Balancing Speed and Accuracy in Neural-Enhanced Phonetic Name Matching

Philip Blair(✉)[0000−0002−0074−9914], Carmel Eliav[0000−0001−9098−7387], Fiona Hasanaj[0000−0003−0670−436X], and Kfir Bar[0000−0002−1354−2955]

Basis Technology, 1060 Broadway, Somerville MA 02144, USA
{pblair,carmel,fiona,kfir}@basistech.com
https://www.basistech.com

Abstract. Automatic co-text free name matching has a variety of important real-world applications, ranging from fiscal compliance to border control. Name matching systems use a variety of engines to compare two names for similarity, with one of the most critical being phonetic name similarity. In this work, we re-frame existing work on neural sequence-to-sequence transliteration such that it can be applied to name matching. Subsequently, for performance reasons, we then build upon this work to utilize an alternative, non-recurrent neural encoder module. This ultimately yields a model which is 63% faster while still maintaining a 16% improvement in averaged precision over our baseline model.

Keywords: Name matching · Transliteration · Natural Language Processing · Sequence-to-Sequence · Multilingual · Performance.

1 Introduction

Names are an integral part of human life. From people to organizations and beyond, understanding what things are called is a critical aspect of natural language processing. A significant challenge in many applications is the fact that systems of appellation vary widely across cultures and languages, meaning that it can be challenging for a human, let alone a machine, to determine that two names are equivalent. Automatic evaluation of pairs of names has many important real-world applications, ranging from border control to financial know-your-customer ("KYC") compliance, which both involve searching for a name inside of large databases.

Computerized name matching systems attempt to accomplish this through a variety of statistical measurements that compare different properties of the given names. These systems are built with the goal of assigning a score to a pair of names that reflects the likelihood that those two names are "equivalent." For example, a name matching system should assign a high score to the input pair "Nick" and "Nicholas". Similarly, it ideally is able to competently handle a variety of other name-specific phenomena, such as missing components (e.g. "Franklin D. Roosevelt" and "Franklin Roosevelt"), initialisms (e.g. "J. J. Smith" and "John Joseph Smith"), transliteration variations (e.g. "Abdul Rasheed" and "Abd

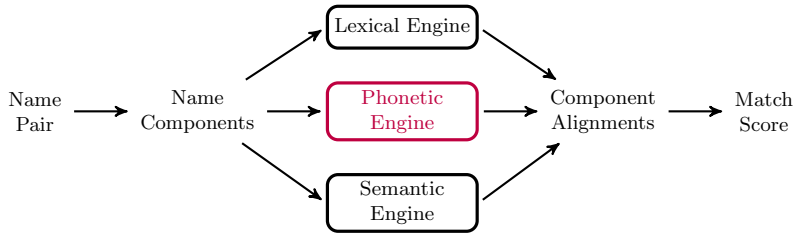


Fig. 1: Architecture of an enterprise name matching system.

al-Rashid"), different scripts (e.g. "Caesar" and "シーザー" (*Shīzā*)), and so on. In principle, this is accomplished via partitioning a full name into components (i.e. turning "John Smith" into ["John", "Smith"]) and comparing the components of each name via a variety of engines. Using those comparisons, the components are aligned appropriately and combined into a single name match score.

One engine that is particularly important is phonetic similarity, which measures how close the pronunciation of the names are to one another. For example, a phonetic name engine would assign a low similarity to the input pair ("John", "J."), but it would assign a high similarity to the input pair ("Chris", "クリス" (*kurisu*)). This could be powered by a number of different technologies, ranging from Soundex-based [35] indexing to statistical modeling techniques. In this paper, we focus on this phonetic-based engine and how it can be improved with neural techniques.

1.1 Challenges

When dealing with name matching across different languages, there is often a difference in writing scripts, which presents unique challenges. For some writing systems this is not particularly difficult, whereas with others (such as with the Latin alphabet and Japanese syllabaries), it is more challenging. This is because there is not a one-to-one correspondence in the characters used for each alphabet, meaning that it can be difficult to transliterate Latin alphabet-based languages into Japanese. For example, the word "photo" would be transliterated as "フォト" (*foto*). Not only is this transliteration two characters shorter than the English word, but the interactions between different parts of the English word inform the transliteration in a non-trivial fashion. Specifically, the small "ォ" (*o*) character can only be used when forming a digraph with another character (in this case, the "フ" (*fu*) character).

