Optimal Transport-based Alignment of Learned Character Representations for String Similarity

Derek Tam[1], **Nicholas Monath**[1], **Ari Kobren**[1],
Aaron Traylor[2], **Rajarshi Das**[1], **Andrew McCallum**[1]
[1]College of Information and Computer Sciences, University of Massachusetts Amherst
[2]Department of Computer Science, Brown University
{dptam,nmonath,akobren,rajarshi,mccallum}@cs.umass.edu
aaron_traylor@brown.edu

Abstract

String similarity models are vital for record linkage, entity resolution, and search. In this work, we present STANCE–a *learned* model for computing the similarity of two strings. Our approach encodes the characters of each string, aligns the encodings using Sinkhorn Iteration (alignment is posed as an instance of optimal transport) and scores the alignment with a convolutional neural network. We evaluate STANCE's ability to detect whether two strings *can* refer to the same entity–a task we term *alias detection*. We construct five new alias detection datasets (and make them publicly available). We show that STANCE (or one of its variants) outperforms both state-of-the-art and classic, parameter-free similarity models on four of the five datasets. We also demonstrate STANCE's ability to improve downstream tasks by applying it to an instance of cross-document coreference and show that it leads to a 2.8 point improvement in B^3 F1 over the previous state-of-the-art approach.

1 Introduction

String similarity models are crucial in record linkage, data integration, search and entity resolution systems, in which they are used to determine whether two strings refer to the same *entity* (Bilenko and Mooney, 2003; McCallum et al., 2005; Li et al., 2015). In the context of these systems, measuring string similarity is complicated by a variety of factors including: the use of nicknames (e.g., `Bill Clinton` instead of `William Clinton`), token permutations (e.g., `US Navy` and `Naval Forces of the US`) and noise, among others. Many state-of-the-art systems employ either classic similarity models, such as Levenshtein, longest common subsequence, and Jaro-Winkler, or *learned* models for string similarity (Levin et al., 2012; Li et al., 2015; Ventura et al., 2015; Kim et al., 2016a; Gan et al., 2017).

While classic and learned approaches can be effective, they both have a number of shortcomings. First, the classic approaches have few parameters making them inflexible and unlikely to succeed across languages or across domains with unique characteristics (e.g. company names, music album titles, etc.) (Needleman and Wunsch, 1970; Smith and Waterman, 1981; Winkler, 1999; Gionis et al., 1999; Bergroth et al., 2000; Cohen et al., 2003). Classic models also assume that each edit has equal cost, which is unrealistic. For example, consider the names `Chun How` and `Chun Hao`–which can refer to the same entity–and the names `John A. Smith` and `John B. Smith`, which cannot. Even though the first pair differ by 2 edits and the second pair by 1, transforming `ow` to `ao` in the first pair should cost less than transforming `A` to `B` in the second. Learned string similarity models address these problems by learning distinct costs for various edits and have thus proven successful in a number of domains (Bilenko and Mooney, 2003; McCallum et al., 2005; Gan et al., 2017). Some learned string similarity models, such as the SVM (Bilenko and Mooney, 2003) and CRF-based (McCallum et al., 2005) approaches, use edit patterns akin to insertions/swaps/deletions, which may lead to strong inductive biases. For example, even when costs are learned, two strings related by a token permutation–e.g., `Grace Hopper` and `Hopper, Grace`–are likely to have high cost even though they clearly refer to the same entity. Gan et al. (2017), on the other hand, provide less structure, encoding each string with a single vector embedding and measuring similarity between the embedded representations.

In this paper, we present a learned string similarity model that is flexible, captures sequential dependencies of characters, and is readily able to learn a wide range of edit patterns–such as token permutations. Our approach is comprised of three

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5907–5917
Florence, Italy, July 28 - August 2, 2019.

components: the first encodes each character in both strings using a recurrent neural network; the second softly aligns the two encoded sequences by solving an instance of optimal transport; the third scores the alignment with a convolutional neural network. Each component is differentiable, allowing for end-to-end training. Our model is called STANCE–an acronym that stands for: **S**imilarity of **T**ransport-**A**ligned **N**eural **C**haracter **E**ncodings.

We evaluate STANCE's ability to capture string similarity in a task we term *alias detection*. The input to alias detection is a query *mention* (i.e., a string) and a set of candidate mentions, and the goal is to score query-candidate pairs that *can* refer to the same *entity* higher than pairs that cannot. For example, an accurate model scores the query `Philips` with candidates `Philips Corporation` and `Katherine Philips` higher than with `M. Phelps`. Alias detection differs from both coreference and entity linking in that neither surrounding natural language context of the mention nor external knowledge are available. A similar task is studied in recent work (Gan et al., 2017).

