An Empirical Study of Chinese Name Matching and Applications

Nanyun Peng[1] and **Mo Yu**[2] and **Mark Dredze**[1]
[1]Human Language Technology Center of Excellence
Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD, 21218
[2]Machine Intelligence and Translation Lab
Harbin Institute of Technology, Harbin, China
npeng1@jhu.edu, gflfof@gmail.com, mdredze@cs.jhu.edu

Abstract

Methods for name matching, an important component to support downstream tasks such as entity linking and entity clustering, have focused on alphabetic languages, primarily English. In contrast, logogram languages such as Chinese remain untested. We evaluate methods for name matching in Chinese, including both string matching and learning approaches. Our approach, based on new representations for Chinese, improves both name matching and a downstream entity clustering task.

1 Introduction

A key technique in entity disambiguation is name matching: determining if two mention strings could refer to the same entity. The challenge of name matching lies in name variation, which can be attributed to many factors: nicknames, aliases, acronyms, and differences in transliteration, among others. In light of these issues, exact string match can lead to poor results. Numerous downstream tasks benefit from improved name matching: entity coreference (Strube et al., 2002), name transliteration (Knight and Graehl, 1998), identifying names for mining paraphrases (Barzilay and Lee, 2003), entity linking (Rao et al., 2013) and entity clustering (Green et al., 2012).

As a result, there have been numerous proposed name matching methods (Cohen et al., 2003), with a focus on person names. Despite extensive exploration of this task, most work has focused on Indo-European languages in general and English in particular. These languages use alphabets as representations of written language. In contrast, other languages use logograms, which represent a word or morpheme, the most popular being Chinese which uses hanzi (汉字). This presents challenges for name matching: a small number of hanzi represent an entire name and there are tens of thousands of hanzi in use. Current methods remain largely untested in this setting, despite downstream tasks in Chinese that rely on name matching (Chen et al., 2010; Cassidy et al., 2011). Martschat et al. (2012) point out errors in coreference resolution due to Chinese name matching errors, which suggests that downstream tasks can benefit from improvements in Chinese name matching techniques.

This paper presents an analysis of new and existing approaches to name matching in Chinese. The goal is to determine whether two Chinese strings can refer to the same entity (person, organization, location) based on the strings alone. The more general task of entity coreference (Soon et al., 2001), or entity clustering, includes the context of the mentions in determining coreference. In contrast, standalone name matching modules are context independent (Andrews et al., 2012; Green et al., 2012). In addition to showing name matching improvements on newly developed datasets of matched Chinese name pairs, we show improvements in a downstream Chinese entity clustering task by using our improved name matching system. We call our name matching tool Mingpipe, a Python package that can be used as a standalone tool or integrated within a larger system. We release Mingpipe as well as several datasets to support further work on this task.[1]

2 Name Matching Methods

Name matching originated as part of research into record linkage in databases. Initial work focused

[1]The code and data for this paper are available at: https://github.com/hltcoe/mingpipe

on string matching techniques. This work can be organized into three major categories: 1) Phonetic matching methods, e.g. Soundex (Holmes and McCabe, 2002), double Metaphone (Philips, 2000) etc.; 2) Edit-distance based measures, e.g. Levenshtein distance (Levenshtein, 1966), Jaro-Winkler (Porter et al., 1997; Winkler, 1999), and 3) Token-based similarity, e.g. soft TF-IDF (Bilenko et al., 2003). Analyses comparing these approaches have not found consistent improvements of one method over another (Cohen et al., 2003; Christen, 2006). More recent work has focused on learning a string matching model on name pairs, such as probabilistic noisy channel models (Sukharev et al., 2014; Bilenko et al., 2003). The advantage of trained models is that, with sufficient training data, they can be tuned for specific tasks.

While many NLP tasks rely on name matching, research on name matching techniques themselves has not been a major focus within the NLP community. Most downstream NLP systems have simply employed a static edit distance module to decide whether two names can be matched (Chen et al., 2010; Cassidy et al., 2011; Martschat et al., 2012). An exception is work on training finite state transducers for edit distance metrics (Ristad and Yianilos, 1998; Bouchard-Côté et al., 2008; Dreyer et al., 2008; Cotterell et al., 2014). More recently, Andrews et al. (2012) presented a phylogenetic model of string variation using transducers that applies to pairs of names string (supervised) and unpaired collections (unsupervised).