Thus, a statistical name matcher must take into account a nontrivial scope of contextual information when doing name matching. In our work, we explore the relationship between name matching and the related task of name *transliteration*, which seeks to produce the corresponding transliteration of a name in one language, given that name in another language.

Moreover, as our objective is to deploy a system in a production environment, speed becomes a concern. While recurrent neural networks are extremely powerful

tools for sequence modeling, they incur a significant amount of overhead in comparison to non-neural based techniques. To combat this, we explore an alternative, non-recurrent architecture in search of a model which balances the improved modeling capacity of neural networks with the superior speed of non-neural graphical models.

To summarize, our contributions in this work are the following: (1) we utilize prior work on neural transliteration in order to perform name matching, (2) we address performance issues surrounding the deployment of neural network-based name matching systems in an industrial context, and (3) we do so by suggesting an alternative neural architecture for name transliteration.

2 Related Work

Approaches to matching names across languages using non-neural machine learning techniques have a long history in literature. A variety of existing work opts for a cost-based approach to the problem [1, 2, 25, 31], which computes various similarity metrics between a given pair of names. For example, when doing monolingual name matching, thresholding a simple Levenshtein distance [21] may be a sufficient approach. While much of this cost-based work has focused around languages with large amounts of written variation, such as Arabic [1, 2], only a smaller amount of it has focused on cross-orthographic name matching [25]. The challenge with applying cost-based algorithms to cross-script name matching is that it often relies, at some level, on normalization techniques such as romanization or Soundex-based [35] indexing, which can introduce noise into the name matching process.

A closely related problem to name matching in NLP research is that of entity linking. The primary differentiating aspect in this scenario is the presence of co-text; that is, the problem is that of linking an entity mentioned in a larger piece of text to an entity present in some larger knowledge base. While the scope of our research could certainly enhance real-world applications of these systems in order to grapple with representations of entities in unseen scripts, research in this area [19, 24, 41] typically (implicitly or otherwise) assumes that the entity in the document and the knowledge base have been written in the same script. This is why the task is sometimes referred to as "named-entity disambiguation," as the focus tends to be more centered around leveraging co-text in order to disambiguate amongst a list of candidate entities.

Name matching is a subset of the broader problems of entity matching and record linkage. Neural networks have been applied to this area [11, 30, 23], but this prior work does not specifically focus on the name matching problem. Research in this area often emphasizes the structured nature of records (e.g. attempting to correlate many distinct schemas together) so a direct comparison to this work is made difficult. Further complicating the issue is the fact that record linkage research is, to the authors' knowledge, always an end-to-end process which operates on *full* names, not specific name components. This work focuses on the integration of neural-based techniques into an *existing*, larger enterprise

name matching system, so a meaningful direct comparison to the performance of our specific sub-engines is challenging if not impossible.

Direct statistical modeling of name matching has been published as well. Most similar to what we discuss in our baseline system outlined Section 3 is Nabende et al.'s work [27] on using Pair-HMMs [10] to model cross-lingual name matching and transliteration. Moreover, there is a body of work that has applied deep neural networks to the task of name matching [20, 43]; however, these do so by directly training a discriminative model that classifies names as matching or not. In contrast, our work is based on solving this problem by modeling the corresponding generative distribution, which can yield benefits ranging from lower data requirements [29] to lower amounts of supervision (we need only collect pairs of known transliterations, without needing to collect known non-transliterations).

As discussed in Section 3, modeling name transliteration is an important step in our system's process, so it is apt to review prior art in this domain as well. Neural name transliteration based on sequence-to-sequence models has been described in a limited amount of prior work [14]. Furthermore, there has been further research on neural techniques for transliteration in general [34, 38, 26]. Beyond this, there have been a variety of non-neural approaches to transliteration, ranging from systems based on conditional random fields [9] to local classification of grapheme clusters [33, 36], statistical machine translation techniques [32, 6, 40], and modeling transliteration as a mixture distribution [22].

In this work, we take this existing work on transliteration and re-frame it such that it can be applied to name matching. We then, for performance reasons, build upon this work to utilize an alternative, non-recurrent neural encoder module.