In experiments, we compare STANCE to state-of-the-art and classic models of string similarity in alias detection on 5 newly constructed datasets–which we make publicly available. Our results demonstrate that STANCE outperforms all other approaches on 4 out of 5 datasets in terms of Hits@1 and 3 out of 5 datasets in terms of mean average precision. Of the two cases in which STANCE is outperformed by other methods in terms of mean average precision, one is by a variant of STANCE in an ablation study. We also demonstrate STANCE's capacity for supporting downstream tasks by using it in cross-document coreference for the Twitter at the Grammy's dataset (Dredze et al., 2016). Using STANCE improves upon the state-of-the-art by 2.8 points of B^3 F1. Analyzing our trained model reveals STANCE effectively learns sequence-aware character similarities, filters noise with optimal transport, and uses the CNN scoring component to detect unconventional similarity-preserving edit patterns.

2 STANCE

Our goal is to learn a model, $f(\cdot, \cdot)$, that measures the similarity between two strings–called *mentions*. The model should produce a high score when its inputs are *aliases* of the same entity, where a mention is an alias of an entity if it can be used to refer to that entity. For example, the mentions `Barack H. Obama` and `Barry Obama` are both aliases of the entity `wiki/Barack_Obama`. Note that the alias relationship is not transitive: both of the pairs `Obama-Barack Obama` and `Obama-Michelle Obama` are aliases of the same entity, but the pair `Barack Obama-Michelle Obama` are not.

In this section we describe our proposed model, STANCE, which is comprised of three stages: encoding both mentions and constructing a corresponding similarity matrix, softly aligning the encoded mentions, and scoring the alignment.

2.1 Mention Encoding Similarity Matrix

A flexible string similarity model is sequence-aware, i.e., the cost of each character transformation should depend on the surrounding characters (e.g., transforming `Chun How` to `Chun Hao` should have low cost). To capture these sequential dependencies, STANCE encodes each mention using a bidirectional long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005). In particular, each character c_i in a mention m is represented by a d-dimensional vector, h_i, where h_i is the concatenation of the hidden states corresponding to c_i produced by running the LSTM in both directions. The encoded representations of the characters are stacked to form a matrix $H^{(m)} \in \mathbb{R}^{L \times d}$ where L (a hyperparameter) is the maximum string length considered by STANCE.

Given a query m and candidate m', STANCE computes a *similarity matrix* of their encodings via an inner product: $S = H^{(m)} H^{(m')\mathrm{T}}$. Each cell in the resultant matrix represents a measure of the similarity between each pair of character encodings from m and m'. Note that for a mention q only the first $|q|$ (i.e., length of the string q) rows of $H^{(q)}$ contain non-zero values.

2.2 Soft Alignment via Optimal Transport

The next component of our model computes a soft alignment between the characters of m and m'. Aligning the mentions is posed as a *transport problem*, where the goal is to convert one mention into another while minimizing cost. In particular, we solve the Kantorovich formulation of optimal transport (OT). In this formulation, two probability measures, p_1 and p_2 are given in addition to a cost matrix, C. This matrix defines the cost of moving

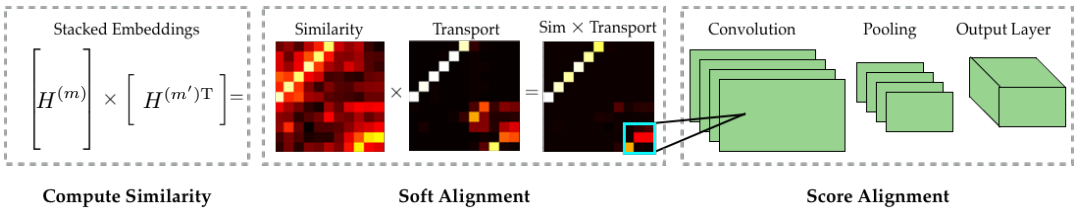
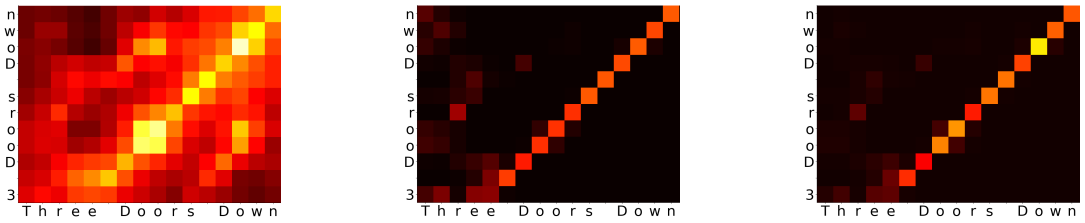


Figure 1: **STANCE Model architecture:** Character Similarities (§2.1), soft alignment (§2.2), and scoring (§2.3)



(a) Similarity Matrix (b) Transport Matrix (c) Similarity × Transport

Figure 2: **Three Heatmaps:** in all three heatmaps, brighter cells correspond to higher similarity. Figure 2a visualizes the character similarity matrix for two mentions: `Three Doors Down` and `3 Doors Down`. Figure 2b visualizes the transport matrix and Figure 2c visualizes the element-wise product of the similarity and transport matrices. Many of the characters are highly similar. Multiplying by the transport matrix amplifies the alignment of the mentions while reducing noise, resulting in a clean alignment for the CNN scoring component.

