates the need for user feedback by using a learned model to perform the matching task.

To summarize, this paper makes the following main contributions:

1. This paper introduces an approach to model user relevance feedback in an entity matching scenario, utilizing Natural Language Processing (NLP) techniques in order to simulate such feedback.

2. We performed empirical experiments on three diverse datasets using logs from an online entity matching system to assess the effectiveness of our proposed method.

3. In our results, we demonstrate that our approach is robust towards complex negative samples.

2. Related Works

The general matching problem has *traditionally* relied on techniques such as Support Vector Machines (SVM) for identifying matches [1, 14]. However, recent advances in deep learning have yielded promising results in diverse domains, including natural language processing and computer vision.

The requirement of domain knowledge and feature engineering are the two biggest limitations of traditional matching methods [16]. Multiple deep learning-based approaches employing neural networks to automatically extract features from raw data and learn intricate matching patterns have been proposed to address these challenges.

This matching problem has also received significant attention across various domains, with commonalities observed in problems such as finding matches between **asymmetrical sequences, learning embedding spaces, relevance/matching scores, and architecture choices**. While our primary focus is on entity matching, we can still draw inspiration from related fields such as information retrieval, recommender systems and text matching.

Information Retrieval. *In IR, the primary goal of matching is to retrieve the top-k matches for a given query. This matching problem compares a short query against a long document, where the challenge lies in identifying the most relevant and useful information for the query from within the document.*

The researchers in [10] propose a deep relevance matching model for ad-hoc retrieval. The model uses a deep neural network to learn a non-linear transformation of the query-document pairs into a joint embedding space, where the relevance score between the query and document is computed. The architecture of the model, which includes multiple layers of feedforward neural networks and a similarity measure based on cosine similarity.

Recommender System. *In this domain, the entity matching problem involves comparing unstructured and asymmetric sequences, such as user data with item data, where the representations of user and item profiles may differ. Therefore, the architecture for this type of problem needs to take into account the possibility of varying representations between the two.*

Z. Deng et al. propose DeepCF [4] as a unified framework for learning representation and matching function in recommender systems. DeepCF adopts a deep neural network to learn the representations of users and items, and leverages a matching function to estimate the personalized preference score. The framework includes three modules: input module, representation learning module, and matching function learning module. The input module preprocesses the raw data and feeds it into the representation learning module to learn the latent representations of users and items. The matching function learning module combines the learned representations and learns to predict the preference score.

Text Matching. *In text matching, the problem of asymmetric sequence matching is similar to that in IR. However, due to the shorter length of both sequence in text matching, identifying a match becomes even more challenging. Unlike, in IR, the query is less likely to be a subset of the document, further adding to the complexity of the matching problem.*

W. Yu et al. propose WD-Match [30] as a novel approach for text matching in asymmetrical domains by incorporating Wasserstein distance regularization into sequence representation learning. Their approach involves two branches competing against each other: one estimates the Wasserstein distance using the projected features, while the other minimizes the Wasserstein distance regularized matching loss. The researchers demonstrate that regularizers helps WD-Match to generate feature vectors that are evenly distributed in the semantic space, making them more appropriate for matching.

D. Gogishvili et al. proposed SiameseCHEM [6], a Siamese Recurrent Neural Network (SRNN) based on the BiLSTM with a self-attention mechanism for bioactivity prediction. The SRNN model uses two recurrent neural networks with shared weights to learn representations of two chemical compounds, and a self-attention mechanism to focus on the most informative parts of the representations. The model architecture also incorporates an attention-based pooling mechanism to further improve the performance of the model.

Entity Matching *The entity matching problem is closely related to the general text matching problem, which has been extensively studied in the literature. Siamese architec-*

tures have emerged as a popular choice for text-matching tasks. In light of this, we draw motivation from a subset of recent state-of-the-art proposals for entity matching with these architectures.

The use of a Siamese hierarchical attention network for entity matching is proposed in [17]. This approach encodes the entity and description using a hierarchical attention mechanism to capture features at various levels of granularity. The Siamese architecture then generates a similarity score by comparing the encoded representations of the entity and description. However, the approach has limitations, including its reliance on high-quality entity descriptions and its inability to handle entities with multiple or ambiguous descriptions.