3 Phonetic Name Matching Systems

In this work, we explore the capabilities of an enterprise statistical phonetic name matching engine, with a focus on matching names across English and Japanese. More specifically, we will be taking names written in the Standard Latin script [16] and comparing equivalent names written in the Japanese Katakana script, which is typically used for foreign names. This is an interesting problem, as we must deal with different scripts that are not one-to-one with each other (for example, the two Latin letters "na" correspond to the single Katakana character "ナ"). Our goal is to have a system that can assign a high score to a pair of names that are pronounced similarly, and a low score to name pairs that are not. One idea for English-Japanese name matching is to develop a probabilistic model describing the likelihood of one name being a transliteration of another; this probability would then directly reflect the same semantics that we are trying to capture with the aforementioned score. More concretely, for a given English name n_{en} and Japanese name n_{ja}, this can be modeled probabilistically as $P_{ja-en}(n_{en}|n_{ja})$ (or vice versa, via an appropriate P_{en-ja} model). Per the chain rule, one approach to learning this conditional distribution is by modeling the following fully-generative distribution:

$$P(n_{en}, n_{ja}) = P_{ja-en}(n_{en}|n_{ja})P(n_{ja}) = P_{en-ja}(n_{ja}|n_{en})P(n_{en}). \quad (1)$$

Semantically, this has an interesting interpretation: If we can fit a generative model that allows us to transliterate (phonetically) a name from one Japanese to English, we are then able to use this to determine the quality of a given potential transliteration. In other words, by probabilistically modeling the task of name transliteration, we are able to directly model name matching by reading these probabilities.

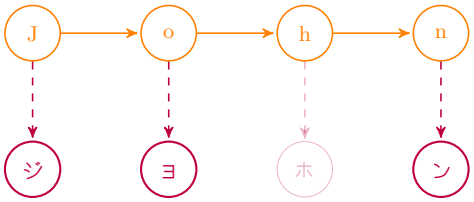


Fig. 2: A Hidden Markov Model-based phonetic name matching system.

For our baseline system, we make use of a Hidden Markov Model [4], as this directly models this distribution. Our model operates on sequences of characters, with the character sequence in one language being modeled as the hidden state sequence and the character sequence in the other language being modeled as emissions. A simplified version of this process is shown in Figure 2. In order to compensate for scripts that are not one-to-one (such as Latin and Katakana), we extend the character alphabet to contain a closed set of digraphs (such as "na", shown above), which are discovered automatically via Expectation Maximization (EM) [8]. For our neural model, we would like to model name matching in a largely similar fashion to our HMM-based technique. To this end, we approach the problem by first developing a neural network that models the process of English-Japanese name transliteration and then use the probability distributions computed by this model to facilitate name matching. Our name transliteration model was inspired by work done on neural machine translation [42]. In this setup, we utilize a sequence-to-sequence architecture [37], translating from a "source" domain of English name character sequences to a "target" domain of the corresponding character sequences for the Japanese transliteration. We explore two variants of this architecture: In the first, seq2seq-LSTM, the encoder module of our model is implemented using a Long Short-Term Memory (LSTM)-based [15], while the second, seq2seq-CNN, uses an encoder based on a CNN [12, 7]. For both of these architectures, we use an LSTM-based decoder module.

3.1 Neural Name Transliteration

Our first approach to neural name transliteration was directly based on the architecture used in Sutskever et al.'s work [37]. We used a bidirectional LSTM encoder module, with the output being fed into an LSTM-based decoder module, augmented with a basic attention mechanism [3], to produce the Japanese

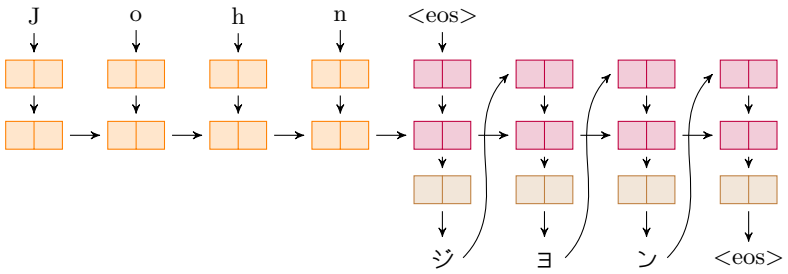


Fig. 3: A seq2seq-based name transliteration system with LSTM-based encoder and decoder (unidirectional encoder shown).

transliteration. This architecture, illustrated in Figure 3, is referred to as seq2seq-LSTM. This architecture is effectively the same as used for general transliteration by Rosca and Breuel [34].