(or converting) each element in the support of p_1 to each element in the support of p_2. The solution to OT is a matrix, $\hat{P}$, called the *transport plan*, which defines how to completely convert p_1 into p_2. A viable transport plan is required to be non-negative and is also required to have marginals of p_1 and p_2 (i.e., if $\hat{P}$ is summed along the rows then p_1 is recovered and if it is summed along the columns p_2 is recovered). The goal is to find the plan with minimal cost,

$$P^\star = \underset{P \in \mathcal{P}}{\operatorname{argmin}} \sum_{i=0}^{|p_1|} \sum_{j=0}^{|p_2|} C_{ij} P_{ij}$$

$$\mathcal{P} = \{P \in \mathbb{R}_+^{L \times L} \mid P\mathbf{1}_L = p_1,\ P^T\mathbf{1}_L = p_2\}$$

where $|\cdot|$ is the number of elements in the support of the corresponding distribution and $\mathcal{P}$ is the set of valid transportation plans. In this sense, a transportation plan can be thought of as a soft alignment of the supports of p_1 and p_2 (i.e., an element in p_1 can be aligned fractionally to multiple elements in p_2). A transportation plan can be computed efficiently via Sinkhorn Iteration exploiting parallelism using GPUs (empirically it has been shown to be quadratic in L) (Cuturi, 2013). The transport plan is defined as $P = \mathrm{diag}(\boldsymbol{u})K\mathrm{diag}(\boldsymbol{v})$ where $K := e^{-\lambda C}$, $\boldsymbol{u}$ and $\boldsymbol{v}$ are found using the iterative algorithm, λ is the entropic regularizer, and $\mathrm{diag}(\cdot)$ gives a matrix with its input argument as the diagonal (Cuturi, 2013). We specifically use the regularized objective that has been shown to be effective for training (Cuturi, 2013; Genevay et al., 2018).

Optimal transport has been effectively used in several natural language-based applications such as computing the similarity between two documents as the transport cost (Kusner et al., 2015; Huang et al., 2016), in measuring distances between point cloud-based representations of words (Frogner et al., 2019), and learning correspondences between word embedding spaces across domains/languages (Alvarez-Melis and Jaakkola, 2018; Alvarez-Melis et al., 2019).

In our case, p_1 represents the mention m and p_2 represents m'. The distribution p_1 is defined as a point cloud consisting of the character embeddings computed by the LSTM applied to m, i.e., $H^{(m)}$. Formally, it is a set of evenly weighted Dirac Delta functions in $\mathbb{R}^d$ where d is the embedding dimensionality of the character representations. The distribution p_2 is defined similarly for m'. The cost of transporting a character, c_i of m to a character c_j of m' has cost, $C_{i,j} = S_{\max} - S_{i,j}$ where $S_{\max} = \max_{i',j'} S_{i',j'}$ and $S_{i,j}$ is the inner product of h_i and h_j. The resulting transport plan is multiplied by the similarity matrix (Section 2.1) and subsequently fed as input to the next component of our model (Section 2.3). Despite being a soft alignment, this step helps mitigate spurious errors

by reducing the similarity of characters pairs that are not aligned.

2.3 Alignment Score

The transport plan, $\hat{P} \in \mathbb{R}_{+}^{L \times L}$ describes how the characters in m are softly aligned to the characters in m'. We compute the element-wise product of the similarity matrix, S, and the transport plan: $S' = S \circ \hat{P}$. Cells containing high values in S' correspond to similar character pairs from m and m' that are also well-aligned.

Note the distinction between this alignment and the way in which the transport cost can be used as distance measure. The alignment is used as a re-weighting of the similarity matrix. In this way, the transport plan is closely related to attention-based models (Bahdanau et al., 2015; Parikh et al., 2016; Vaswani et al., 2017; Kim et al., 2017).

Finally, we employ a two dimensional convolutional neural network (CNN) to score S' (LeCun et al., 1998). With access to the full matrix S', the CNN is able to detect multiple, aligned, character subsequences from m and m' that are highly similar. By combining evidence from multiple–potentially non-continguous–aligned character subsequences, the CNN detects long-range similarity-preserving edit patterns. This is crucial, for example, in computing a high score for the pair `Obama, Barack` and `Barack Obama`.

The architecture of the alignment-scoring CNN is a three layer network with filters of fixed size. A linear model is used to score the final output of the CNN. See Figure 1 for a visual representation of the STANCE architecture.

Training We train on mention triples, (q, p, n), where there exists an entity for which q and p are both aliases (i.e., (q, p) is a positive example), and there does not exist an entity for which both q and n are aliases (i.e., a negative example). We use the Bayesian Personalized Ranking objective (Rendle et al., 2009): $\sigma(f(q, p) - f(q, n))$.

