Tutorial Proposal: Synergy of Database Techniques and Machine Learning Models for String Similarity Search and Join

Jiaheng Lu
University of Helsinki
jiaheng.lu@helsinki.fi

Chunbin Lin
Amazon AWS
lichunbi@amazon.com

Jin Wang
University of California, Los Angeles
jinwang@cs.ucla.edu

Chen Li
University of California Irvine
chenli@ics.uci.edu

ABSTRACT

String data is ubiquitous and string similarity search and join are critical to the applications of information retrieval, data integration, data cleaning, and also big data analytics. To support these operations, many techniques in the database and machine learning areas have been proposed independently. More precisely, in the database research area, there are techniques based on the filtering-and-verification framework that can not only achieve a high performance, but also provide guaranteed quality of results for given similarity functions. In the machine learning research area, string similarity processing is modeled as a problem of identifying similar text records; Specifically, the deep learning approaches use embedding techniques that map text to a low-dimensional continuous vector space.

In this tutorial, we review a large number of studies of string similarity search and join in these two research areas. We divide the studies in each area into different categories. For each category, we provide a comprehensive review of the relevant works, and present the details of these solutions. We conclude this tutorial by pinpointing promising directions for future work to combine techniques in these two areas.

KEYWORDS

databases, machine learning, string similarity join, string similarity search, data integration

ACM Reference Format:
Jiaheng Lu, Chunbin Lin, Jin Wang, and Chen Li. 2019. Tutorial Proposal: Synergy of Database Techniques and Machine Learning Models for String Similarity Search and Join . In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 03–07, 2019, Beijing, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3357384.3360319

1 INTRODUCTION

Today's society is immersed in a wealth of text data, ranging from news articles, to social media, research literature, medical records, and corporate reports. Identifying data referring to same real-world entities is a core task of information retrieval, data integration, data cleansing and data mining when integrating data from multiple sources. This problem can be resolved with string similarity search and join [2, 4, 9, 17, 22, 29, 31, 50, 51, 57, 58]. Data referring to the same real-world entity is usually represented in different formats due to the following possibilities: (i) data stored in different sources may contain typos and be inconsistent. E.g., "NBA Mcgrady" and "Macgrady NBA" refer to the same NBA player though there are minor errors in the texts; and (ii) data stored in different sources use different representations .E.g., "University of Washington" and "UW" refer to the same university, while "Amzon Lab126" and "1100 Enterprise Way, Sunnyvale, CA 94089" refer to the same building. It is very challenging to identify matched strings correctly. In addition, when scaling this problem to big data volume, it brings extra performance challenge.

In this tutorial, we introduce the state-of-the art techniques in database and machine learning areas to solve the challenges by providing high-quality matching approaches and efficient algorithms for string similarity search and join [7, 19, 40, 62]. We first present the existing works in database area and machine learning area separately, then we discuss the connection between the techniques in these two different areas. In database area, a plethora of works are proposed to design various index structures to improve the performance of similarity string search and join. In machine learning area, similarity string matching is modeled as the problem of entity matching, which aims at identifying whether two entities are with the same identification. And different kinds of models are trained to find such entity pairs. Recently, deep learning techniques have been extensively adopted in identifying string semantic similarity, where texts are mapped into low-dimensional continuous vector space. They use embedding techniques to find matched entities. The building blocks of deep learning for string similarity measurement are mainly Recurrent Neural Network (RNN) [20] and distributed representation learning [39]. Figure 1 provides an overview of the history of the string similarity measures and entity matching techniques.

