A Comparison of Entity Matching Methods between English and Japanese Katakana

Michiharu Yamashita **Hideki Awashima** **Hidekazu Oiwa**[*]
Recruit Co., Ltd. / Megagon Labs
Tokyo, Japan
{chewgen, awashima}@r.recruit.co.jp, hidekazu.oiwa@gmail.com

Abstract

Japanese Katakana is one component of the Japanese writing system and is used to express English terms, loanwords, and onomatopoeia in Japanese characters based on the phonemes. The main purpose of this research is to find the best entity matching methods between English and Katakana. We built two research questions to clarify which types of entity matching systems works better than others. The first question is what transliteration should be used for conversion. We need to transliterate English or Katakana terms into the same form in order to compute the string similarity. We consider five conversions that transliterate English to Katakana directly, Katakana to English directly, English to Katakana via phoneme, Katakana to English via phoneme, and both English and Katakana to phoneme. The second question is what should be used for the similarity measure at entity matching. To investigate the problem, we choose six methods, which are Overlap Coefficient, Cosine, Jaccard, Jaro-Winkler, Levenshtein, and the similarity of the phoneme probability predicted by RNN. Our results show that 1) matching using phonemes and conversion of Katakana to English works better than other methods, and 2) the similarity of phonemes outperforms other methods while other similarity score is changed depending on data and models.

1 Introduction

Cleansing and preprocessing data is one of the essential tasks in data analysis such as natural language processing (Witten et al., 2016). In particular, finding the same entity from multiple datasets is a important task. For example, when the same entities are expressed by different languages, you need to convert them to the same writing format before entity matching.

The Japanese language has three kinds of character types, and they are used for different purposes (Nagata, 1998). One of the character types is Katakana, which is used to convert English words, foreign languages, and alphabet letters into Japanese characters (Martin, 2004). Katakana is often transliterated by phonemes unique to Japanese and that is similar but different from English pronunciation. In addition, whether terms are expressed in English or Katakana is dependent on sites. For example, on Japanese web pages, there are many restaurants written in English and Japanese even if they are the same stores such as “Wendy’s” and “ウェンディーズ”. If it is the same type of character, it is easier to identify the entity simply by calculating the similarity of the string, but in the case of different writing systems like English and Katakana, it is difficult to identify the entity.

In this research, we clarify the problem by exploring the following two research questions.

(1) What transliteration should be used for conversion?

In order to change the same string form, the following method can be considered.

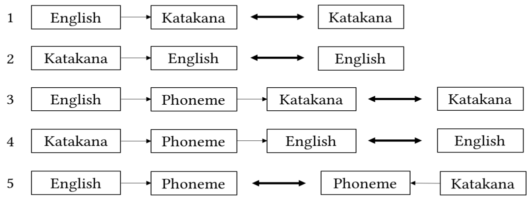

Figure 1: Method to Convert the Entity Name.

The first and second methods are to convert English to Katakana or Katakana to English and then match the entities.

The third and fourth methods are to use pronunciation information. Katakana is based on phonemes and is a syllable system, where each

[*] The author is now at Google Inc.

Proceedings of the 15th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 84–92
Brussels, Belgium, October 31, 2018. ©2018 The Special Interest Group on Computational Morphology and Phonology
https://doi.org/10.18653/v1/P17

syllabogram corresponds to one sound in the Japanese language. Therefore, the methods match the entities after converting English or Katakana into phonemes and converting the transliterated phonemes to Katakana or English.

The fifth method also uses phonemes. This method matches the entities based on the transliterated phoneme from both English and Katakana.

(2) What should be used for the similarity measure?

In order to calculate the similarity of a character string for entity matching, it is necessary to select measures from many similarity measures. In this research, as commonly used similarity measures, we use the similarity of Overlap Coefficient, Cosine, Jaccard, Jaro-Winkler, and Levenshtein (Cohen et al., 2003). Moreover, we propose a similarity method using the probability of the phonemes by prediction model. We clarify which of the six similarity methods should be used to compare the accuracy.

2 Related Work

Entity matching is a crucial task, and there is a lot of research on entity matching (Shen et al., 2015; Cai et al., 2013; Carmel et al., 2014; Mudgal et al., 2018). In these studies, the attribute information of an entity is used. In the case where there is no attribute and there is only the entity name, the character name information must be used. Different from general entity linking tasks, some works match entities only on entries in tables (Muñoz et al., 2014; Sekhavat et al., 2014). Although these studies match entities by collecting additional information on the entity, pronunciation information is not used.