Beyond name matching in a single language, several papers have considered cross lingual name matching, where name strings are drawn from two different languages, such as matching Arabic names (El-Shishtawy, 2013) with English (Freeman et al., 2006; Green et al., 2012). Additionally, name matching has been used as a component in cross language entity linking (McNamee et al., 2011a; McNamee et al., 2011b) and cross lingual entity clustering (Green et al., 2012). However, little work has focused on logograms, with the exception of Cheng et al. (2011). As we will demonstrate in § 3, there are special challenges caused by the logogram nature of Chinese. We believe this is the first evaluation of Chinese name matching.

3 Challenges

Numerous factors cause name variations, including abbreviations, morphological derivations, historical sound or spelling change, loanword formation, translation, transliteration, or transcription error (Andrews et al., 2012). In addition to all the above factors, Chinese name matching presents unique challenges (Table 1):

Examples	**Notes**
许历农 v.s. 許歷農	simplified v.s. traditional
東盟 v.s. 东南亚国家联盟	Abbreviation and traditional v.s. simplified
亚的斯亚贝巴 v.s. 阿迪斯阿貝巴 / iʌ·ti·si·iʌ·bei·bʌ / v.s. / ʌ·ti·si·ʌ·bei·bʌ /	Transliteration of Addis Ababa in Mainland and Taiwan. Different hanzi, similar pronunciations.
佛罗伦萨 v.s. 翡冷翠 / fo·luo·luən·sʌ / v.s. / fei·lɛŋ·ts^{h}ɵʏ /	Transliteration of Florence in Mainland and Hong Kong. Different writing and dialects.
鲁弗斯·汉弗莱 v.s. 韓魯弗 / *lu·fu·sɯ·xan·fu·laɪ* / v.s. / xan·lu·fu /	Transliteration of Humphrey Rufus in Mainland and Hong Kong. The first uses a literal transliteration, while the second does not. Both reverse the name order (consistent with Chinese names) and change the surname to sound Chinese.

Table 1: Challenges in Chinese name matching.

- There are more than 50k Chinese characters. This can create a large number of parameters in character edit models, which can complicate parameter estimation.

- Chinese characters represent morphemes, not sounds. Many characters can share a single pronunciation[2], and many characters have similar sounds[3]. This causes typos (mistaking characters with the same pronunciation) and introduces variability in transliteration (different characters chosen to represent the same sound).

- Chinese has two writing systems (simplified, traditional) and two major dialects (Mandarin, Cantonese), with different pairings in different regions (see Table 2 for the three dominant regional combinations.) This has a significant impact on loanwords and transliterations.

[2] 486 characters are pronounced / *tɕi* / (regardless of tone).

[3] e.g. 庄 and 张 (different orthography) are pronounced similar (/*tʂuaŋ*/ and /*tʂaŋ* /).

Region	Writing System	Dialect
Hong Kong	Traditional	Cantonese
Mainland	Simplified	Mandarin
Taiwan	Traditional	Mandarin

Table 2: Regional variations for Chinese writing and dialect.

4 Methods

We evaluate several name matching methods, representative of the major approaches to name matching described above.

String Matching We consider two common string matching algorithms: Levenshtein and Jaro-Winkler. However, because of the issues mentioned above we expect these to perform poorly when applied to Chinese strings. We consider several transformations to improve these methods.