To the best of our knowledge, this is the first tutorial to discuss the synergy between database techniques and machine learning approaches on string similarity search and join. Note that the problem

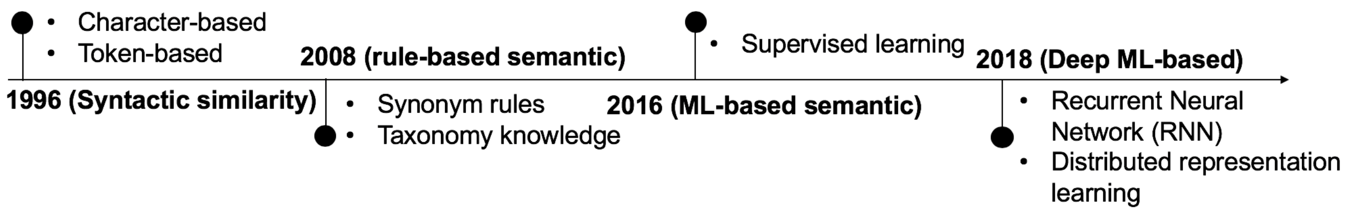

Figure 1: Recent 20 years of string similarity measures

of string similarity matching was well studied in computer science. The first references to this problem could be traced to the sixties and seventies [41], where the problem appeared in a number of different fields, such as computational biology, signal processing, and text retrieval. However, this tutorial will review this problem from a novel angle through the synergy between database and machine learning techniques. The existing tutorial and survey review this problem separately from their respective fields. We have identified few tutorials (e.g. [24]) on string similarity search, which are mainly from VLDB and ICDE. However, they were given in 2009, ten years ago. This tutorial, on the other hand, focuses on the state-of-the-art works on string similarity search and join bridging the two fields of database and machine learning.

Relevance to CIKM The topic of the tutorial is closely related to CIKM. The 2019 theme "AI for the future life" emphasizes the target of AI with all their features. This tutorial will combine the technique of databases and machine learning for string similarity processing. Efficient search and join of string data is a fundamental problem for various fields covered by CIKM conference, e.g., information retrievals, data management and data mining. In this tutorial, we will show the approaches and features that the current string processing frameworks offer, as well as their limitations and open challenging research areas.

2 COVERED TOPICS

2.1 Background and preliminaries

String is ubiquitous in computer science. In the first part of the tutorial, we will provide several motivating examples of string similarity search and join, followed by a brief introduction on their wide applications in various fields, such as information retrieval, databases and data mining, to name a few.

2.2 Database techniques

To identify similarity strings, most existing studies utilize either character-based similarity functions, e.g., *edit distance* and *hamming distance*, or token-based similarity functions, e.g. *Overlap*, *Jaccard* and *Cosine*, to quantify similarity of two strings. There are also similarity functions to combine character-based and token-based similarity functions together (e.g. [49, 53]).

2.2.1 String Similarity Search. String similarity search finds all strings that are similar to a given query from a defined string dataset. There are two variants of similarity search problem: *threshold-based* and *top-k* search. Similarity search needs to construct the index in an offline step without knowing the similarity thresholds ahead of time. Existing studies proposed a series of indexing techniques to improve the search performance, such as inverted list based indexes [5, 26, 32] and tree based ones [35, 48, 61, 65–67].

2.2.2 String Similarity Join. String similarity join is a fundamental operation in many applications, such as data cleaning and integration. Given two collections of strings, string similarity join aims at finding all similar string pairs from the two collections. Most existing studies adopt the filter-and-verification framework to improve the performance of string similarity join. In the filtering step, signatures are generated for each string and used to identify candidate pairs. In the verification step, the real similarity is calculated on the candidate pairs to generate the final results. To improve the pruning powers, a variety of filtering techniques are proposed, such as count filter [22], prefix filter [4, 9, 50], position filter [58], mismatch filter [57] and segment filter [17, 31].

Recently there is an increasing demand for more efficient approaches which can scale up to increasingly vast volume of data as well as make good use of rich computing resources. Similar to previous centralized algorithms, such distributed algorithms focused on finding high quality signatures to improve the overall performance. Meanwhile, distributed algorithms also need to address the problem of data skew so as to ensure load balance between all workers in a cluster. To support string similarity joins over big data, many recent contributions focus on implementing algorithms on Map-Reduce [21]. Vernica et al. [47] adopted the prefix filter in the MapReduce framework to enhance the filter power of parallel similarity join. A large body of studies followed its way to make optimization in computing similarity score [37], estimating cost [1], improving the signatures [16] and addressing data skewness [44]. Recently a systems-oriented approach [27] is proposed to efficiently support the similarity search and join operations upon the Apache AsterixDB system.

