Learning String Alignments for Entity Aliases

Aaron Traylor* , **Nicholas Monath***, **Rajarshi Das, Andrew McCallum**
College of Information and Computer Science
University of Massachusetts Amherst
{atraylor, nmonath, rajarshi, mccallum}@cs.umass.edu

Abstract

Entity resolution systems often rely on string similarity between entity mention spellings. These string similarity models are potentially more effective when they are learned for a particular domain. For example, "John A. Smith" is more similar to "John Austin Smith" than "John B. Smith" for Western names and "ABC" is more similar to "ABC Co." than "CBC" for business names. However, an unweighted edit distance model would make wrong predictions in both of these cases. In this paper, we train neural network models of string similarity to predict how likely two mentions are to refer to the same entity based solely on the spelling of the mentions. Our approach uses recurrent neural network models to learn embedded representations of the characters of a string, and a learned model to score the alignment of the embedded representations of a pair of strings. We describe an approach using convolutional neural networks to score the alignment of the embedded representation. We compare our approach and several baseline approaches on large datasets and find that the convolutional alignment model significantly outperforms the next best baseline.

1 Introduction

String similarity measures are a crucial component in many entity resolution, deduplication, and coreference systems. Entity resolution approaches are pervasively used in automated knowledge base construction systems to ground ambiguous *mentions* to the *entities* to which they refer. String similarity measures are used to determine how likely two (or more) mention spellings are to refer to the same entity independent of other contextual features. Simple string similarity measures such as Levenshtein, longest common subsequence, and Jaro-Winkler are used in many state of the art approaches [20, 23, 24, 33]. Levin et al [23] use Jaro-Winkler similarities when performing entity resolution on a knowledge base of scientific authors, as do Li et al [24] in record linkage for business names. While these simple similarity measures which compute the number of transformations needed to align one sequence of characters with the other can be effective, they have several shortcomings. These measures have only a few parameters describing the alignment of strings, which are not robust enough to capture unique characteristics of string similarity in a given domain (e.g. company names, music album titles) [12, 25]. More specifically, these measures often lack a parameterization of the surrounding *context* of the string where an edit is applied. This is required, for example, to capture that it is more likely to see a company designation (e.g "Inc.", "L.L.C.") at the end of a string than in the middle or beginning. These models typically also have a relatively simple model of edits, which gives a single weight to each edit type for only a few simple operations and is independent of the characters involved in the edit. This does not allow the model to capture that changing a person's middle name from a full name to an initial is more likely than changing one middle initial to a different initial (i.e. "John A. Smith" is more similar to "John Austin Smith" than "John B. Smith").

*The first two authors contributed equally.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

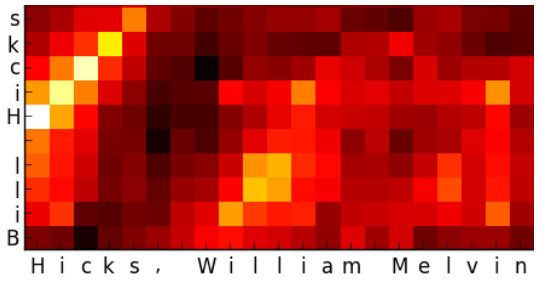

Figure 1: A 'soft alignment' matrix for two strings obtained by multiplying the LSTM hidden state representations at each character position. The soft alignment matrix captures regularities at various positions. For different input strings, these regularities would be spread over in the matrix. These patterns can be easily captured by a convolutional neural network.

Learned string edit models capture both the context of an edit as well as the contents involved in the edit. These models are a specific case of the more generally studied problem of sequence alignment. [8, 26, 30, 35]. Learned sequence alignment models have been shown to be very powerful for string edits, computational morphology, written-to-spoken form conversion, and other related problems [14, 15, 16, 25, 27].