First, we map all strings to a single writing system: simplified. This is straightforward since traditional Chinese characters have a many-to-one mapping to simplified characters. Second, we consider a pronunciation based representation. We convert characters to pinyin[4], the official phonetic system (and ISO standard) for transcribing Mandarin pronunciations into the Latin alphabet. While pinyin is a common representation used in Chinese entity disambiguation work (Feng et al., 2004; Jiang et al., 2007), the pinyin for an entire entity is typically concatenated and treated as a single string ("string-pinyin"). However, the pinyin string itself has internal structure that may be useful for name matching. We consider two new pinyin representations. Since each Chinese character corresponds to a pinyin, we take each pinyin as a token corresponding to the Chinese character. We call this "character-pinyin". Additionally, every Mandarin syllable (represented by a pinyin) can be spelled with a combination of an initial and a final segment. Therefore, we split each pinyin token further into the initial and final segment. We call this "segmented-pinyin"[5].

Transducers We next consider methods that can be trained on available Chinese name pairs. Transducers are common choices for learning edit distance metrics for strings, and they perform better than string similarity (Ristad and Yianilos, 1998; Andrews et al., 2012; Cotterell et al., 2014). We use the probabilistic transducer of Cotterell et al. (2014) to learn a stochastic edit distance. The model represent the conditional probability $p(y|x;\theta)$, where y is a generated string based on editing x according to parameters θ. At each position x_i, one of four actions (copy, substitute, insert, delete) are taken to generate character y_j. The probability of each action depends on the string to the left of x_i ($x_{(i-N_1):i}$), the string to the right of x_i ($x_{i:(i+N_2)}$), and generated string to the left of y_j ($y_{(j-N_3):j}$). The variables N_1, N_2, N_3 are the context size. Note that characters to the right of y_j are excluded as they are not yet generated. Training maximizes the observed data log-likelihood and EM is used to marginalize over the latent edit actions. Since the large number of Chinese characters make parameter estimation prohibitive, we only train transducers on the three pinyin representations: string-pinyin (28 characters), character-pinyin (384 characters), segmented-pinyin (59 characters).

Name Matching as Classification An alternate learning formulation considers name matching as a classification task (Mayfield et al., 2009; Zhang et al., 2010; Green et al., 2012). Each string pair is an instance: a positive classification means that two strings can refer to the same name. This allows for arbitrary and global features of the two strings. We use an SVM with a linear kernel.

To learn possible edit rules for Chinese names we add features for pairs of n-grams. For each string, we extract all n-grams (n=1,2,3) and align n-grams between strings using the Hungarian algorithm.[6] Features correspond to the aligned n-gram pairs, as well as the unaligned n-grams. To reduce the number of parameters, we only include features which appear in positive training examples. These features are generated for two string representations: the simplified Chinese string (**simplified n-grams**) and a pinyin representation (**pinyin n-grams**), so that we can incorporate both orthographic features and phonetic features. We separately select the best performing pinyin representation (string-pinyin, character-pinyin, segmented-pinyin) on development data

[4]Hong Kong has a romanization scheme more suitable for Cantonese, but we found no improvements over using pinyin. Therefore, for simplicity we use pinyin throughout.

[5]For example, the pinyin for 张 is segmented into / *zh* / and / *ang* /.

[6]We found this performed much better than directly aligning characters or tokens. We also tried n-gram TF-IDF cosine similarity, but it degraded results (Cohen et al., 2003).

Feature Type	Number of Features
Simplified n-grams	~10k
Pinyin n-grams	~9k
Jaccard similarity	6×10
TF-IDF similarity	2×10
Levenshtein distance	2×10
Other	7

Table 3: Features for SVM learning.

for each dataset.

We measure **Jaccard similarity** between the two strings separately for 1,2,3-grams for each string representation. An additional feature indicates no n-gram overlap. The best performing **Levenshtein** distance metric is included as a feature. Finally, we include **other** features for several name properties: the difference in character length and two indicators as to whether the first character of the two strings match and if its a common Chinese last name. Real valued features are binarized.

Table 3 lists the feature templates we used in our SVM model and the corresponding number of features.

5 Experiments

5.1 Dataset

We constructed two datasets from Wikipedia.

REDIRECT: We extracted webpage redirects from Chinese Wikipedia pages that correspond to entities (person, organization, location); the page type is indicated in the page's metadata. Redirect links indicate queries that all lead to the same page, such as "Barack Hussein Obama" and "Barack Obama". To remove redirects that are not entities (e.g. "44th president") we removed entries that contain numerals and Latin characters, as well as names that contain certain keywords.[7] The final dataset contains 13,730 pairs of person names, 10,686 organizations and 5,152 locations, divided into $\frac{3}{5}$ train, $\frac{1}{5}$ development and $\frac{1}{5}$ test.