2.2.3 Approximate Entity Extraction. Dictionary based Approximate Entity Extraction(AEE) is a typical application scenario of the string similarity join techniques.Given a predefined dictionary of entities, a similarity threshold and a collection of documents, it aims at identifying all substrings from a document that are similar with entities in the dictionary by employing syntactic similarity functions (e.g. Jaccard and Edit Distance).To accelerate this process, previous studies utilized different approaches, such as hashing [8], neighborhood generation [56] and segmentation indexing [14].Faerie [15] is an all-purposed framework to support multiple kinds of similarity metrics in AEE problem.Wang et al. [55] addressed the local similarity search problem, which is a variant of AEE problem but with more limitations.

2.2.4 Synonym-based Semantic Similarity. Though above studies have achieved significant degree of success in identifying the syntactic similar strings, they would still miss some results that are semantically similar with each other. To this end, Arasu et al. [3] integrated synonym rules with syntactic similar metrics to improve the quality of string matching. Lu et al. [33, 34] followed this route and proposed efficient string join algorithms with synonyms. Recently Wang et al. [52] introduced synonym rules into the approximate entity extraction problem to combine the synonym rules with syntactic similarity metrics. Xu et al. [59, 60] study how to use taxonomy knowledge to enable semantic similarity joins.

2.3 Machine leaning techniques

2.3.1 Traditional Machine Learning based Approaches. Machine Learning techniques have been prove to be effective in the task of Entity Matching (a.k.a Entity Resolution), which is an important application that relies on string similarity measurement. It aims at identifying data instances that refer to the same real world entity. Traditional approaches modeled entity matching as a supervised learning problem. MARLIN [6] utilized Support Vector Machine to capture the syntactic features and attributes, while [18] relied on Bayesian network. The Magellan system [29] provided an end-to-end solution based on a variety of similarity measurements. It supported entity matching with both machine learning and rule bases methods. The Falcon system [13] further integrated human labor into this process to improve the effectiveness and provide a system deployment on top of Apache Hadoop.

2.3.2 Deep Learning Approaches. Nowadays deep learning techniques have been extensively adopted in identifying string semantic similarity. The building blocks of deep learning for string similarity measurement are mainly Recurrent Neural Network (RNN) [20] and distributed representation learning [39]. RNN is a family of neural networks with a dynamic temporal behavior. A neuron in RNN is fed information not only from the previous time step, but also from its own previous state in time, to process sequences of inputs. Distributed representation learning maps texts into a low-dimensional continuous vector space to learn the *embedding* of them. There are a series of studies in this category, ranging from the embedding of word [38, 39, 43], phrase [10, 25, 45], sentence [12, 28, 30, 42] to document level [36, 46, 54, 64]. Compared with traditional rule-based methods, these embedding approaches can make better use of the large training data and achieve superior performance. Traditional Natural Language Processing tasks that related to semantic similarity can also benefit from the techniques of distributed word representation and deep neural networks. Typical examples include natural language inference [11], semantic matching [23] and relation extraction [63].

Recently there are several studies that utilized deep learning techniques for string similarity matching. DeepER [19] aimed at learning the latent representation of entities to enable more accurate matching. It can be applied in cases whether the pre-trained word embedding is available or not. Mudgal et al. [40] extended the work of DeepER and introduced a unified template of deep learning methods for entity matching by considering 3 crucial steps: attribute embedding, attribute similarity and classifier.