As discussed in Section 4, we observed this architecture's throughput to be too low for our use case. We hypothesized that this slowdown was largely due to two aspects of the seq2seq-LSTM network: (1) the expense of computing the attention step at each point of the decoding process and (2) the lack of parallelizability of the recurrent architecture used in the encoder and decoder modules. When developing our second model, known as seq2seq-CNN, we addressed the former by simply eschewing the attention mechanism. Additionally, drawing from character encoding successes utilizing convolutional neural networks (CNNs) [12, 7] in other text applications [17, 5], this second architecture uses a CNN for the encoder module, based on work done by Gehring et al. [13].

As in Gehring et al.'s work, we utilize a series of CNN kernels that span the full character embedding dimension and process different-sized windows of characters. In contrast to their work, we run the channels through a max-pooling layer, providing us with an efficiently-computable, fixed-size representation of the full sequence (with one dimension for every kernel used). This representation is then passed to an LSTM decoder in the same way that the encoder representation was in the seq2seq-LSTM network.

3.2 Neural Name Matching

Running the network for name *matching* is very similar to running it for transliteration, but, instead of using the probability distribution produced by the decoder module at each time step to determine the next Japanese character to produce, we simply use the *known* Japanese character sequence to select it. As is traditionally done when working with language models, we then take the observed probabilities assigned to this known character sequence in order to compute the perplexity

$$PP(n_{ja}|n_{en}) = 2^{H(n_{en},n_{ja})}, \tag{2}$$

where

$$H(n_{en}, n_{ja}) = -\frac{1}{N}\sum_{i=1}^{N} \log P(n_{ja}|n_{en})[i],$$

$$P(n_{ja}|n_{en})[i] = P_{en-ja}(n_{ja,i}|n_{en}; n_{ja,1\ldots i-1}),$$

$n_{ja,k}$ is the kth character of n_{ja}, and N is the length of the target Japanese character sequence. We note that, in this mode of operation, we feed in the next character from the target sequence at each time step (i.e., we will keep going, even if the decoder would have predicted <eos> at a given time step). Now, conceptually, when comparing a name with a list of potential names, we would like to assign the highest score to the one that is *closest* to what the model predicts. Therefore, we define the reciprocal of the perplexity as the scoring function:

$$S(n_{en}, n_{ja}) = (PP(n_{ja}|n_{en}))^{-1}. \tag{3}$$

4 Experimental Results

Fig. 4: Illustration of the token alignment procedure used for data generation between "John Smith" and "ジョン・スミス" (*Jon Sumisu*). Note that the "・" character is treated as a special token, and ignored during the matching procedure.

4.1 Training and Hyperparameters

We train our system with a list of 33,215 component pairs; these were produced by collecting aligned English-Japanese full name pairs from Wikipedia, tokenizing them, and recording the aligned tokens in both scripts (throwing away items with non-Katakana names and pairs with differing numbers of tokens), as shown in Figure 4. This was possible due to the regularity of the dataset we collected and the nature of English-Japanese transliterations; however, to be safe, a bilingual Japanese-English speaker reviewed 6% of the data and confirmed that the process had produced properly-aligned word pairs.

This training set was divided into a 90% train and 10% validation split. All neural networks were trained with the Adam optimizer [18]. We trained the model for 100 epochs. The LSTM was trained with a single layer (for both the encoder and decoder), using a dropout rate of 0.5 on the decoder outputs, an embedding

size of 60 dimensions, and a hidden layer size of 100. Our CNN-based system utilizes six two-dimensional convolutional blocks with 100 output channels and kernel sizes of 2, 3, 4, 5, 6, and 7. The input embeddings (of dimension 60) are summed with trained position embeddings (up to a maximum length of 23, as seen in the training data) and passed through a dropout layer with a dropout rate of 0.25. These inputs are then passed in parallel to each of the convolutional blocks with ReLU activations[28], whose outputs are then max-pooled into six 100-dimensional vectors and concatenated. This is then passed to a linear layer to reduce it to a 200-dimensional vector, which is concatenated to each decoder input. The LSTM decoder is otherwise the same as the model using the LSTM-based encoder, except for the fact that it uses a hidden size of 100.