NAME GROUPS: Chinese Wikipedia contains a handcrafted mapping between the entity name and various transliterations,[8] including for Mainland, Hong Kong and Taiwan. We created two datasets: Mainland-Hong Kong (1288 people pairs, 357 locations, 177 organizations), and Mainland-Taiwan (1500 people, 439 locations, 112 organizations). Data proportions are split as in REDIRECT.

[7]Entries that contain 列表(list), 代表(representative) , 运动 (movement), 问题 (issue) and 维基 (wikipedia).

[8]http://zh.wikipedia.org/wiki/Template:CGroup

Method	Character	prec@1	prec@3	MRR
Levenshtein	original	0.773	0.838	0.821
	simplified	0.816	0.872	0.856
	string-pinyin	0.743	0.844	0.811
	character-pinyin	**0.824**	**0.885**	**0.866**
	segment-pinyin	0.797	0.877	0.849
Jaro-Winkler	original	0.690	0.792	0.767
	simplified	0.741	0.821	0.803
	string-pinyin	0.741	0.818	0.800
	character-pinyin	**0.751**	**0.831**	**0.813**
	segment-pinyin	0.753	0.821	0.808

Table 4: String matching on development data.

5.2 Evaluation

We evaluated performance on a ranking task (the setting of Andrews et al. (2012)). In each instance, the algorithm was given a query and a set of 11 names from which to select the best match. The 11 names included a matching name as well as 10 other names with some character overlap with the query that are randomly chose from the same data split. We evaluate using precision@1,3 and mean reciprocal rank (MRR). Classifiers were trained on the true pairs (positive) and negative examples constructed by pairing a name with 10 other names that have some character overlap with it. The two SVM parameters (the regularizer co-efficient C and the instance weight w for positive examples), as well as the best pinyin representation, were selected using grid search on dev data.

Results For string matching methods, simplified characters improve over the original characters for both Levenshtein and Jaro-Winkler (Table 4). Surprisingly, pinyin does not help over the simplified characters. Segmented pinyin improved over pinyin but did not do as well as the simplified characters. Our method of character pinyin performed the best overall, because it utilizes the phonetic information the pinyin encodes: all the different characters that have the same pronunciation are reduced to the same pinyin representation. Over all the representations, Levenshtein outperformed Jaro-Winkler, consistent with previous work (Cohen et al., 2003).

Compared to the best string matching method (Levenshtein over pinyin characters), the transducer improves for the two name group datasets but does worse on REDIRECT (Table 5). The heterogeneous nature of REDIRECT, including variation from aliases, nicknames, and long-distance re-ordering, may confuse the transducer. The SVM does best overall, improving for all datasets over string matching and

Method	Dataset	prec@1	prec@3	MRR
Levenshtein	REDIRECT	0.820	0.868	0.859
	Mainland-Taiwan	0.867	0.903	0.897
	Mainland-Hong Kong	0.873	0.937	0.911
Transducer	REDIRECT	0.767	0.873	0.833
	Mainland-Taiwan	0.889	0.938	0.921
	Mainland-Hong Kong	**0.925**$^{(*)}$	**0.989**$^{(*)}$	**0.954**$^{(*)}$
SVM	REDIRECT	**0.888**$^{(**)}$	**0.948**$^{(**)}$	**0.924**$^{(**)}$
	Mainland-Taiwan	**0.926**	**0.966**$^{(**)}$	**0.951**$^{(*)}$
	Mainland-Hong Kongs	0.882	0.972	0.928

Table 5: Results on test data. * better than Levenshtein; ** better than all other methods ($p = 0.05$)

Features	Datasets	
	REDIRECT	Name Groups
ALL	0.921	0.966
- Jaccard similariy	0.908	0.929
- Levenshtein	0.919	0.956
- Simplified pairs	0.918	0.965
- Pinyin pairs	0.920	0.960
- Others	0.921	0.962

Table 6: Ablation experiments on SVM features

tying or beating the transducer. Different pinyin representations (combined with the simplified representation) worked best on different datasets: character-pinyin for REDIRECT, segmented-pinyin for Mainland-Hongkong and string-pinyin for Mainland-Taiwan. To understand how the features for SVM affect the final results, we conduct ablation tests for different group of features when trained on person names (only) for each dataset (Table 6). Overall, Jaccard features are the most effective.