2.4 The Integration of Database and Machine Learning Techniques

In above sections, we introduce the recent studies in both database and machine learning fields. On the one hand, the studies in the database field can provide high efficiency and good interpretability. However, they mainly rely on syntactic features and simple semantic resources like synonym rules, which result in limited power of effectiveness. On the other hand, the machine learning based works can automatically learn from massive corpus and capture extensive semantics. But they require tedious training time and human annotated dataset, which is rather expensive. An emerging trend is to combine database techniques with those in the field of machine learning to take advantage of both of them. Recently Smurf [7] is the up-to-date work which combines machine learning and database techniques to enable automatic matching between two string sets. It can first automatically acquire the features by considering predefined rules as well as syntactic similarity. Then it adopts the random forest model to learn the features for matching similar strings. It further adopts multiple techniques from database community, such as indexing, query planning and pruning, to improve the efficiency of the learnign process.

2.5 Future challenge

Optimize the pipeline of String Similarity Queries As shown above, there have been many studies about string similarity search and join in the database community. However, they just focus on the algorithm level optimization. To make the similarity search and join techniques more usable in practical applications, it further calls for an end-to-end pipeline to deal with the whole life cycle of the task. It is essential to build such a pipeline on relational database management systems like MySQL or NoSQL database like MongoDB.

More Efficient ML based Approaches The up-to-date approaches adopt deep learning techniques, i.e. word embedding and variants of RNN models, on the task of entity matching and achieve state-of-the-art performance. Though the promising results in the aspect of effectiveness, there are still remaining problems on efficiency. Compared with the rule-based approaches as well as the string similarity search and join approaches, the machine learning based methods need a relatively long time for training the model. Moreover, since the dominant approaches in this category are supervised ones, it requires a large amount of labeled data to serve as the training set, which is rather expensive due to human labor for annotation. How to automatically acquire training data or develop efficient unsupervised approaches for the problem of entity matching remain to be an open problem.

Combine Human-in-the-Loop Approaches with ML As is well known, some important data analytic tasks cannot be completely addressed by automated processes. It is essential to involve human cognitive ability into this process to enhance the overall performance. Recently the crowdsourcing approaches have been widely adopted in different tasks related to string similarity measurement. They serve as a crucial signal in recognizing the truth. The future direction is to automatically identify when and to what extent should human labor be involved in the process with effective machine learning tools. The Falcon system [13] is the first work towards this goal and there is further room to improve.

3 TUTORIAL ORGANIZATION

This tutorial consists of 4 parts and is planned for 3 hours. The details are explained as following.

Motivation and Background (30 minutes).

- A brief overview of the history of string similarity search and join.
- The real world applications of string similarity search and join.
- The formal problem definition and necessary background.

Database Related Techniques (60 Minutes).

- State-of-the-art algorithms about String similarity join.
- String similarity search algorithms.
- Enhancing string similarity and join with semantic features including synonym and taxonomy knowledge
- Application of string search and join techniques, including approximate entity extraction, query auto-complete and conjunction with other kinds of data, e.g. graph, spatial and streaming.

Machine Learning Related Techniques (60 Minutes).

- Traditional machine learning approaches for string similarity measurement.
- Preliminary about deep learning and its application in natural language processing.
- String similarity measurement with deep learning techniques.

Synergy between Database and Machine Learning (15 Minutes).

- Applying database filtering technique to enhance machine leaning algorithms.
- Incorporating machine leaning models in string similarity functions for string joins in databases.

Open Problems (15 Minutes).

- General purposed pipeline for string similarity search and join.
- Accelerating the machine learning based approaches.
- Combining Human-in-the-loop with machine learning approaches for better performance.

4 GOALS OF THE TUTORIAL

String similarity search and join are very generic problems in both database and machine learning areas. Though each individual area has many promising techniques proposed to solve the problems, they are considered as orthogonal approaches. This tutorial will think this problem in a higher level of view by combining techniques in both database and machine learning areas. In the following, we present the main learning outcomes of this tutorial in two aspects: (i) the take-away items from the existing work; and (ii) the challenge and directions to motivate future researches.