In this paper, we focus on the problem of learning string similarity measures for entity aliases. More precisely, we train neural network models of string similarity to predict how likely two mentions are to refer to the same entity based solely on the spelling of the mentions. Mentions can appear in noisy or ambiguous forms– nicknames can be used and names can appear in reverse order (Figure 1). Our approach uses recurrent neural network models to learn embedded representations of the characters of a string and a learned model to score the alignment of the embedded representations of a pair of strings. The recurrent models of each string provide a string-context dependent representation of characters. These context dependent representations can be used to form a soft alignment matrix (Figure 1). Mentions belonging to the same entities often exhibit diagonal patterns which are spread out in the soft alignment matrix. Such patterns can be easily captured by filters of convolutional neural networks. Our model is trained end-to-end and we evaluate our model on five automatically constructed datasets from publicly available knowledge bases. The knowledge bases span entity alias information on a variety of domains including patent assignees, music artists, disease names, and entities in Wikipedia. We compare our approach and several baseline approaches on these datasets and find that the convolutional alignment model significantly outperforms the next best baseline.

2 Learning String Similarity Measures

2.1 Problem

Our goal is to learn a string similarity measure for entity mentions. Our model only takes into account the *spelling* of the entity mention and does not consider any additional contextual information which may be available. We focus on this problem with a narrower scope with the belief that our approach could be used in conjunction with any existing or new entity resolution system that makes use of a string similarity measure. As string similarity measures are more typically used in entity resolution, where the task is to discover or cluster mentions into entities rather than link to an existing entity in a knowledge base, we focus on measuring similarities between mention spellings and do not model entities themselves.

We refer to a pair of mention spellings as **aliases** if both could be used to refer to the same entity. For instance, the mention strings `Barack H. Obama` and `Barry Obama` are aliases as they both refer to the entity `wiki/Barack_Obama`. Note that the pairs (`Barack Obama`, `Obama`) and (`Michelle Obama`, `Obama`) are aliases, but of course (`Michelle Obama`, `Barack Obama`) are not. In other words, the **aliases** relation is not transitive. The task of learning a string similarity measure is to produce a function which maps a pair of strings to a similarity score proportional to how likely the pair of strings are aliases.

2.2 Modeling Entity Aliases

Given two mention spellings m and m', the similarity score is based on the "alignment" of m and m'. Classic string alignment algorithms including Levenshtein distance, Longest Common Subsequence, Needleman-Wunsch [26], and Smith-Waterman [30] consider a similarity or "soft alignment" matrix

S of dimension $\max(|m|, |m'|) \times \max(|m|, |m'|)$ and an algorithm for scoring the alignment by a hard alignment matrix D. The value of S_{ij} is the similarity score between the i^{th} character of m (i.e. m_i) and the j^{th} character of m'. The matrix D is typically defined with the recursive formula $D_{ij} = S_{ij} \cdot \mathbb{I}[m_i = m'_j] + \mathbb{I}[m_i \neq m'_j] \cdot \max(w^{(1)}_{ij} D_{i-1,j}, w^{(2)}_{i-1,j} D_{i,j-1}, w^{(3)}_{ij} D_{i-1,j-1})$. $\mathbb{I}$ is the indicator function and the weights $w^{(1)}_{\cdot,\cdot}$ refer to deletion, $w^{(2)}_{\cdot,\cdot}$ to insertion, and $w^{(3)}_{\cdot,\cdot}$ to substitution, all of which are parameterized globally or at a particular position in the matrix. The alignment score is the value $D_{|m|,|m'|}$, computed by dynamic programming. Rather than using fixed values for the similarity matrix and the dynamic program, we learn the values of the similarity values in the alignment matrix as well as a method for scoring the alignment using labeled alias data extracted automatically from publicly available knowledge bases.

Alignment Matrix - Character Level RNN The similarity matrix is based on an embedding-based approach that represents characters in a way that is dependent on their surrounding characters in the sequence. We do this by learning character representations with a bidirectional long short-term memory network (LSTM) [18, 19] applied to the mention spelling m. For each character m_i in m, the model encodes the character with the d dimensional vector, h_i, that is the concatenation of the i^{th} hidden state of the LSTM in both directions. For the mention spelling m, we stack the hidden states of the bidirectional LSTM together to form a matrix $H_m \in \mathbb{R}^{L \times d}$ where L is the maximum string length considered by our models. Note that only the first $|m|$ rows of the matrix will contain values and the remaining rows will be filled with zeros. For two mentions m and m' we simply compute the alignment matrix by $S = H_m H^T_{m'}$.