Error Analysis We annotated 100 randomly sampled REDIRECT development pairs incorrectly classified by the SVM. We found three major types of errors. 1) Matches requiring external knowledge (43% of errors), where there were nicknames or aliases. In these cases, the given name strings are insufficient for determining the correct answer. These types of errors are typically handled using alias lists. 2) Transliteration confusions (13%) resulting from different dialects, transliteration versus translation, or only part of a name being transliterated. 3) Noisy data (19%): Wikipedia redirects include names in other languages (e.g. Japanese, Korean) or orthographically identical strings for different entities. Finally, 25% of the time the system simply got the wrong answer, Many of these cases are acronyms.

5.3 Entity Clustering

We evaluate the impact of our improved name matching on a downstream task: entity clustering

Method	Dev			Test		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Exact match	84.55	57.46	68.42	63.95	65.44	64.69
Jaro-winkler	84.87	58.35	69.15	70.79	66.21	68.42
Levenshtein	83.16	61.13	70.46	69.56	67.27	68.40
Transducer	**90.33**	**74.92**	**81.90**	73.59	63.70	68.29
SVM	90.05	63.90	74.75	**74.33**	**67.60**	**70.81**

Table 7: Results on Chinese entity clustering.

(cross document coreference resolution), where the goal is identify co-referent named mentions across documents. Only a few studies have considered Chinese entity clustering (Chen and Martin, 2007), including the TAC KBP shared task, which has included clustering Chinese NIL mentions (Ji et al., 2011). We construct an entity clustering dataset from the TAC KBP entity linking data. All of the 2012 Chinese data is used as development, and the 2013 data as test. We use the system of Green et al. (2012), which allows for the inclusion of arbitrary name matching metrics. We follow their setup for training and evaluation (B^3) and use TF-IDF context features. We tune the clustering cutoff for their hierarchical model, as well as the name matching threshold on the development data. For the trainable name matching methods (transducer, SVM) we train the methods on the development data using cross-validation, as well as tuning the representations and model parameters. We include an exact match baseline.

Table 7 shows that on test data, our best method (SVM) improves over all previous methods by over 2 points. The transducer makes strong gains on dev but not test, suggesting that parameter tuning overfit. These results demonstrate the downstream benefits of improved name matching.

6 Conclusion

Our results suggest several research directions. The remaining errors could be addressed with additional resources. Alias lists could be learned from data or derived from existing resources. Since the best pinyin representation varies by dataset, work could automatically determine the most effective representation, which may include determining the type of variation present in the proposed pair, as well as the associated dialect.

Our name matching tool, Mingpipe, is implemented as a Python library. We make Mingpipe and our datasets available to aid future research on this topic.[9]

[9] https://github.com/hltcoe/mingpipe

References

Nicholas Andrews, Jason Eisner, and Mark Dredze. 2012. Name phylogeny: A generative model of string variation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 344–355.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 16–23.

Mikhail Bilenko, Raymond Mooney, William Cohen, Pradeep Ravikumar, and Stephen Fienberg. 2003. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.

Alexandre Bouchard-Côté, Percy Liang, Dan Klein, and Thomas L Griffiths. 2008. A probabilistic approach to language change. In *Advances in Neural Information Processing Systems (NIPS)*, pages 169–176.

Taylor Cassidy, Zheng Chen, Javier Artiles, Heng Ji, Hongbo Deng, Lev-Arie Ratinov, Jing Zheng, Jiawei Han, and Dan Roth. 2011. Cuny-uiuc-sri tac-kbp2011 entity linking system description. In *Text Analysis Conference (TAC)*.