3 Alias Detection

String similarity is a crucial piece of data integration, search and entity resolution systems, yet there are few large-scale datasets for training and evaluating domain-specific string similarity models. Unlike in coreference resolution, a high quality model should return high scores for mention pairs

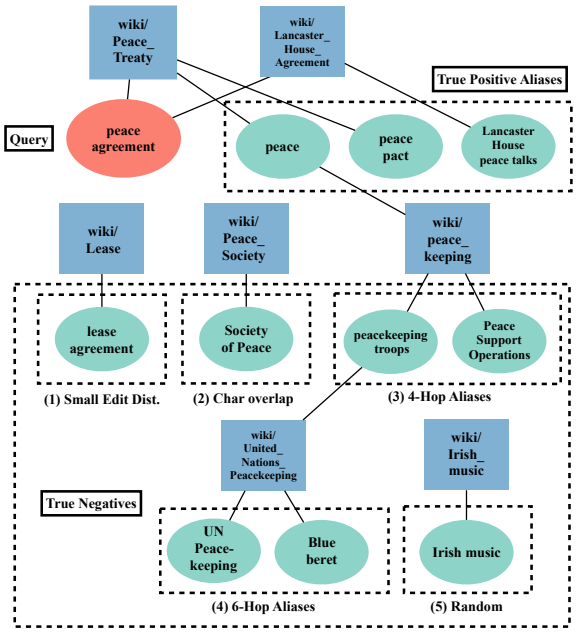


Figure 3: **True positive and negative aliases**. A depiction of the source KB with mentions as ovals, entities as squares, and the query in a red oval. Links indicate that an entity is referred to by that mention.

in which both strings are aliases of (i.e., *can* refer to) the same entity. For example, the mention `Clinton` should exhibit high score with both `B. Clinton` and `H. Clinton`.

We construct five datasets for training and evaluating string similarity models derived from four large-scale public knowledge bases, which encompass a diverse range of entity types. The five datasets are summarized below:

1. **Wikipedia (W)** – We consider pages in Wikipedia to be entities. For each entity, we extract spans of text hyperlinked to that entity's page and use these as aliases.[1]
2. **Wikipedia-People (WP)** – The Wikipedia dataset restricted to entities with type `person` in Freebase (Bollacker et al., 2008).
3. **Patent Assignee (A)** – Aliases of assignees (mostly organizations, some persons) found by combining entity information[2] with non-disambiguated assignees in patents[3].
4. **Music Artist (M)** – MusicBrainz (Swartz, 2002) contains alternative names for music artists.

[1] We used a xml dump of Wikipedia from 2016-03-05. We restrict the entities and hyperlinked spans to come from non-talk, non-list Wikipedia pages.

[2] `sites.google.com/site/patentdataproject/Home/downloads`

[3] `www.patentsview.org/`

5. **Diseases (D)** – The Comparative Toxicogenomics Database (Davis et al., 2014) stores alternative names for disease entities.

For each dataset, entities are divided into training, development, and testing sets, such that each entity appears in *only one set*. This partitioning scheme is meant to ensure that performant models capture a general notion of similarity, rather than learning to recognize the aliases of particular entities. Dataset statistics can be found in Table 1.

Most mention-pairs selected uniformly at random are not aliases of the same entity. A model trained on such pairs may learn to always predict "Non-alias." To avoid learning such degenerate models and to avoid test sets for which degenerate models are performant, we carefully construct the training, development and test sets by including a mix of positive and negative examples and by generating negative examples designed to be difficult and practical. We use a mixture of the following five heuristics to generate negative examples:

1. **Small Edit Distance** – mentions with Levenshtein distance of 1 or 2 from the query;
2. **Character Overlap** – mentions that share a 4-gram word prefix or suffix with the query;
3. **4-Hop Aliases** – first, construct a bipartite graph of mentions and entities where an edge between a mention and an entity denotes that the mention is an alias of the entity. Then, sample a mention that is not an alias of an entity for which the query is also an alias, and whose shortest path to the query requires 4 hops in the graph. Note that all mentions 2 hops from the query are aliases of an entity for which the query is also an alias.
4. **6-Hop Aliases** – sample a mention whose shortest path to the query in the bipartite mention-entity graph is 6 hops.
5. **Random** – randomly sample mentions that are not aliases of the entity for which the query is also an alias. We do this by first sampling an entity and then sampling an alias of that entity uniformly at random.

In all cases, we sample such that entities that appear more frequently in the corpus and entities that have a larger number of aliases are more likely to be sampled (intuitively, these entities are more relevant and more challenging). For the Wikipedia-based datasets, we sample entities proportionally to the number of hyperlink spans linking to the entity. For the Assignee dataset, we estimate entity frequency by the number of patents held by the entity. For the Music Artist dataset, entity frequency is estimated by the number of entity occurrences in the Last-FM-1k dataset (Last.fm; Celma, 2010). For the disease dataset, we do not have frequency information and so sampling is performed uniformly at random. For each dataset, 300 queries are selected for use in the development set and 4000 queries for use in the test set. Each query is paired with up to 1000 negative examples of each type mentioned above. For training, we also construct datasets using the approaches above for creating negative examples.

Figure 3 illustrates how negative (and positive) examples are generated for the query `peace agreement` (which is used to refer to the entities `wiki/Peace_Treaty` and `wiki/Lancaster_House_Agreement`). 4-Hop (negative) aliases include `Peace Support Operations` and `peacekeeping troops` and 6-Hop (negative) examples include `UN Peacekeeping` and `Blue beret`. Note that for each type of negative example, any mention that is a true positive alias of the query is excluded from being a negative example, even if it satisfies one of the above heuristics.