Alignment Models Given the similarity matrix S as defined, we want to compute a score associated with whether or not m and m' are aliases. Our proposed approach uses a convolutional neural network (CNN) [22] applied to S. The CNN approach provides several advantages over the dynamic program. The CNN is more computationally efficient. It also is able to capture multiple patterns of alignment and longer range dependencies in the strings. For instance, it is quite natural for the CNN to discover an alignment between `Obama, Barack` and `Barack Obama`, which would score low in many standard edit models that operate over the sequence in a linear way. The CNN can detect these two independently aligned regions and determine if the pattern corresponds to a match. Our particular CNN architecture is a three layer network with decreasing filter sizes. A linear model is used to score the final output of the CNN. In some ways, this strategy is closely related to the block alignment approach to longest common subsequence in which regions of the string are identified independently and then merged together. See Figure 2 for a pictoral representation of the CNN model.

Alternative Approaches The model consists of two subcomponents: the character RNN which produces the similarity matrix, and the CNN which is akin to a scoring mechanism. We compare this approach to alternatives that serve as an ablation study. In Section 3, we replace the learned similarity matrix with a binary matrix containing ones where the two strings share the same character and zeroes elsewhere. We also compare to an approach that uses the learned similarity matrix, but does not use the CNN architecture, just a linear layer. Lastly, we compare to an approach that only uses the similarity between the last hidden states of the LSTM applied to m and m'. This is exactly the value stored in the cell $S_{|m|,|m'|}$ of the learned similarity matrix.

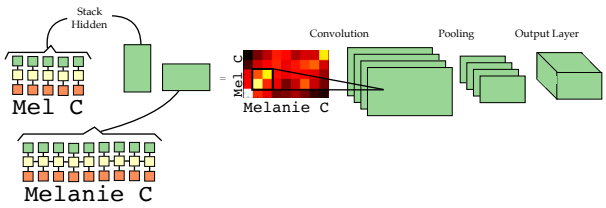

Figure 2: CNN Alignment Model

Training Objective We train each network on triples of mention strings (s, t, n) where s is a mention string, t is a string that is an alias of s, and n is a string that is not an alias of s. We use the Bayesian Personalized Ranking objective [28]: $\sigma(f(s, t) - f(s, n))$. The training triples come from one of the publicly available knowledge bases containing entity aliases as described in Section 3.2

Mention Query	Predictions		
	AlignCNN	Jaro-Winkler	LCS
Bill Hicks	Hicks, Bill	Hicks, Bill	Bill Hicks and His Sizzlin' Six
	Hawks, Billy	Hocus Pocus	William Melvin Hicks
	Bill Hicks and His Sizzlin' Six	The Zombies	Bill Challis and His Orchestra
	Hicks, William Melvin	Jim Reevers	Bill Hickey
Abdul Kalam	Late APJ Abdul Kalam	Abul Kalam	Dr.A.P.J. Abdul Kalam
	Dr A P J Abdul Kalam	Abdul Salam	Late APJ Abdul Kalam
	President Dr. A.P.J. Abdul Kalam	Abdul Kahar	Abdul Khalid bin Ibrahim
	Dr A. P. J. Abdul Kalam	Abdul Karim	Dr. APJ Abdul Kalam

Figure 3: Example Nearest Neighbors according to String Edit Models

3 Experiments

3.1 Task and Metrics

As described in Section 2.1, the focus of this paper is on predicting whether or not two mentions are *aliases* based solely on the string spelling of the mentions. We evaluate models for detecting aliases with the following retrieval-based evaluation: given a mention spelling as a *query*, rank among a set of candidate aliases based on how likely the candidate is an alias of the query. We evaluate the predicted ranking with respect to the ground truth labeled aliases on the Mean Average Precision and Hits at K={1, 10, 50}.

3.2 Datasets

To the best of our knowledge, there are no existing large scale datasets for this particular task, and so we constructed training and evaluation datasets from public knowledge bases, which contain entity alias information. Our evaluation datasets are the following: **Wikipedia** – all pages in Wikipedia [5] (with the exception of redirects, disambiguation, and talk pages) are considered to be entities. For each entity, we extract spans of text in Wikipedia hyperlinked to that entity's page as aliases for that entity. **Wikipedia-People** – a filtered version of the Wikipedia dataset restricted to entities with the type `person` in Freebase [10]. **Patent Assignee** – The National Bureau of Economic Research provides entity information for assignees in United States patents [2]. We align this data with the non-disambiguated assignee field in the patent records available in PatentsView [3] to determine the aliases of each entity. **Music Artist** – The MusicBrainz [32] database contains alternative name spellings and aliases for music artists. **Diseases** – The Comparative Toxicogenomics Database contains alternative name spellings and aliases for disease entities [13].