4.1 Take-away items.

- The history and motivation for string similarity searches and joins.
- The categorization and methods of string similarity searches and joins in database area, including various filtering strategies, distributed algorithms, synonym and taxonomy based algorithms;
- The methodology and models of string similarity searches and joins in machine learning area, including traditional machine learning based approaches and deep learning approaches.
- The synergy of database and machine learning techniques to enable automatic string similarity processing.

4.2 Future directions of the synergy.

- **From database to machine learning**. Existing work in database area focus more on improving the performance by applying fast filtering techniques, which can be modified and utilized in machine learning algorithms. One example is to use such database techniques to fast filter dirty training data.
- **From machine learning to database**. Existing work in machine learning improves the matching quality by considering the semantics, which can be applied in database algorithms to enhance the similarity measure functions.

4.3 Intended Audience

This tutorial is intended for a wide scope of audience ranging from academic researchers to industrial data scientists that want to understand the string similarity processing algorithms in the era of big data and AI. Basic knowledge in string processing in databases is sufficient to follow the tutorial. Some background in machine learning and deep learning techniques would be useful but not necessary.

5 SHORT BIBLIOGRAPHIES OF TUTORS

Jiaheng Lu is an Associate Professor at the University of Helsinki, Finland. His main research interests lie in the big data management and database systems, and specifically in the challenge of efficient data processing from real-life, massive data repository and Web. He has written four books on Hadoop and NoSQL databases, and more than 80 journal and conference papers published in SIGMOD, VLDB, TODS, and TKDE, etc. He will attend CIKM and present part of the tutorial.

Chunbin Lin is a database engineer at Amazon Web Services (AWS) and he is working on AWS Redshift. He completed his Ph.D. in computer science at the University of California, San Diego (UCSD). His research interests are database management and big data management. He has more than 20 journal and conference papers published in SIGMOD, VLDB, VLDB J, and TODS, etc. He will attend CIKM and present part of the tutorial.

Jin Wang is a fourth year PhD student at the University of California, Los Angeles. Before joining UCLA, he obtained his master degree in computer science from Tsinghua University in the year 2015. His research interest mainly lies in the field of data management and text analytics. He has published more than 10 papers in top-tier conferences and journals like ICDE, IJCAI, EDBT, TKDE and VLDB Journal. He will attend CIKM and present part of the tutorial.

Chen Li is a professor in the Department of Computer Science at UC Irvine. He received his Ph.D. degree in Computer Science from Stanford University. His research interests are in the field of data management, including data-intensive computing, query processing and optimization, visualization, and text analytics. He will help prepare the tutorial material. Due to his teaching workload in UCI, he probably will not attend CIKM conference.

REFERENCES

[1] F. N. Afrati, A. D. Sarma, D. Menestrina, A. G. Parameswaran, and J. D. Ullman. Fuzzy joins using mapreduce. In *ICDE*, pages 498–509, 2012.

[2] P. Agrawal, A. Arasu, and R. Kaushik. On indexing error-tolerant set containment. In *SIGMOD*, pages 927–938, 2010.

[3] A. Arasu, S. Chaudhuri, and R. Kaushik. Transformation-based framework for record matching. In *ICDE*, pages 40–49, 2008.

[4] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *WWW*, pages 131–140, 2007.

[5] A. Behm, C. Li, and M. J. Carey. Answering approximate string queries on large data sets using external memory. In *ICDE*, pages 888–899, 2011.

[6] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *ACM SIGKDD*, pages 39–48, 2003.

[7] P. S. G. C., A. Ardalan, A. Doan, and A. Akella. Smurf: Self-service string matching using random forests. *PVLDB*, 12(3):278–291, 2018.

[8] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin. An efficient filter for approximate membership checking. In *SIGMOD*, pages 805–818, 2008.

[9] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE*, page 5, 2006.

[10] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[11] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680, 2017.

[12] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni. What you can cram into a single \$&!#* vector: Probing sentence embeddings for linguistic properties. In *ACL*, pages 2126–2136, 2018.