The primary purpose of this component-level engine is to exist as the keystone of our *full name* matching system. As such, we felt it would be most appropriate to measure its utility in that context for our evaluation. Rather than evaluating on another split-off piece of our word-level training data, we evaluate our models on a separate Wikipedia-sourced test set of 60,706 full name pairs. We measure Averaged Precision (AP) by calculating a match score between two full names using different underlying phonetic engines. Additionally, to simulate real-world usage in an information retrieval system, we use them to query a database of 14,941 names with a list of 500 names, and re-score the resulting name matches, measuring the Mean Averaged Precision (MAP), as is common practice.

4.2 Results

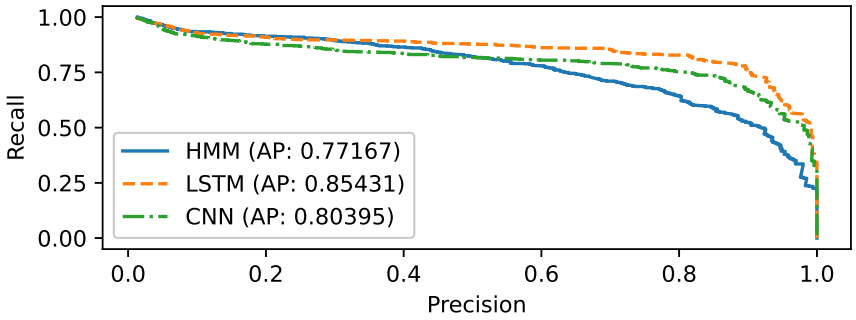


Fig. 5: Precision-Recall trade-off of different algorithms at various thresholds.

The models described in this work are now used to serve clients in real-life scenarios. Specifically, one client provided us with a list of 98 names that were incorrectly matched using the baseline HMM algorithm, so we began to explore neural techniques for name matching in order to reduce these false positives. When looking at this list of names, we determined that the mistakes were likely

Table 1: Selected examples of name pair scores using different phonetic engines.

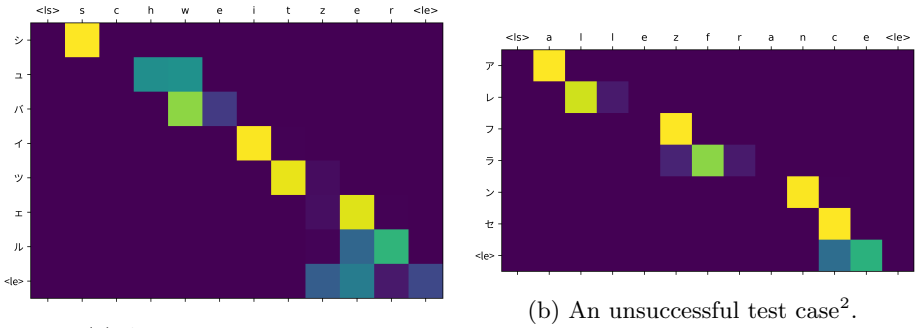
Name 1	**Name 2**	**HMM Based**	**LSTM Based**	**CNN Based**
Ada Lovelace	エイダ・ラヴレス (*Eida Raburesu*)	0.48	**0.74**	0.63
Albert Schweitzer	アルベルト・シュバイツエル (*Aruberuto Shubaitseru*)	0.69	**0.86**	0.81
Christopher Marlowe	クリストファー・マーロー (*Kurisutofā Mārō*)	0.76	**0.95**	0.83
Easy Goer[1]	イージーゴーアー (*Ījīgōā*)	0.39	0.66	**0.82**
James Whale	ジェームズ・ホエール (*Jēmuzu Hoēru*)	0.62	0.80	**0.88**
Alexandre Trauner	トラウネル・シャーンドル (*Torauneru Shāndoru*)	**0.84**	0.49	0.49
Allez France[1]	アレフランセ (*Arefuranse*)	**0.80**	0.46	0.46
U Nu	ウ・ヌー (*U Nū*)	**0.96**	0.93	0.68

caused by the HMM overly biasing towards frequencies in the training data, so an improvement would reflect an algorithm which better incorporates context. The first algorithm we tried was the seq2seq-LSTM architecture.

Table 2: Accuracy and speed results.