We use a simple automated process to convert the entity alias information into a dataset for mention alias detection. For each dataset, we randomly split the entities evenly into three groups for training, development, and test. We sample mention queries by first sampling an entity[1]. We then sample a mention alias of this entity to be the *query*. We determine which mention spellings in the knowledge base are aliases of the query (true positives). This follows the definition given in Section 2.1; this is the set of aliases of the entities to which the query is known to refer. Lastly, we carefully select a subset of spellings that are not aliases of the query. We select five types of negative examples, which are illustrated in Figure 4. The groups are (1) strings which are a Levenshtein distance of 1 or 2 of the query, which are not true positives; (2) strings which share a 4-gram word prefix or suffix with the query and are not true positives; (3) aliases of the true positives, which are themselves not true positives (i.e. cases where transitivity does not hold); (4) the aliases of the strings in set (3), which are not true positives; (5) strings chosen at random which are not true positives.

3.3 Methods

We compare the following methods in our experiments: **Classic String Similarity Approaches** – Levenshtein Distance (Lev), Jaro-Winkler distance (JW), Longest Common Subsequence (LCS). **Phonetic Relaxation** – We apply Soundex phonetic mapping to strings and then perform Levenshtein

[1] Proportional to a measure of frequency or popularity of that entity when possible. For the Wikipedia-based datasets this is estimated by the number of hyperlink spans linking to the entity. For the Assignee dataset, this is estimated by the number of patents held by the entity. For the Music Artist dataset, it is done by the entity occurrences in the Last-FM-1k dataset [1, 11]. For the disease dataset, we do not have this information and sample uniformly at random.

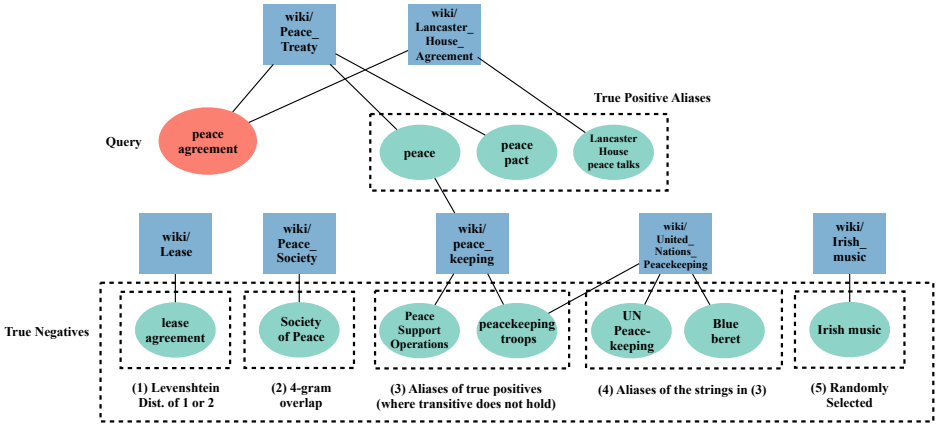

Figure 4: Illustration of true positive aliases and the five types of true negatives used in evaluation and described in the text. The figure depicts the source knowledge base with mentions as ovals, entities as squares, and the query in a red oval. Links indicate that an entity is referred to by that mention.

Dataset	Lev	JW	LCS	Sdx	AlignDot	AlignLinear	AlignBinary	AlignCNN
Wiki	0.238	0.297	0.332	0.294	0.230	0.208	0.287	0.340
WikiPPL	0.246	0.283	0.397	0.308	0.328	0.234	0.411	0.538
Assignee	0.720	0.850	0.622	0.733	0.790	0.797	0.838	0.910
Artist	0.296	0.328	0.293	0.354	0.399	0.250	0.403	0.475
Disease	0.206	0.244	0.191	0.259	0.247	0.230	0.252	0.360

Table 1: Mean Average Precision Results

distance on the mapped results (Sdx). **AlignCNN** – Our approach using a learned similarity matrix with the CNN based alignment scoring model. **AlignLinear** – Our approach using a learned similarity matrix with a linear alignment model. **AlignBinary** – Our approach using a binary similarity matrix ($S_{ij} = \mathbb{I}[m_i = m'_j]$) and CNN alignment model. **AlignDot** – Our approach that uses the dot product between the last hidden state of m and m' as the alignment score (i.e. $S_{|m||m'|}$).