[13] S. Das, P. S. G. C., A. Doan, J. F. Naughton, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, and Y. Park. Falcon: Scaling up hands-off crowdsourced entity matching to build cloud services. In *SIGMOD*, pages 1431–1446, 2017.

[14] D. Deng, G. Li, and J. Feng. An efficient trie-based method for approximate entity extraction with edit-distance constraints. In *ICDE*, pages 762–773, 2012.

[15] D. Deng, G. Li, J. Feng, Y. Duan, and Z. Gong. A unified framework for approximate dictionary-based entity extraction. *VLDB J.*, 24(1):143–167, 2015.

[16] D. Deng, G. Li, S. Hao, J. Wang, and J. Feng. Massjoin: A mapreduce-based method for scalable string similarity joins. In *ICDE*, pages 340–351, 2014.

[17] D. Deng, G. Li, H. Wen, and J. Feng. An efficient partition based method for exact set similarity joins. *PVLDB*, 9(4):360–371, 2015.

[18] X. Dong, A. Y. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, pages 85–96, 2005.

[19] M. Ebraheem, S. Thirumuruganathan, S. R. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. *PVLDB*, 11(11):1454–1467, 2018.

[20] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[21] F. Fier, N. Augsten, P. Bouros, U. Leser, and J.-C. Freytag. Set similarity joins on mapreduce: An experimental survey. *PVLDB*, 11(10):1110–1122, 2018.

[22] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB*, pages 491–500, 2001.

[23] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. Semantic matching by non-linear word transportation for information retrieval. In *CIKM*, pages 701–710, 2016.

[24] M. Hadjieleftheriou and C. Li. Efficient approximate search on string collections. *PVLDB*, 2(2):1660–1661, 2009.

[25] F. Hill, K. Cho, A. Korhonen, and Y. Bengio. Learning to understand phrases by embedding the dictionary. *TACL*, 4:17–30, 2016.

[26] J. Kim, C. Li, and X. Xie. Hobbes3: Dynamic generation of variable-length signatures for efficient approximate subsequence mappings. In *ICDE*, pages 169–180, 2016.

[27] T. Kim, W. Li, A. Behm, I. Cetindil, R. Vernica, V. R. Borkar, M. J. Carey, and C. Li. Supporting similarity queries in apache asterixdb. In *EDBT*, pages 528–539, 2018.

[28] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[29] P. Konda, S. Das, P. S. G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra. Magellan: Toward building entity matching management systems. *PVLDB*, 9(12):1197–1208, 2016.

[30] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.

[31] G. Li, D. Deng, J. Wang, and J. Feng. PASS-JOIN: A partition-based method for similarity joins. *PVLDB*, 5(3):253–264, 2011.

[32] W. Li, L. Deng, Y. Li, and C. Li. Zigzag: Supporting similarity queries on vector space models. In *SIGMOD*, pages 873–888, 2018.

[33] J. Lu, C. Lin, W. Wang, C. Li, and H. Wang. String similarity measures and joins with synonyms. In *SIGMOD*, pages 373–384, 2013.

[34] J. Lu, C. Lin, W. Wang, C. Li, and X. Xiao. Boosting the quality of approximate string matching by synonyms. *ACM Trans. Database Syst.*, 40(3):15:1–15:42, 2015.

[35] J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. In *SIGMOD*, pages 349–360, 2011.

[36] L. Luo, X. Ao, F. Pan, J. Wang, T. Zhao, N. Yu, and Q. He. Beyond polarity: Interpretable financial sentiment analysis with hierarchical query-driven attention. In *IJCAI*, pages 4244–4250, 2018.

[37] A. Metwally and C. Faloutsos. V-smart-join: A scalable mapreduce framework for all-pair similarity joins of multisets and vectors. *PVLDB*, 5(8):704–715, 2012.

[38] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[39] T. Mikolov and G. Zweig. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*, pages 234–239, 2012.

[40] 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 *SIGMOD*, pages 19–34, 2018.