The article [18] proposes a Siamese-based BERT network called SiBERT for the alignment of Chinese medical entities. The SiBERT network utilizes a Siamese neural network architecture to learn the similarity between two entities, where a shared BERT network encodes the two input entities. The output of the Siamese network is a binary classification indicating whether the two input entities match or not.

L. Chen et al.in [2] present a multi-modal Siamese network architecture for entity alignment that integrates textual and visual information for entity matching across different knowledge graphs. The proposed architecture uses two subnetworks for the textual and visual modalities, which share weights and are trained jointly using a Siamese network approach to learn a similarity metric between entities. The authors also introduce a novel attention mechanism that selectively attends to different parts of the textual and visual features to improve the alignment accuracy.

This paper takes a different approach from those previously discussed above in that it focuses on the problem of simulating user feedback for systems that employ human-in-the-loop systems. Our objective in this paper is to introduce an approach for modeling user feedback in an entity matching scenario.

3. Problem Definition

Consider a system designed for entity matching between two databases, where one is a local database consisting of N tuples and the other is an external database consisting of M tuples, as depicted in Figure 2.

Given local tuple l_i and external tuple e_i, the user provides explicit relevance feedback to the system by identifying if l_i and e_i refer to the same entity (Equation 1). The system is dependent on user feedback for optimal performance.

$$relevance(l_i, e_i) = \begin{cases} 1, & \text{if } l_i \text{ and } e_i \text{ refer to the same entity} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

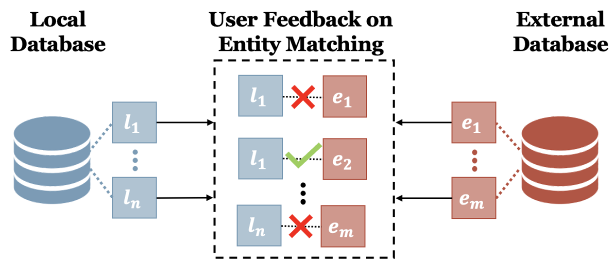

Figure 2. Incorporating user feedback for the entity matching problem.

Our objective is to mitigate this reliance on user data in online entity matching systems by learning a relevance model that can effectively simulate user feedback.

4. Model Architecture

We utilize a Siamese network to model user feedback in the context of entity matching. The architecture of our model is illustrated in Figure 3.

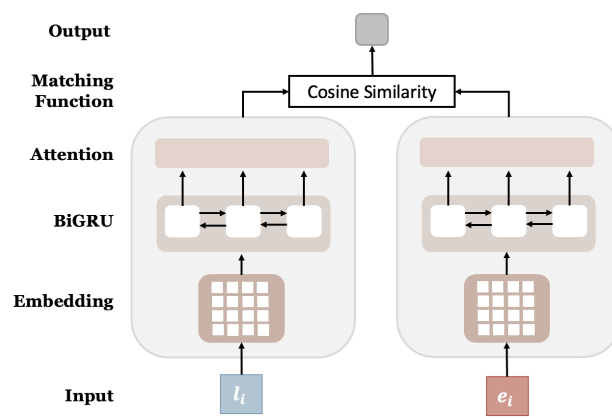

Figure 3. Model Architecture: Siamese network that encodes a pair of tuples (l_i, e_i) as input and measures the cosine similarity between them. The encoder, represented by the grey box, shares weights between both sides of the network.

4.1. Encoder

Given a pair of tuples (l_i, e_i), we encode local tuple l_i and external tuple e_i using an identical encoder with shared weights. The grey box in Figure 3 illustrates the encoder for our model.