Matching Engine	**MAP**	**AP**	**Speed (sec/1000 tokens)**	**Slowdown**
HMM	63.61	77.17	**0.76**	**1x**
seq2seq-LSTM	**69.47**	**85.43**	16.3	21.26x
seq2seq-CNN	66.69	80.40	5.9	7.7x

Qualitatively, we found that the neural models are able to reduce false positives in our matching software on inputs that require more nuanced interpretation and composition of contextual information. This is empirically supported by the data in Figure 5, which shows a larger area under the precision-recall curve for our neural models than our baseline HMM model. A manual inspection of test cases, seen in Table 1, with large differences in scores from the baseline gives some insight into the strengths and weaknesses of the neural-based matcher. The



(a) A successful test case.

(b) An unsuccessful test case[2].

Fig. 6: Selected attention heatmaps from our LSTM-based name matcher architecture when measuring perplexity.

primary strength appears to be improved performance on particularly difficult transliterations. There are relatively few examples of the neural models doing markedly worse than the HMM-based model, but they broadly appear to fall under two categories. First, the neural engines seem to handle transliterated nicknames poorly (e.g. "Alexandre" and "シャーンドル" (*Shāndoru*); the latter being a transliteration of "Sandra"). Second, for extremely infrequent characters, such as ヌ (*nu*) (which appears in only around 1% of the training data), the neural models are less confident than the HMM. We hypothesize that this could also be compounded by the high amount of variance in how ヌ is transliterated (for example, the "ne" suffix on words such as "neptune" and "arsene" is often transliterated as ヌ), which could lead our model unable to learn a high-probability transliteration for this type of character. Finally, as a sanity-check, we inspect the attention heatmaps produced by the LSTM-based sequence-to-sequence models, shown in Figure 6. We can see that these indicate which sequences are not in line with what the model would expect to see: in subfigure (a), which shows an input pair that the model correctly assigns a high score to, we see that the primary activations align neatly with the corresponding Katakana characters in a manner similar to what one would intuitively expect. Conversely, in subfigure (b), which shows an input pair which was incorrectly assigned a low score, we see that the French phonology is the source of the confusion: at the position of the "フ" (*fu*) in the Katakana sequence, the attention is focused on the "z" in the Latin sequence. This indicates that the model is not expecting to treat the "z" as silent when performing the transliteration, meaning that it is surprised by the fact that the Katakana sequence is already on the "france" portion of the Latin name.

[1] These are the names of a racehorses.

[2] The original input, as shown in Table 1, has two English tokens, but the name matching framework attempts to concatenate them when evaluating whether they match the given Japanese token.

Quantitatively, as demonstrated in Table 2 and Figure 5, our initial seq2seq-LSTM architecture led to a dramatic improvement in accuracy. One will also notice the downside of such an approach: a roughly 21x speed degradation. While it is expected that a neural algorithm will incur some amount of additional performance overhead, the scale of data processed by our customers made this level of slowdown too great for us to accept. This is what led us to take a look at alternative seq2seq approaches, which are more efficiently computable thanks to improved parallelization. After developing the seq2seq-CNN network, we found that it lied directly in this sought-after "sweet spot" of having improved evaluation set performance over the HMM without the dramatic slowdown that the LSTM suffers from or sacrificing the qualitative improvements we observed with the seq2seq-LSTM model.

5 Conclusion and Future Work

Our work demonstrates that neural machine translation techniques can be applied to the problem of name matching, with notable success over a traditional graphical-based technique. Moreover, we show that modifying the encoder module of this neural network to use a convolutional neural network yields a significant speed improvement with only a moderate sacrifice in accuracy (while still outperforming the baseline algorithm).

There are a number of future steps we wish to explore with this work. The primary bottleneck remaining in our system's speed is the decoder. Due to the recurrent nature of our LSTM decoder, the perplexity of an input name component must be computed in $\mathcal{O}(n)$ time, character-by-character. We wish to remove this limitation by exploring alternative encoder architectures which predict the output sequence simultaneously, such as the work in [13]. Another natural extension of our experiments would be to additionally try a Transformer-based [39] encoder module, as one would expect this to yield speed improvements over the LSTM without the same degree of sacrifice in accuracy. Finally, this work explores how we can apply neural networks to a phonetic name matching engine, but we would like to explore what opportunities exist in other types of name matching engines.