In addition to studies of entity matching, transliteration is also studied. Transliteration is a task that converts a word in a language into a character of a different language and makes it as closely as possible to the native pronunciation. Many studies on transliteration are also conducted such as those on Hindi and Myanmar (Pandey and Roy, 2017; Thu et al., 2016). Some studies consider pronunciation information in transliteration (Yao and Zweig, 2015; Toshniwal and Livescu, 2016; Rao et al., 2015). Transliteration differs from entity matching itself in the purpose of the task, but it is applicable to entity matching because transliteration can extend the information of the entity. Therefore, we use transliteration to solve the task of entity matching.

There are some transliteration and entity matching studies, but there is little research that solves entity matching using transliteration information. Our motivation is to extend our database from external data by entity matching because we have relations of many types of clients such as restaurants, beauty salons, and companies and extension of data is essential for discovery of new clients. Therefore, we need to transform the name of the entities and to find which methods are the best for entity matching between English and Katakana.

3 Japanese Characters

Japanese characters are normally written in a combination of three character types. One type is ideographic characters, Kanji, from China, and the other two types, Hiragana and Katakana, are phonetic characters. Kanji is mainly used for nouns and stems of adjectives and verbs, and Hiragana is used for helpful markings of adjectives, verbs and Japanese words that are not expressed in Kanji. On the other hand, Katakana is used to write sound effects and transcribe foreign words (Martin, 2004). When we try entity matching with Japanese data, we usually face English expressed in Japanese Katakana in restaurants, companies, books, electrical items, and so on. We usually cannot find two names where one is written in English and the other is written in Katakana within enormous data because Japanese speakers use both English and Katakana to write foreign words.

Dictionaries already exist for English words with Japanese meanings, but few dictionaries also exist for English with Katakana. The report (Benson et al., 2009) mentions that the Japanese language is based on morae rather than syllables. A mora is a unit of sound that contributes to a syllable’ s weight. Katakana is more accurately described as a way to write the set of Japanese morae rather than the set of Japanese syllables, as each symbol represents not a syllable but a unit of sound of Japanese speech. A mora-based writing system in Japanese represents a dimension of the language that has no corresponding representation in English. This challenges the transliteration task of English and Katakana. Therefore, it is not easy to convert English into Katakana.

The Japanese language also has a method to transliterate Katakana into alphabet characters, and this transliterated alphabet is called Romaji, which is a phoneme of Japanese characters (Smith, 1996). Romaji is used in any context for non-Japanese speakers who cannot read Japanese characters, such as for names, passports, and any Japanese entities. Romaji is the most common way to input Japanese into computers and to display Japanese on devices that do not support Japanese characters (DeFrancis, 1984), and almost all Japanese people learn Romaji and are able to read and write Japanese using Romaji. Therefore, generally speaking, Japanese people who do not write in English usually use Romaji to express Katakana or foreign terms without Japanese characters.

4 Methods

To solve the task of entity matching between English and Katakana, the entity name must somehow be transliterated. Figure 2 shows the frame of the task. For example, the word “angel” has four features which are English, phonemes of English, Japanese Katakana and Romaji. We state how to convert the entity name, how to calculate the similarity, and what the baseline is below.

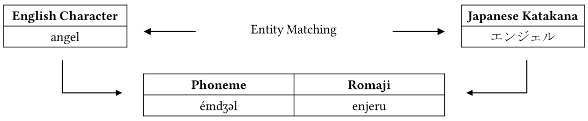

Figure 2: Entity Matching Task between English and Japanese Katakana.

4.1 Baseline

As the baseline, we used Romaji transliteration, which is characters used to transliterate Katakana to an alphabet sequence (DeFrancis, 1984). We converted Katakana to English using Romaji and then perform entity matching using both alphabets. There are many different popular ways of romanizing Katakana, and we use the Hepburn romanization because the romanization of a Katakana word generally faithfully represents the pronunciation of that Japanese word and many Japanese speakers use it to express Katakana and foreign terms without Japanese characteres. We used the module romkan[1] for this method. For example, in terms of“Japan”, the Romaji transliteration is “ジャパン” and the actual Katakana is also ”ジャパン”. As another example, in terms of “Orange” , the Romaji transliteration is “オランゲ” but the actual Katakana is ”オレンジ”. In addition, we also used Soundex[2] as a benchmark of entity matching between phoneme pairs. Soundex is a phonetic algorithm for indexing names by sound, as pronounced in English.