4.1.1 Embeddings

Embeddings refer to vector representations of both external and local tuples. A straightforward approach to learning these embeddings is through static numbering of each word in these sequences, (*static word embeddings*). However, a major drawback of this approach is that all meanings of a word with multiple senses must share the same representation, which limits its efficacy [5]. To overcome this limitation, we leverage *contextual word representations* generated by state-of-the-art language model, GPT-2 [21].

$$embed_{l_i} = \textit{last-hidden-layer}(\textit{GPT-2}(l_i)) \quad (2a)$$
$$embed_{e_i} = \textit{last-hidden-layer}(\textit{GPT-2}(e_i)) \quad (2b)$$

To obtain the embeddings, we feed a tuple (e.g. l_i or e_i) as a sequence of words to GPT-2 and extract the last hidden layer as our embeddings. The final hidden layer of GPT-2 serves as a vectorized representation for each word in the input sequence.

4.1.2 BiGRU

Bidirectional Gated Recurrent Unit (BiGRU) [26] is a neural network architecture commonly used in NLP and sequence modeling tasks. BiGRU combines two Gated Recurrent Unit (GRU) layers, where one layer processes the input sequence in a forward direction, and the other processes the sequence in a backward direction. The architecture of BiGRU allows the model to capture contextual information from both the past and future context of each input token, thus enabling the model to better understand the meaning and structure of a sequence.

The BiGRU takes in a sequence embedding (Equation 2) as input and outputs a sequence representation b_t for tuple $t \in \{l_i, e_i\}$. Output for each side of the network is as follows:

$$b_{l_i} = BiGRU(embed_{l_i}) \quad (3a)$$
$$b_{e_i} = BiGRU(embed_{e_i}) \quad (3b)$$

4.1.3 Attention

An attention function maps a query and a set of key-value pairs to an output. The output is computed as a weighted sum of the values, where each weight assigned to a value is computed by a dot product of the query with the corresponding key [29]. We employ attention to enhance the representation of a sequence by attending each token to every other token (*self-attention*).

Given $b_t = (t_1, t_2, \ldots, t_n)$ as a sequence output from our BiGRU layer. The Attention layer computes the weights α_i for each token t_i based on its relationship with the other tokens in the same sequence. These attention weights not only reveal the relative importance of each token in the overall sequence semantics but also encode the similarities between tokens.

4.2. Matching Function

Similarity metrics are mathematical functions that measure the degree of similarity or dissimilarity between two objects. Similarity metrics are widely used in data analysis and machine learning to compare data points.

One popular similarity metric used in entity matching is cosine similarity [27]. It measures the cosine of the angle between two vectors in a multi-dimensional space. The resulting value ranges from -1 to 1, where 1 indicates that the two vectors are identical and -1 indicates that they are completely dissimilar.

Given attention vectors from each branch of our network, $\vec{a_l}$ and $\vec{a_e}$, their cosine similarity is:

$$Sim_C(\vec{a_l}, \vec{a_e}) = \frac{\vec{a_l} \cdot \vec{a_e}}{\| \vec{a_l} \| \| \vec{a_e} \|} \quad (4)$$

5. Loss Metrics

Loss metrics are used in training neural networks to measure the error between the predicted outputs and the true target values. The choice of loss metrics is task dependent. Our entity matching task has a binary classification form as our models try to predict between *match(1)* and *non-match(0)*.

5.1. Cross Entropy Loss

The cross-entropy method was introduced as an adaptive approach for calculating probabilities of infrequent events and addressing combinatorial optimization problems [25]. The cross-entropy loss function measures the dissimilarity between predicted and true probability distributions and has been shown to effectively improve model performance in binary classification tasks. [7]. The formula for cross-entropy loss can be expressed as follows:

$$H(p, q) = -\sum_i p(i) \log q(i) \quad (5)$$

where p is the true probability distribution and q is the predicted class distribution.

5.2. Contrastive Loss

Contrastive loss is a popular loss function for evaluating the performance of siamese networks and distinguishing between two similar representations. It works by pulling together neighbors and pushing apart non-neighbors within a set of high-dimensional training vectors [12].