4 Experiments

We evaluate STANCE directly via alias detection and also indirectly via cross document coreference. We also conduct an ablation study in order to understand the contribution of each of STANCE's three components to its overall performance.

4.1 Alias Detection

In the first experiment, we compare STANCE with both classic and learned similarity models in alias detection. Specifically, we compare STANCE to following approaches:

- **Deep Conflation Model (DCM)** – state of the art model that encodes each string using a 1-dimensional CNN applied to character n-grams and computes cosine similarity (Gan et al., 2017). We use the available code [4].
- **Learned Dynamic Time Warping (LDTW)** – encode mentions using a bidirectional LSTM and compute similarity via dynamic time warping (DTW). We note equivalence between LDTW and weighted finite state trans-

[4] `github.com/zhegan27/Deep_Conflation_Model`

Data	Unique Strings	Entity Count	Avg. Num. of Mentions/Ent	Avg. TP/Ent (Dev)	Avg. TP/Ent (Test)
W	9.32×10^6	4.64×10^6	2.54 ± 4.65	125.01 ± 356.45	80.31 ± 317.42
WP	1.88×10^6	1.16×10^6	1.83 ± 2.06	9.82 ± 23.71	10.53 ± 43.35
A	3.30×10^5	2.27×10^5	1.501 ± 2.64	30.76 ± 63.46	11.42 ± 25.02
M	1.83×10^6	1.16×10^6	1.694 ± 3.23	5.08 ± 13.63	9.20 ± 136.28
D	7.69×10^4	1.19×10^4	6.67 ± 9.10	7.21 ± 10.60	7.46 ± 10.72

Table 1: Qualities of the 5 created datasets. True positive are correct entity aliases included in the dev or test set.

	Ours	Alias Detection								Ablation		
Data	STANCE	Lev	JW	LCS	Sdx	CRF	LSTM	DCM	LDTW	-CNN	-LSTM	-OT
W	**.416**	.238	.297	.332	.294	.299	.230	.288	.362	.208	.287	.340
WP	**.594**	.246	.283	.397	.308	.515	.328	.352	.413	.234	.411	.538
A	.906	.720	.850	.622	.733	.780	.790	.782	.903	.797	.838	**.910**
M	**.597**	.296	.328	.293	.354	.319	.399	.509	.396	.250	.403	.475
D	.417	.206	.244	.191	.259	.162	.247	**.437**	.347	.230	.252	.360

Table 2: Mean Average Precision (MAP).

ducers where the transducer topology is the edit distance (insert, delete, swap) program. Parameters are learned such that DTW distance is meaningful (Cuturi and Blondel, 2017).

- **LSTM** – represent each mention using the final hidden state of a bidirectional LSTM. Similarity is the dot product of mention representations (i.e. $S_{|m||m'|}$).
- **Classic Approaches** – Levenshtein Distance (Lev), Jaro-Winkler distance (JW), Longest Common Subsequence (LCS).
- **Phonetic Relaxation (Sdx)** – transform mentions using the Soundex phonetic mapping and then compute Levenshtein.
- **CRF** – implementation [5] of the model defined in (McCallum et al., 2005).

Given a query mention, q, and a set of candidate mentions, we use each model to rank candidates by similarity to q. We compute the mean average precision (MAP) and hits at $k = \{1, 10, 50\}$ of the ranking with respect to a set of ground truth labeled aliases. We report MAP and hits at k averaged over all test queries. The set of candidates for query q include all corresponding positive and negative examples from the test set (Section 3).

For models with hyperparameters, we tune the hyperparameters on the dev set using a grid search over: embedding dimension, learning rate, hidden state dimension, and number of filters (for the CNN). All models were implemented in PyTorch, utilizing SinkhornAutoDiff [6], and optimized with Adam (Kingma and Lei Ba, 2015). Our implementation is publicly available [7].

4.2 Ablation Study

Our second experiment is designed to reveal the purpose of each of STANCE's components. To do so, we compare variants of STANCE with components removed and/or modified. Specifically, we compare the following variants:

- **WITHOUT-OT (-OT)** – STANCE with LSTM encodings and CNN scoring but without optimal transport-based alignment.
- **CNN-TO-LINEAR (-CNN)** – STANCE with the CNN scoring model replaced by a linear scoring model. Again, the optimal transport-based alignment is removed.
- **LSTM-TO-BINARY (-LSTM)** – A binary similarity matrix ($S_{ij} = \mathbb{I}[m_i = m'_j]$) and CNN scoring model, designed to assess the importance of the initial mention encodings. Once more, the optimal transport-based alignment is removed.

We evaluate each model variant using MAP and hits at k on the 5 datasets as in the first experiment. Results can be found in Table 2 and Table 3, respectively. We note that these ablations are equivalent to the models proposed by Traylor et al. (2017).