3.4 Alias Detection Evaluation

Table 1 provides the MAP results for each of the methods across the datasets. Table 2 provides the Hits at 1, 10, and 50 results. We compare the methods across each of the datasets and observe that the AlignCNN performs the best (and often offering significant improvement) on nearly all datasets and metrics. We hypothesize that this is due to the CNN's ability to capture more complex patterns by means of the alignment matrix. In each experiment, we optimize the hyperparameters of the models on the dev sets using a grid search over embedding dimension, learning rate and number of filters. The AlignCNN model uses a three layer CNN with filter sizes of 7,5, and 5, max-pooling, embedding dimension of 100, and RNN hidden state size of 200. All models were implemented in PyTorch [4] and our implementation is available here [2].

4 Related Work

Sequence modeling and alignment is a widely studied problem in both theoretical and applied computer science and is too vast to be properly covered in this section. We note that the most relevant work in this area to this paper is other work on learned string edit models including the work of McCallum et al [25] which uses a CRF based model, and Bilenko et al which uses a SVM based model [9]. A generative version of a similar model was developed by Andrews et al [6] and also used for joint cross document coreference and string edit modeling tasks [7]. Closely related work also appears in computational morphology [14, 15, 27]. Much of this work uses weighted finite state transducers with learned parameters. This is a more complicated model class than our character-

[2] `https://github.com/iesl/learned-string-alignments`

Dataset	Hits at	Lev	JW	LCS	Sdx	AlignDot	AlignLinear	AlignBinary	AlignCNN
Wiki	1	0.553	0.630	0.569	0.545	0.436	0.358	0.509	0.586
	10	0.38	0.471	0.450	0.381	0.383	0.355	0.444	0.515
	50	0.373	0.488	0.441	0.366	0.448	0.446	0.507	0.556
WikiPPL	1	0.434	0.506	0.570	0.422	0.421	0.300	0.550	0.680
	10	0.397	0.397	0.475	0.323	0.469	0.357	0.544	0.665
	50	0.417	0.488	0.517	0.370	0.745	0.547	0.672	0.745
Assignee	1	0.850	0.920	0.726	0.808	0.863	0.821	0.870	0.932
	10	0.805	0.896	0.738	0.746	0.870	0.879	0.904	0.950
	50	0.847	0.930	0.817	0.789	0.927	0.940	0.946	0.970
Artist	1	0.442	0.475	0.417	0.382	0.46	0.251	0.483	0.562
	10	0.369	0.386	0.398	0.328	0.538	0.388	0.525	0.581
	50	0.448	0.506	0.502	0.430	0.707	0.595	0.682	0.743
Disease	1	0.514	0.517	0.458	0.451	0.449	0.314	0.381	0.505
	10	0.266	0.300	0.285	0.26	0.329	0.334	0.349	0.475
	50	0.305	0.395	0.371	0.324	0.470	0.497	0.511	0.604

Table 2: Hits at K Results

character interaction matrix and we hope to consider these models with richer transformations in future work. Similar neural network architectures have been used for related sequence alignment problems. The Match-SRNN computes a similarity matrix over two sentence representations and uses an RNN applied to the matrix in a manner akin to the classic dynamic program for question answering and IR tasks [34]. A similar RNN-based alignment approach was also used for phoneme recognition [17]. Character-level models have been the study of many previous works [21, 31]. String similarity models are crucial to record linkage, deduplication, and entity linking tasks. These include author coreference [23], record linkage in databases [24], and record linkage systems with impactful downstream applications [29].

5 Conclusion and Discussion

In this paper, we described and evaluated methods for learning a string similarity measure to detect whether two mention strings could refer to the same entity. We presented an approach based on learning character-level representations with neural networks as well as learning to align these representations with convolutional networks. There is room for much future work in this area, including: using a relaxed/differentiable version of the classic string alignment dynamic program in place of the CNN, jointly using the string edit modeling approach in this paper in an entity resolution system similar to Andrews et al [7], and finally using richer input structures, such as FST states, to the CNN alignment model.