For example, given a batch of N local and external tuple matches (l_i, e_i), contrastive loss leverages the $N \times N$ possible tuple pairings by maximizing the cosine similarity of the N true matches while minimizing the cosine similarity of the $N^2 - N$ non-matches. Given that our model outputs the cosine similarity $Sim_C(\vec{a_l}, \vec{a_e})$, we compute contrastive loss as described in [20]:

$$
\begin{aligned}
output &= Sim_C(\vec{a_l}, \vec{a_e}) && (6)\\
labels &= [0 : batchSize] && (7)\\
loss_l &= H(labels, output)) && (8)\\
loss_e &= H(labels, output^\top) && (9)\\
total\_loss &= \frac{loss_l + loss_e}{2} && (10)
\end{aligned}
$$

where H is cross entropy loss defined by Equation 5.

6. Experimental Setup

6.1. Datasets

To evaluate the performance of our model, we generate datasets using interaction logs from an online entity matching system that processed datasets from three prominent areas: Products, Drugs, and News.

Products. The products dataset contains product information from e-commerce websites Amazon and Google [3]. Shared identifiers (ISBNs, SKUs, etc.) that could be used to easily match entities together were removed.

Drugs. The drug dataset used in our experiments consists of drug reviews collected from the website "Drugs.com" and corresponding descriptions of the same drugs scraped from "Wikipedia" [8]. While the language used in the reviews is informal and diverse, which may not always be directly related to the drug, Wikipedia articles are heavily edited and provide a more formal and standardized description of the drug.

News. The news dataset comprises articles from 38 prominent media organizations [9]. It includes both the article titles and summaries (local) as well as the articles themselves (external). The summaries were generated by different authors, using various techniques, resulting in varying degrees of overlap.

6.2. User Simulation

We analyze interactions between the entity matching system and the user in the following fashion. During an interaction, the system presents one local tuple and 20 external tuples to the user. The user's task is to identify which external tuple(s), if any, match the local tuple. For instance, if the user identifies one external tuple as a match for a given local tuple, it implies that the remaining 19 external tuples do not match the local tuple. We consider matches as positive samples and non-matches as negative samples.

6.3. Models

Baseline. For our baseline, we construct a siamese style network similar to that in Figure 3. As described in Section 4, local and external tuples are processed by passing them through an embedding layer using GPT-2 *without fine-tuning*. The output of the embedding layer is passed to a fully connected layer for each branch of the network. The final layer takes the encoded representations from each branch and computes their cosine similarity. Due to the class imbalance in our datasets (i.e. more negative matches than positive matches), we use contrastive loss while training our model (5.2).

BiGRU + Attention. As shown in Figure 3, we first process the local and external tuples by embedding them with a pre-trained transformer-based model (e.g. GPT-2) *without fine-tuning*. These embeddings are then fed into a BiGRU. The output of the BiGRU is passed along to a 'matching-layer', which outputs a score of how relevant the external tuple is to the local tuple. To account for the imbalance in our data (i.e. more negative matches than positive matches), we use contrastive loss for training our model (5.2).

6.4. Experiment Details

We split the positive samples into *train* and *test* sets. The training set consists of 80% of the positive samples, while the test set consists of 20%.

To better simulate real-world user interactions and assess our model's capacity to distinguish the correct match among similar negative samples, we develop a *hard-negative* test set. This test set is created by including positive samples from the *test* set and their corresponding negative samples extracted from the same interaction (Section 6.1). In other words, each batch will contain 1 positive sample and $batchSize - 1$ corresponding negative samples. By including negative samples in a batch, we aim to better resemble a real-world user interaction, further testing our model's ability to select the correct match among similar negative samples.

Due to time constraints, our models were trained for different durations depending on the size of the dataset. Specifically, we trained the models for 1000 epochs for the Products dataset, 800 epochs for the Drugs dataset, and 50 epochs for the News dataset. During training, we employed the Adam optimizer with weight decay [13]. Additionally, we set a batch size of 16 for both training and testing across all datasets.