4.3 Results and Analysis

Table 2 and Table 3 contain the MAP and hits at k (respectively) for each method and dataset (for alias detection and ablation experiments). The results reveal that with the exception of the disease dataset, STANCE (or one of its variants) performs best in terms of both metrics. The results suggest that the

[5] github.com/dirko/pyhacrf

[6] github.com/gpeyre/SinkhornAutoDiff

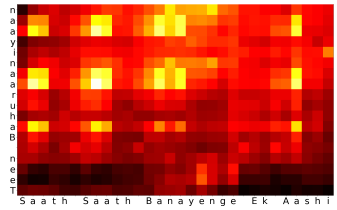
[7] github.com/iesl/stance

		Ours	Alias Detection								Ablation		
Data	K	STANCE	Lev	JW	LCS	Sdx	CRF	LSTM	DCM	LDTW	-CNN	-LSTM	-OT
W	1	**.698**	.553	.630	.569	.545	.599	.436	.610	.570	.358	.509	.586
	10	**.599**	.380	.471	.450	.381	.464	.383	.440	.525	.355	.444	.515
	50	**.604**	.373	.488	.441	.366	.474	.448	.431	.556	.446	.507	.556
WP	1	**.744**	.434	.506	.570	.422	.648	.421	.528	.456	.300	.550	.680
	10	**.708**	.397	.397	.475	.323	.646	.469	.459	.573	.357	.544	.665
	50	**.766**	.417	.488	.517	.370	.716	.745	.546	.729	.547	.672	.745
A	1	**.942**	.850	.920	.726	.808	.867	.863	.881	.926	.821	.870	.932
	10	.932	.805	.896	.738	.746	.840	.870	.841	.947	.879	.904	**.950**
	50	.966	.847	.930	.817	.789	.896	.927	.883	**.970**	.940	.946	**.970**
M	1	**.698**	.442	.475	.417	.382	.465	.460	.614	.406	.251	.483	.562
	10	**.690**	.369	.386	.398	.328	.371	.538	.623	.532	.388	.525	.581
	50	**.806**	.448	.506	.502	.430	.452	.707	.746	.716	.595	.682	.743
D	1	.589	.514	.517	.458	.451	.410	.449	**.630**	.508	.314	.381	.505
	10	**.521**	.266	.300	.285	.260	.232	.329	.499	.455	.334	.349	.475
	50	**.638**	.305	.395	.371	.324	.316	.470	.571	.600	.497	.511	.604

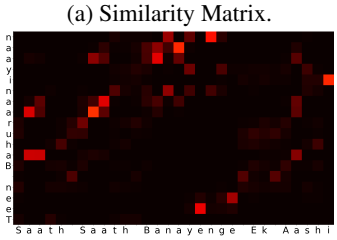
Table 3: Hits at K.

optimal transport and CNN-based alignment scoring components of STANCE lead to a more robust model of similarity than inner-product based models, like **LSTM** and **DCM**. We hypothesize that using n-grams as opposed to individual characters embeddings is advantageous on the disease dataset, leading to **DCM**'s top performance. Surprisingly, -OT is best on the assignee dataset. We hypothesize that this is due to many corporate acronyms.

To better understand STANCE's performance and improvement over the baseline methods we provide analysis of particular examples highlighting two advantages of the model: it leverages optimal transport for noise reduction, and it uses its CNN-based scoring function to learn non-standard similarity-preserving string edit patterns that would be difficult to learn with classic edit operations (i.e., insert, delete and substitute).



(a) Similarity Matrix.



(b) Noise Filtered

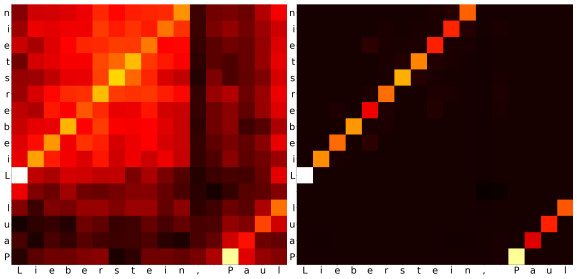
Figure 4: **Noise Filtering:** OT effectively reduces noise in the similarity matrix even when many character n-grams are common to both mentions (Teen Bahuraaniyaan / Saath Saath Banayenge Ek Aashi).

Noise Reduction. Since the model leverages distributed representations for characters, it often discovers many similarities between the characters in two mentions. For example, Figure 4a shows two strings that are not aliases of the same entity. Despite this, there are many regions of high similarity due to multiple instances of the character bigrams `aa`, `an` and `en` in both mentions. In experiments, we find that this leads the -OT model astray. However, STANCE's optimal transport component constructs a transport plan that contains little alignment between the characters in the mentions as seen in Figure 4b, which displays the product of the similarity matrix and the transportation plan. Ultimately, this leads STANCE to correctly predict that the two strings are not similar.

Token Permutation. A natural and frequently occurring similarity-preserving edit pattern that occurs in our datasets is token permutation, i.e., the tokens of two aliases of the same entity are ordered differently in each mention. For example, consider the similarity matrix in Figure 5b. The CNN easily learns that two strings may be aliases of the same entity even if one is a token permutation of the other. This is because it identifies multiple contiguous "diagonal lines" in the similarity matrix. Classic and learned string similarity measures do not learn this relationship easily.