Acknowledgments

This work was supported in part by the UMass Amherst Center for Data Science and the Center for Intelligent Information Retrieval, in part by DARPA under agreement number FA8750-13-2-0020, in part by Amazon Alexa Science, in part by Defense Advanced Research Agency (DARPA) contract number HR0011-15-2-0036, in part by the National Science Foundation (NSF) grant numbers DMR-1534431 and IIS-1514053 and in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction. The work reported here was performed in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

[1] Last.fm. `https://www.last.fm/`.

[2] National bureau of economic research. `https://sites.google.com/site/patentdataproject/Home/downloads`.

[3] PatentsView. http://www.patentsview.org/.

[4] PyTorch. http://pytorch.org.

[5] Wikipedia. https://dumps.wikimedia.org/enwiki/, 03 2015.

[6] Nicholas Andrews, Jason Eisner, and Mark Dredze. Name phylogeny: A generative model of string variation. *EMNLP*, 2012.

[7] Nicholas Andrews, Jason Eisner, and Mark Dredze. Robust entity clustering via phylogenetic inference. *ACL*, 2014.

[8] Lasse Bergroth, Harri Hakonen, and Timo Raita. A survey of longest common subsequence algorithms. *String Processing and Information Retrieval*, 2000.

[9] Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. *KDD*, 2003.

[10] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. *ICDM*, 2008.

[11] O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[12] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. *KDD workshop on data cleaning and object consolidation*, 2003.

[13] Allan Peter Davis, Cynthia J Grondin, Kelley Lennon-Hopkins, Cynthia Saraceni-Richards, Daniela Sciaky, Benjamin L King, Thomas C Wiegers, and Carolyn J Mattingly. The comparative toxicogenomics database's 10th year anniversary: update 2015. *Nucleic acids research*, 43(D1):D914–D920, 2014.

[14] Markus Dreyer, Jason R Smith, and Jason Eisner. Latent-variable modeling of string transductions with finite-state methods. *EMNLP*, 2008.

[15] Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. Morphological inflection generation using character sequence to sequence learning. *NAACL*, 2016.

[16] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. *VLDB*, 1999.

[17] Alex Graves. Sequence transduction with recurrent neural networks. *Representation Learning Worksop, ICML*, 2012.

[18] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 2005.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[20] Kunho Kim, Madian Khabsa, and C Lee Giles. Random forest dbscan for uspto inventor name disambiguation. *JCDL*, 2016.

[21] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-aware neural language models. *AAAI*, 2016.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Michael Levin, Stefan Krawczyk, Steven Bethard, and Dan Jurafsky. Citation-based bootstrapping for large-scale author disambiguation. *Journal of the American Society for Information Science and Technology*, 2012.

[24] Pei Li, Xin Luna Dong, Songtao Guo, Andrea Maurino, and Divesh Srivastava. Robust group linkage. *WWW*, 2015.

[25] Andrew McCallum, Kedar Bellare, and Fernando Pereira. A conditional random field for discriminatively-trained finite-state string edit distance. *UAI*, 2005.

[26] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 1970.

[27] Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. Weighting finite-state transductions with neural context. *NAACL*, 2016.

[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *UAI*, 2009.

[29] Peter Sadosky, Anshumali Shrivastava, Megan Price, and Rebecca C Steorts. Blocking methods applied to casualty records from the syrian conflict. *arXiv preprint arXiv:1510.07714*, 2015.

[30] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[31] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. *ICML*, 2011.

[32] Aaron Swartz. Musicbrainz: A semantic web service. *IEEE Intelligent Systems*, 2002.

[33] Samuel L. Ventura, Rebecca Nugent, and Erica R.H. Fuchs. Seeing the non-stars: (some) sources of bias in past disambiguation approaches and a new public tool leveraging labeled records. *Research Policy*, 2015.

[34] Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. Match-srnn: Modeling the recursive matching structure with spatial rnn. *IJCAI*, 2016.

[35] William E Winkler. The state of record linkage and current research problems. *Statistical Research Division, US Census Bureau*, 1999.