References

1. Al-Hagree, S., Al-Sanabani, M., Alalayah, K.M., Hadwan, M.: Designing an accurate and efficient algorithm for matching Arabic names. In: 2019 First International Conference of Intelligent Computing and Engineering (ICOICE). pp. 1–12 (2019). https://doi.org/10.1109/ICOICE48418.2019.9035184
2. Al-Hagree, S., Al-Sanabani, M., Hadwan, M., Al-Hagery, M.A.: An improved n-gram distance for names matching. In: 2019 First International Conference of Intelligent Computing and Engineering (ICOICE). pp. 1–7 (2019). https://doi.org/10.1109/ICOICE48418.2019.9035154

3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1409.0473
4. Baum, L.E., Petrie, T.: Statistical inference for probabilistic functions of finite state Markov chains. Ann. Math. Statist. **37**(6), 1554–1563 (12 1966). https://doi.org/10.1214/aoms/1177699147, https://doi.org/10.1214/aoms/1177699147
5. Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., Glass, J.: What do neural machine translation models learn about morphology? In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 861–872. Association for Computational Linguistics, Vancouver, Canada (Jul 2017). https://doi.org/10.18653/v1/P17-1080, https://www.aclweb.org/anthology/P17-1080
6. Chen, Y., Skiena, S.: False-friend detection and entity matching via unsupervised transliteration. CoRR **abs/1611.06722** (2016), http://arxiv.org/abs/1611.06722
7. Cun, Y.L., Boser, B., Denker, J.S., Howard, R.E., Habbard, W., Jackel, L.D., Henderson, D.: Handwritten digit recognition with a back-propagation network. In: Advances in Neural Information Processing Systems 2. p. 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1990)
8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological) **39**(1), 1–38 (1977), http://www.jstor.org/stable/2984875
9. Dhore, M., Shantanu, K., Sonwalkar, T.: Hindi to English machine transliteration of named entities using conditional random fields. International Journal of Computer Applications **48** (07 2012). https://doi.org/10.5120/7522-0624
10. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.J.: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press (1998). https://doi.org/10.1017/CBO9780511790492, https://doi.org/10.1017/CBO9780511790492
11. Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., Tang, N.: Distributed representations of tuples for entity resolution. Proc. VLDB Endow. **11**(11), 1454–1467 (Jul 2018). https://doi.org/10.14778/3236187.3236198, https://doi.org/10.14778/3236187.3236198
12. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics **36**(4), 193–202 (apr 1980). https://doi.org/10.1007/bf00344251, https://doi.org/10.1007\%2Fbf00344251
13. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. p. 1243–1252. ICML'17, JMLR.org (2017)
14. Gong, J., Newman, B.: English-Chinese name machine transliteration using search and neural network models (2018)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (Nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735, https://doi.org/10.1162/neco.1997.9.8.1735
16. ISO: ISO Standard 646, 7-Bit Coded Character Set for Information Processing Interchange. International Organization for Standardization, second edn. (1983), http://www.iso.ch/cate/d4777.html, also available as ECMA-6.

17. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-aware neural language models. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. p. 2741–2749. AAAI'16, AAAI Press (2016)
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
19. Kolitsas, N., Ganea, O.E., Hofmann, T.: End-to-end neural entity linking. In: Proceedings of the 22nd Conference on Computational Natural Language Learning. pp. 519–529. Association for Computational Linguistics, Brussels, Belgium (Oct 2018). https://doi.org/10.18653/v1/K18-1050, https://www.aclweb.org/anthology/K18-1050
20. Lee, C., Cheon, J., Kim, J., Kim, T., Kang, I.: Verification of transliteration pairs using distance LSTM-CNN with layer normalization. In: Annual Conference on Human and Language Technology. pp. 76–81. Human and Language Technology (2017)
21. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady **10**, 707 (Feb 1966)
22. Li, T., Zhao, T., Finch, A., Zhang, C.: A tightly-coupled unsupervised clustering and bilingual alignment model for transliteration. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 393–398. Association for Computational Linguistics, Sofia, Bulgaria (Aug 2013), https://www.aclweb.org/anthology/P13-2070
23. Li, Y., Li, J., Suhara, Y., Doan, A., Tan, W.C.: Deep entity matching with pre-trained language models. Proc. VLDB Endow. **14**(1), 50–60 (Sep 2020). https://doi.org/10.14778/3421424.3421431, https://doi.org/10.14778/3421424.3421431
24. Martins, P.H., Marinho, Z., Martins, A.F.T.: Joint learning of named entity recognition and entity linking. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop. pp. 190–196. Association for Computational Linguistics, Florence, Italy (Jul 2019). https://doi.org/10.18653/v1/P19-2026, https://www.aclweb.org/anthology/P19-2026
25. Medhat, D., Hassan, A., Salama, C.: A hybrid cross-language name matching technique using novel modified Levenshtein distance. In: 2015 Tenth International Conference on Computer Engineering Systems (ICCES). pp. 204–209 (2015). https://doi.org/10.1109/ICCES.2015.7393046
26. Merhav, Y., Ash, S.: Design challenges in named entity transliteration. In: Proceedings of the 27th International Conference on Computational Linguistics. pp. 630–640 (2018)
27. Nabende, P., Tiedemann, J., Nerbonne, J.: Pair hidden Markov model for named entity matching. In: Sobh, T. (ed.) Innovations and Advances in Computer Sciences and Engineering. pp. 497–502. Springer Netherlands, Dordrecht (2010)
28. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: ICML (2010)
29. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic. p. 841–848. NIPS'01, MIT Press, Cambridge, MA, USA (2001)