7. Evaluation

To evaluate the performance of our models we take the softmax over the model's output (i.e. cosine similarity) and interpret it as a probability distribution across the labels.

$$
\begin{aligned}
output &= Sim_C(\vec{a_l}, \vec{a_e})\\
labels &= [0 : batchSize]\\
probs &= softmax(output, axis = -1)\\
pred &= argmax(probs, axis = 1)
\end{aligned}
$$

Dataset	Source	Attributes	Positive Samples	Negative Samples
Products	Local External	name, description, manufacturer, price title, description, manufacturer, price	632	12,797
DrugCentral	Local External	'name', 'description', 'indication', 'synonyms', 'products',... page_title, wikipedia_summary	1791	34,439
News	Local External	title, article_summary article_content	25,570	633,638

Table 1. Details of datasets used in our evaluation with sampling distribution skewed towards negative (mismatch) samples

We calculate our model's prediction accuracy by comparing the predicted values to the true labels and report the results in Table 2. In the *hard-negative* test, we record the prediction accuracy for the positive sample only.

7.1. Results & Analysis

Train & Test Sets. Our findings indicate that the *BiGRU + Attention* model achieved better prediction accuracy than the *Baseline* on the Products and Drugs datasets, however, we also observed that it had higher loss metrics on the Products dataset (Table 2). In contrast to this trend, we found that the *Baseline* outperformed the *BiGRU + Attention* model on the News dataset.

We find it intriguing that the *BiGRU + Attention* model fared worse than the *Baseline*, despite the News dataset featuring more coherent natural language text. It is possible that the embeddings from GPT-2 provided a better representation for this dataset. However, we believe that the *BiGRU + Attention* model will surpass the *Baseline* given adequate training time.

Hard-Negative Test Set. With the exception of the *Baseline* model on the News dataset, we observed that hard-negative test accuracy was higher than both the train and test accuracy, even though the dataset contained the same positive samples as the test set (Table 2). Despite the *Baseline* outperforming the *BiGRU + Attention* model on the test and train sets, we observe that *BiGRU + Attention* performs better on the hard-negative test set, which better resembles a real-world entity matching interaction.

We find the high accuracy trend observed on the hard-negative test set to be particularly noteworthy since it is the test set that most closely simulates real-world entity-matching interactions. We are uncertain as to why the hard-negative test set outperforms the standard test set, given that it includes the same positive samples. However, we hypothesize that our original assumption of the true match being more similar to the negative samples may require additional investigation.

7.2. Conclusion

Our proposed architecture effectively simulates user feedback in an entity matching scenario. Moreover, we demonstrate that our model is robust against negative samples and can accurately identify the correct match in tests that closely resemble user interactions. Although our model did not outperform the baseline on the (largest) News dataset, we believe that with additional training time, it will surpass the baseline's performance.

8. Limitation and Future Work

Our work currently utilizes contextual word representations generated by GPT-2 [21]. However, the choice of embeddings can significantly affect the performance of any learning model. In future, we plan to explore other popular embeddings such as SBERT [24] and ELMo [19]. We also believe that leveraging task-specific embeddings rather than a *one-size-fits-all* approach could further improve our model's performance. For example, in the case of the Drugs dataset, we may benefit from using embeddings from BioBERT [15].

Another potential limitation of our approach pertains to the model settings and architecture choices. For instance, we did not incorporate fine-tuning for GPT-2 during training due to memory constraints. In future studies, we plan to investigate fine-tuning and compare our current recurrent-style architecture with a transformer-style architecture.

We also aim to improve our model's performance by exploring the use of *hard-negatives* also in the training process, which will include all examples (positive and negative) to better simulate real-world entity-matching scenarios. Moreover, a real-world scenario will include interactions where a batch of samples may only include negative samples. Therefore, we wish to explore the degree of certainty in our predictions by setting thresholds on the predicted probability distribution.