Ying Chen and James Martin. 2007. Towards robust unsupervised personal name disambiguation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 190–198.

Ying Chen, Peng Jin, Wenjie Li, and Chu-Ren Huang. 2010. The chinese persons name disambiguation evaluation: Exploration of personal name disambiguation in chinese news. In *CIPS-SIGHAN Joint Conference on Chinese Language Processing*.

Gang Cheng, Fei Wang, Haiyang Lv, and Yinling Zhang. 2011. A new matching algorithm for chinese place names. In *International Conference on Geoinformatics*, pages 1–4. IEEE.

Peter Christen. 2006. A comparison of personal name matching: Techniques and practical issues. In *IEEE International Conference on Data Mining Workshops*, pages 290–294.

William Cohen, Pradeep Ravikumar, and Stephen Fienberg. 2003. A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*, pages 73–78.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic fsts. In *Association for Computational Linguistics (ACL)*, pages 625–630.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1080–1089.

Tarek El-Shishtawy. 2013. A hybrid algorithm for matching arabic names. *arXiv preprint arXiv:1309.5657*.

Donghui Feng, Yajuan Lü, and Ming Zhou. 2004. A new approach for english-chinese named entity alignment. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 372–379.

Andrew T Freeman, Sherri L Condon, and Christopher M Ackerman. 2006. Cross linguistic name matching in english and arabic: a one to many mapping extension of the levenshtein edit distance algorithm. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 471–478.

Spence Green, Nicholas Andrews, Matthew R. Gormley, Mark Dredze, and Christopher D. Manning. 2012. Entity clustering across languages. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 60–69.

David Holmes and M Catherine McCabe. 2002. Improving precision and recall for soundex retrieval. In *International Conference on Information Technology: Coding and Computing*, pages 22–26.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011. Overview of the tac 2011 knowledge base population track. In *Text Analytics Conference*.

Long Jiang, Ming Zhou, Lee-Feng Chien, and Cheng Niu. 2007. Named entity translation with web mining and transliteration. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1629–1634.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

A Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Sebastian Martschat, Jie Cai, Samuel Broscheit, Éva Mújdricza-Maydt, and Michael Strube. 2012. A multigraph model for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP) and the Conference on Natural Language Learning (CONLL)*, pages 100–106.

James Mayfield, David Alexander, Bonnie J Dorr, Jason Eisner, Tamer Elsayed, Tim Finin, Clayton Fink, Marjorie Freedman, Nikesh Garera, Paul McNamee, et al. 2009. Cross-document coreference resolution: A key technology for learning by reading. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, pages 65–70.

Paul McNamee, James Mayfield, Dawn Lawrie, Douglas W Oard, and David S Doermann. 2011a. Cross-language entity linking. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 255–263.

Paul McNamee, James Mayfield, Douglas W Oard, Tan Xu, Ke Wu, Veselin Stoyanov, and David Doermann. 2011b. Cross-language entity linking in maryland during a hurricane. In *Empirical Methods in Natural Language Processing (EMNLP)*.

L Philips. 2000. The double metaphone search algorithm. *C/C++ Users Journal*, 18(6).

Edward H Porter, William E Winkler, et al. 1997. Approximate string comparison and its effect on an advanced record linkage system. In *Advanced record linkage system. US Bureau of the Census, Research Report*.

Delip Rao, Paul McNamee, and Mark Dredze. 2013. Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, Multilingual Information Extraction and Summarization*, pages 93–115. Springer.

Eric Sven Ristad and Peter N Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544.

Michael Strube, Stefan Rapp, and Christoph Müller. 2002. The influence of minimum edit distance on reference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 312–319.

Jeffrey Sukharev, Leonid Zhukov, and Alexandrin Popescul. 2014. Learning alternative name spellings. *arXiv preprint arXiv:1405.2048*.

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

Wei Zhang, Jian Su, Chew Lim Tan, and Wen Ting Wang. 2010. Entity linking leveraging: Automatically generated annotation. In *International Conference on Computational Linguistics (COLING)*, pages 1290–1298.