[41] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, Mar. 2001.

[42] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. K. Ward. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 24(4):694–707, 2016.

[43] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[44] C. Rong, C. Lin, Y. N. Silva, J. Wang, W. Lu, and X. Du. Fast and scalable distributed set similarity joins for big data analytics. In *ICDE*, pages 1059–1070, 2017.

[45] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, pages 801–809, 2011.

[46] B. Tian, Y. Zhang, J. Wang, and C. Xing. Hierarchical inter-attention network for document classification with multi-task learning. In *IJCAI*, pages 3569–3575, 2019.

[47] R. Vernica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using mapreduce. In *SIGMOD*, pages 495–506, 2010.

[48] J. Wang, G. Li, D. Deng, Y. Zhang, and J. Feng. Two birds with one stone: An efficient hierarchical framework for top-k and threshold-based string similarity search. In *ICDE*, pages 519–530, 2015.

[49] J. Wang, G. Li, and J. Feng. Fast-join: An efficient method for fuzzy token matching based string similarity join. In *ICDE*, pages 458–469, 2011.

[50] J. Wang, G. Li, and J. Feng. Can we beat the prefix filtering?: an adaptive framework for similarity join and search. In *SIGMOD*, pages 85–96, 2012.

[51] J. Wang, G. Li, J. X. Yu, and J. Feng. Entity matching: How similar is similar. *PVLDB*, 4(10):622–633, 2011.

[52] J. Wang, C. Lin, M. Li, and C. Zaniolo. An efficient sliding window approach for approximate entity extraction with synonyms. In *EDBT*, pages 109–120, 2019.

[53] J. Wang, C. Lin, and C. Zaniolo. Mf-join: Efficient fuzzy string similarity join with multi-level filtering. In *ICDE*, pages 386–397, 2019.

[54] J. Wang, Z. Wang, D. Zhang, and J. Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, pages 2915–2921, 2017.

[55] P. Wang, C. Xiao, J. Qin, W. Wang, X. Zhang, and Y. Ishikawa. Local similarity search for unstructured text. In *SIGMOD*, pages 1991–2005, 2016.

[56] W. Wang, C. Xiao, X. Lin, and C. Zhang. Efficient approximate entity extraction with edit distance constraints. In *SIGMOD*, pages 759–770, 2009.

[57] C. Xiao, W. Wang, and X. Lin. Ed-join: an efficient algorithm for similarity joins with edit distance constraints. *PVLDB*, 1(1):933–944, 2008.

[58] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, pages 131–140, 2008.

[59] P. Xu and J. Lu. Efficient taxonomic similarity joins with adaptive overlap constraint. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 1563–1566, 2018.

[60] P. Xu and J. Lu. Towards a unified framework for string similarity joins. *PVLDB*, 12(11):1289–1302, 2019.

[61] J. Yang, Y. Zhang, X. Zhou, J. Wang, H. Hu, and C. Xing. A hierarchical framework for top-k location-aware error-tolerant keyword search. In *ICDE*, pages 986–997, 2019.

[62] M. Yu, G. Li, D. Deng, and J. Feng. String similarity search and join: a survey. *Frontiers of Computer Science*, 10(3):399–417, 2016.

[63] D. Zeng, K. Liu, Y. Chen, and J. Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, pages 1753–1762, 2015.

[64] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657, 2015.

[65] Y. Zhang, X. Li, J. Wang, Y. Zhang, C. Xing, and X. Yuan. An efficient framework for exact set similarity search using tree structure indexes. In *ICDE*, pages 759–770, 2017.

[66] Y. Zhang, J. Wu, J. Wang, and C. Xing. A transformation-based framework for knn set similarity search. *IEEE Trans. Knowl. Data Eng.*, 2019.

[67] Z. Zhang, M. Hadjieleftheriou, B. C. Ooi, and D. Srivastava. Bed-tree: an all-purpose index structure for string similarity search based on edit distance. In *SIGMOD*, pages 915–926, 2010.