References

[1] Surajit Chaudhuri, Bee-Chung Chen, Venkatesh Ganti, and Raghav Kaushik. Example-driven design of efficient record matching queries. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, page 327–338. VLDB Endowment, 2007. 2

[2] Liyi Chen, Zhi Li, Tong Xu, Han Wu, Zhefeng Wang, Nicholas Jing Yuan, and Enhong Chen. Multi-modal siamese network for entity alignment. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and*

Dataset	Metric	Baseline Train	Baseline Test	Baseline Hard-Negative	BiGRU + Attention Train	BiGRU + Attention Test	BiGRU + Attention Hard-Negative
Products	Accuracy (%)	88.51	87.40	84.62	**89.90**	**90.55**	**1.00**
	Loss	0.36	0.33	-	0.38	0.45	-
Drugs	Accuracy (%)	82.40	79.39	86.01	**97.63**	**97.77**	**1.00**
	Loss	0.45	0.49	-	0.13	0.19	-
News	Accuracy (%)	**84.78**	**84.81**	77.94	76.81	76.53	**1.00**
	Loss	0.43	0.45	-	0.54	0.55	-

Table 2. Experiment results. Train and Test set consist of only positive samples, while Hard-Negative Test set uses positive samples and their corresponding negative samples. Each Hard-Negative Test batch consists of 1 positive sample and (batch size - 1) negative samples. Prediction accuracy (Acc.) is determined by comparing the label with the highest softmax probability against the true prediction label.

Data Mining, KDD '22, page 118–126, New York, NY, USA, 2022. Association for Computing Machinery. 3

[3] Sanjib Das, AnHai Doan, Paul Suganthan G. C., Chaitanya Gokhale, Pradap Konda, Yash Govind, and Derek Paulsen. The magellan data repository. https://sites.google.com/site/anhaidgroup/projects/data. 5

[4] Zhi-Hong Deng, Ling Huang, Chang-Dong Wang, Jian-Huang Lai, and Philip S. Yu. Deepcf: A unified framework of representation learning and matching function learning in recommender system. *CoRR*, abs/1901.04704, 2019. 2

[5] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv*, 2019. 4

[6] Gogishvili D Nittinger E Zhao H Tyrchan C Fernández-Llaneza D, Ulander S. Siamese recurrent neural network with a self-attention mechanism for bioactivity prediction. *ACS Omega*, 3(6):11086–11094, 2021. 2

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016. 4

[8] Felix Gräßer, Surya Kallumadi, Hagen Malberg, and Sebastian Zaunseder. Aspect-based sentiment analysis of drug reviews applying cross-domain and cross-data learning. In *International Conference on Digital Health*, pages 121–125, 2018. 5

[9] Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *NAACL*, 2018. 5

[10] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 55–64, New York, NY, USA, 2016. Association for Computing Machinery. 1, 2

[11] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. page 55–64, 2016. 2

[12] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006. 4

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[14] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Learning-based approaches for matching web data entities. *IEEE Internet Computing*, 14(4):23–31, 2010. 2

[15] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *arXiv*, 2020. 6

[16] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, Jin Wang, Wataru Hirota, and Wang-Chiew Tan. Deep entity matching: Challenges and opportunities. *J. Data and Information Quality*, 13(1), jan 2021. 1, 2

[17] Kai Ma, Liang Wu, Liufeng Tao, Wenjia Li, and Zhong Xie. Matching descriptions to spatial entities using a siamese hierarchical attention network. *IEEE Access*, 6:28064–28072, 2018. 3

[18] Zerui Ma, Linna Zhao, Jianqiang Li, Xi Xu, and Jing Li. Sibert: A siamese-based bert network for chinese medical entities alignment. *Methods*, 205:133–139, 2022. 3

[19] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv*, 2018. 6

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 4

[21] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 4, 6

[22] Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi, and Jimmy Lin. Bridging the gap between relevance matching and semantic matching for short text similarity modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5370–5381, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. 1

[23] Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi, and Jimmy Lin. Bridging the gap between relevance matching and semantic matching for short text similarity modeling. pages 5370–5381, Nov. 2019. 2

[24] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. 6

[25] R. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997. 4

[26] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. 4

[27] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001. 4

[28] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. Deep learning for blocking in entity matching: A design space exploration. *Proc. VLDB Endow.*, 14(11):2459–2472, oct 2021. 2

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 4

[30] Weijie Yu, Chen Xu, Jun Xu, Liang Pang, Xiaopeng Gao, Xiaozhao Wang, and Ji-Rong Wen. Wasserstein distance regularized sequence representation for text matching in asymmetrical domains. *arXiv preprint arXiv:2010.07717*, 2020. 2