30. Nie, H., Han, X., He, B., Sun, L., Chen, B., Zhang, W., Wu, S., Kong, H.: Deep sequence-to-sequence entity matching for heterogeneous entity resolution. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. p. 629–638. CIKM '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3357384.3358018, https://doi.org/10.1145/3357384.3358018
31. Peled, O., Fire, M., Rokach, L., Elovici, Y.: Matching entities across online social networks. Neurocomputing **210**, 91 – 106 (2016). https://doi.org/https://doi.org/10.1016/j.neucom.2016.03.089, http://www.sciencedirect.com/science/article/pii/S0925231216306014, sI:Behavior Analysis In SN
32. Priyadarshani, H., Rajapaksha, M., Ranasinghe, M., Sarveswaran, K., Dias, G.: Statistical machine learning for transliteration: Transliterating names between Sinhala, Tamil and English. In: 2019 International Conference on Asian Language Processing (IALP). pp. 244–249 (2019). https://doi.org/10.1109/IALP48816.2019.9037651
33. Qu, W.: English-Chinese name transliteration by latent analogy. In: Proceedings of the 2013 International Conference on Computational and Information Sciences. p. 575–578. ICCIS '13, IEEE Computer Society, USA (2013). https://doi.org/10.1109/ICCIS.2013.159, https://doi.org/10.1109/ICCIS.2013.159
34. Rosca, M., Breuel, T.: Sequence-to-sequence neural network models for transliteration. arXiv preprint arXiv:1610.09565 (2016)
35. Russell, R.C.: Index (April 1918), US Patent 1,261,167
36. Sarkar, K., Chatterjee, S.: Bengali-to-English forward and backward machine transliteration using support vector machines. In: Mandal, J.K., Dutta, P., Mukhopadhyay, S. (eds.) Computational Intelligence, Communications, and Business Analytics. pp. 552–566. Springer Singapore, Singapore (2017)
37. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. p. 3104–3112. NIPS'14, MIT Press, Cambridge, MA, USA (2014)
38. Upadhyay, S., Kodner, J., Roth, D.: Bootstrapping transliteration with constrained discovery for low-resource languages. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 501–511. Association for Computational Linguistics, Brussels, Belgium (Oct-Nov 2018). https://doi.org/10.18653/v1/D18-1046, https://www.aclweb.org/anthology/D18-1046
39. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, u., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 6000–6010. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
40. Wang, D., Xu, J., Chen, Y., Zhang, Y.: Monolingual corpora based Japanese-Chinese translation extraction for kana names. Journal of Chinese Information Processing **29**(5), 11 (2015)
41. Wu, L., Petroni, F., Josifoski, M., Riedel, S., Zettlemoyer, L.: Scalable zero-shot entity linking with dense entity retrieval. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 6397–6407. Association for Computational Linguistics, Online (Nov 2020). https://doi.org/10.18653/v1/2020.emnlp-main.519, https://www.aclweb.org/anthology/2020.emnlp-main.519

42. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J.: Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR **abs/1609.08144** (2016), http://arxiv.org/abs/1609.08144
43. Yamani, Z., Nurmaini, S., Firdaus, R, M.N., Sari, W.K.: Author matching using string similarities and deep neural networks. In: Proceedings of the Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN 2019). pp. 474–479. Atlantis Press (2020). https://doi.org/https://doi.org/10.2991/aisr.k.200424.073, https://doi.org/10.2991/aisr.k.200424.073