4.4 Cross Document Coreference

We evaluate the impact of using STANCE for in cross-document coreference in the Twitter at



(a) Similarity (b) Similarity x Transport

Figure 5: **Token Permutation:** STANCE learns that token permutations preserve string similarity (Paul Lieberstein / Lieberstein, Paul).

Method	Dev B^3 F1	Test B^3 F1
Ours (HAC + STANCE)	93.5	**82.5**
Green (Spelling Only)	78.0	77.2
Green (with Context)	88.5	79.7
Phylo (Spelling Only)	96.9	72.3
Phylo (with Context)	97.4	72.1
Phylo (with Context & Time)	**97.7**	72.3

Table 4: Cross Document Coreference Results on Twitter at the Grammy's Dataset. Baseline results from (Dredze et al., 2016).

the Grammy's dataset (Dredze et al., 2016). This dataset consists of 4577 mentions of 273 entities in tweets published close in time to the 2013 Grammy awards. We use the same train/dev/test partition with data provided by the authors [8]. The dataset is notable for having significant variation in the spellings of mentions that refer to the same entity. We design a simple cross-document coreference model that ignores the mention context and simply uses STANCE trained on the WikiPPL model. We perform average linkage hierarchical agglomerative clustering using STANCE scores as the linkage function and halt agglomerations according to a threshold (i.e., no agglomerations with linkage below the threshold are performed). We tune the threshold on the development set by finding the value which gives the highest evaluation score (B^3 F1). We compare our method to the previously published state of the art methods (Green (Green et al., 2012) and Phylo (Andrews et al., 2014)). Both of these methods report numbers using their name spelling features alone as well as with context features. We find that our approach outperforms both methods (including those using context features) on the test dataset in terms of B^3 F1 (Table 4).

[8] bitbucket.org/mdredze/tgx

5 Related Work

Classic string similarity methods based on string alignment include Levenshtein distance, Longest Common Subsequence, Needleman and Wunsch (1970), and Smith and Waterman (1981).

Sequence modeling and alignment is a widely studied problem in both theoretical and applied computer science and is too vast to be properly covered entirely. We note that the most relevant prior work focuses on learned string edit models and includes the work of McCallum et al. (2005) which uses a model based on CRFs, and Bilenko and Mooney (2003) which uses a SVM-based model. Andrews et al. (2012, 2014) developed a generative model, which is used for joint cross document coreference and string edit modeling tasks. Closely related work also appears in the field of computational morphology (Dreyer et al., 2008; Faruqui et al., 2016; Rastogi et al., 2016). Much of this work uses WFSTs with learned parameters. JRC-Names (Steinberger et al., 2011; Ehrmann et al., 2017) is a dataset that stores multilingual aliases of person and organization entities.

Similar neural network architectures to our approach have been used for related sequence alignment problems. Santos et al. (2017) uses an RNN to encode toponyms before using a multi-layer perceptron to determine if a pair of toponyms are matching. 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 (Wan et al., 2016). A similar RNN-based alignment approach was also used for phoneme recognition (Graves, 2012). Many previous works have studied character-level models (Kim et al., 2016b; Sutskever et al., 2011).

Alias detection also bears similarity to natural language inference tasks, where instead of aligning characters to determine if two mentions refer to the same entity, the task is to aligns words to determine if two sentences are semantically equivalent (Bowman et al., 2015; Williams et al., 2018).

Optimal transport and the related Wasserstein distance is studied in mathematics, optimization, and machine learning (Peyré et al., 2017; Villani, 2008). It has notably been used in the NLP community for modeling the distances between documents (Kusner et al., 2015; Huang et al., 2016) as the cost of transporting embedded representations of the words in one document to the words of the an-

other, in point cloud-based embeddings (Frogner et al., 2019), and in learning word correspondences across languages and domains. (Alvarez-Melis and Jaakkola, 2018; Alvarez-Melis et al., 2019).

String similarity models are crucial to record linkage, deduplication, and entity linking tasks. These include author coreference (Levin et al., 2012), record linkage in databases (Li et al., 2015), and record linkage systems with impactful downstream applications (Sadosky et al., 2015).

6 Conclusion

In this work, we present STANCE, a neural model of string similarity that is trained end-to-end. The main components of our model are: a character-level bidirectional LSTM for character encoding, a soft alignment mechanism via optimal transport, and a powerful CNN for scoring alignments. We evaluate our model on 5 datasets created from publicly available knowledge bases and demonstrate that it outperforms the baselines in almost all cases. We also show that using STANCE improves upon state of the art performance in cross-document coreference in the Twitter at the Grammy's dataset. We analyze our trained model and show that its optimal transport component helps to filter noise and that is has the capacity to learn non-standard similarity-preserving string edit patterns.

In future work, we hope to further study the connections between our optimal transport-based alignment method and methods based on attention. We also hope to consider connections to work on probabilistic latent representation of permutations and matchings (Mena et al., 2018; Linderman et al., 2018). Additionally, we hope to apply STANCE to a wider-range of entity resolution tasks, for which string similarity is a component of model that considers additional features such as the natural language context of the entity mention.

Acknowledgments

We thank Haw-Shiuan Chang and Luke Vilnis for their helpful discussions. We also thank the anonymous reviewers for their constructive feedback. 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

David Alvarez-Melis and Tommi Jaakkola. 2018. Gromov-wasserstein alignment of word embedding spaces. *Empirical Methods in Natural Language Processing (EMNLP)*.

David Alvarez-Melis, Stefanie Jegelka, and Tommi S. Jaakkola. 2019. Towards optimal transport with global invariances. *Artificial Intelligence and Statistics (AISTATS)*.

Nicholas Andrews, Jason Eisner, and Mark Dredze. 2012. Name phylogeny: A generative model of string variation. *Empirical Methods in Natural Language Processing (EMNLP)*.

Nicholas Andrews, Jason Eisner, and Mark Dredze. 2014. Robust entity clustering via phylogenetic inference. *Association for Computational Linguistics (ACL)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)*.

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

Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. *Knowledge Discovery and Data Mining (KDD)*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. *International Conference on Data Mining (ICDM)*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *Empirical Methods in Natural Language Processing (EMNLP)*.

O. Celma. 2010. *Music Recommendation and Discovery in the Long Tail*. Springer.

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

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems (NeurIPS)*.

Marco Cuturi and Mathieu Blondel. 2017. Soft-dtw: a differentiable loss function for time-series. *International Conference on Machine Learning (ICML)*.

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

Mark Dredze, Nicholas Andrews, and Jay DeYoung. 2016. Twitter at the grammys: A social media corpus for entity linking and disambiguation. *International Workshop on Natural Language Processing for Social Media*.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. *Empirical Methods in Natural Language Processing (EMNLP)*.

Maud Ehrmann, Guillaume Jacquet, and Ralf Steinberger. 2017. Jrc-names: Multilingual entity name variants and titles as linked data. *Semantic Web*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Charlie Frogner, Farzaneh Mirzazadeh, and Justin Solomon. 2019. Learning entropic wasserstein embeddings. *International Conference on Learning Representations (ICLR)*.

Zhe Gan, P. D. Singh, Ameet Joshi, Xiaodong He, Jianshu Chen, Jianfeng Gao, and Li Deng. 2017. Character-level deep conflation for business data analytics. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Aude Genevay, Gabriel Peyré, and Marco Cuturi. 2018. Learning generative models with sinkhorn divergences. *AISTATS*.

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity search in high dimensions via hashing. *Very Large Data Bases (VLDB)*.

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

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

Spence Green, Nicholas Andrews, Matthew R Gormley, Mark Dredze, and Christopher D Manning. 2012. Entity clustering across languages. *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Gao Huang, Chuan Guo, Matt J Kusner, Yu Sun, Fei Sha, and Kilian Q Weinberger. 2016. Supervised word mover's distance. *NeurIPS*.

Kunho Kim, Madian Khabsa, and C Lee Giles. 2016a. Random forest dbscan for uspto inventor name disambiguation. *Joint Conference on Digital Library (JCDL)*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *International Conference on Learning Representations (ICLR)*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016b. Character-aware neural language models. *Association for the Advancement of Artificial Intelligence (AAAI)*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. *International Conference on Machine Learning (ICML)*.

Last.fm. https://www.last.fm/.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

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

Pei Li, Xin Luna Dong, Songtao Guo, Andrea Maurino, and Divesh Srivastava. 2015. Robust group linkage. *The Web Conference (WWW)*.

Scott Linderman, Gonzalo Mena, Hal Cooper, Liam Paninski, and John Cunningham. 2018. Reparameterizing the birkhoff polytope for variational permutation inference. *Artificial Intelligence and Statistics (AISTATS)*.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. *Uncertainty in Artificial Intelligence (UAI)*.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. 2018. Learning latent permutations with gumbel-sinkhorn networks. *International Conference on Learning Representations (ICLR)*.

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

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *Empirical Methods in Natural Language Processing (EMNLP)*.

Gabriel Peyré, Marco Cuturi, et al. 2017. Computational optimal transport. Technical report.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. *Uncertainty in Artificial Intelligence (UAI)*.

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

Rui Santos, Patricia Murrieta-Flores, Pável Calado, and Bruno Martins. 2017. Toponym matching through deep neural networks. *International Journal of Geographical Information Science*.

Temple F Smith and Michael S Waterman. 1981. Identification of common molecular subsequences. *Journal of molecular biology*.

Ralf Steinberger, Bruno Pouliquen, Mijail Kabadjov, Jenya Belyaeva, and Erik van der Goot. 2011. Jrc-names: A freely available, highly multilingual named entity resource. In *International Conference Recent Advances in Natural Language Processing*.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. *International Conference on Machine Learning (ICML)*.

Aaron Swartz. 2002. Musicbrainz: A semantic web service. *IEEE Intelligent Systems*.

Aaron Traylor, Nicholas Monath, Rajarshi Das, and Andrew McCallum. 2017. Learning string alignments for entity aliases. *Workshop on Automated Knowledge Base Construction (AKBC)*.

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

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

Cédric Villani. 2008. *Optimal transport: old and new*.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-srnn: Modeling the recursive matching structure with spatial rnn. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

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