4.2 Predictive Model

We used a sequence to sequence model for transliteration. For example, the input is the sequence of English characters $(x_1, ..., x_n)$, and the output is the sequence of phoneme characters $(y_1, ..., y_m)$. In our model, we estimated the conditional probability p of an output sequence $(y_1, ..., y_m)$ given an input sequence $(x_1, ..., x_n)$ as follows:

$$p(y_1, ..., y_m | x_1, ..., x_n) \tag{1}$$

Given an input sequence $(x_1, ..., x_n)$, LSTM computes a sequence of hidden states $(h_1, ..., h_n)$. During decoding, it defines a distribution over the output sequence $(y_1, ..., y_m)$ given the input sequence $p(y_1, ..., y_m | x_1, ..., x_n)$ is:

$$p(y_1, ..., y_m | x_1, ..., x_n) = \prod_{t=1}^{m} p(y_t | h_{n+t-1}, y_{t-1}) \tag{2}$$

We also used a bi-directional recurrent neural network (Schuster and Paliwal, 1997). In our architecture, one RNN processes the input from left to right, while another processes it right to left. The outputs of the two subnetworks are then combined. This model has been applied to machine translation and, in this case, the phoneme prediction depends on the whole character sequence. Figure 3 shows an architecture of our models in this research. The combinations of input and output are English-Katakana, Katakana-English, English-Phoneme, Katakana-Phoneme, Phoneme-English, and Phoneme-Katakana. The input and the output of Figure 3 shows English-Phoneme as an example, and our other models were created in the same manner.

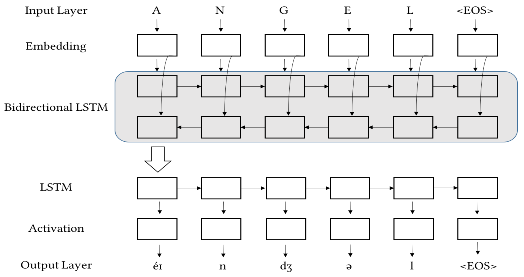

Figure 3: Architecture of the Predictive Model.

[1] https://pypi.python.org/pypi/romkan

[2] https://www.archives.gov/research/census/soundex.html

4.3 Model Settings

We chose each hyper parameter of the model by grid search. Targets of the hyper parameter were word embedding, hidden layers, and input reverse. Input of embedding and hidden layers was set to 256, 512, or 1024, and input of reverse is True or False. In the experiment, we chose the best model that has the lowest loss value for validation data and applied the model for validation data. We used categorical cross entropy as the loss value and the optimizer was Adam (Kingma and Ba, 2014). The hyper parameters used are shown in Table 1. Regarding the six models with RNN, we tried to match the entities of the transliterated word as shown in Figure 1.

At last, we created all word combinations with the dataset and then calculated their similarity score. Each model was trained by 80% of train dataset and 20% was used as test data. Test data was also used as an experiment for this research.

4.4 Similarity Metric

We implemented the module ”py_stringmatching”[3] that consists of a comprehensive and scalable set of string tokenizers such as alphabetical tokenizers, whitespace tokenizers, and string similarity measures. We used this module to calculate the string similarity. We chose Overlap Coefficient, Cosine, Jacquard, Jaro-Winkler, and Levenshtein from the modules. In addition to these similarities, we also proposed a new method that uses a probability of predicted phonemes from a term and compares each string similarity in the tasks of matching entity names. Definitions of each method are as follows.

Overlap coefficient measures the overlap between two sets, and is defined as the size of the intersection divided by the smaller of the size of the two sets. For two sets X and Y, the overlap coefficient is:

$$\frac{|X \cap Y|}{\min(|X|, |Y|)} \tag{3}$$

Cosine similarity measures the cosine of the angle between two non-zero vectors, and we use Ochiai coefficient as the angular cosine similarity and the normalized angle. This measure computes:

$$\frac{|X \cap Y|}{\sqrt{(|X||Y|)}} \tag{4}$$

Jaccard similarity measures the similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. For two sets X and Y, Jaccard similarity score is:

$$\frac{|X \cap Y|}{|X \cup Y|} \tag{5}$$

Jaro-Winkler similarity (Winkler, 1999) measures the edit distance between two sequences. It uses a prefix scale which gives weights to strings that match from the beginning for a set prefix length. For two sets X and Y, when s_i is the length of the string s_i, m is the number of matching characters, t is half the number of transpositions, and l is the length of the common prefix at the start of a string up to a maximum of four characters, Jaro-Winkler similarity sim_w is:

$$\begin{aligned} sim_w &= sim_j + (l \cdot 0.1(1 - sim_j)) \\ sim_j &= \tfrac{1}{3}\left(\tfrac{m}{|s_X|} + \tfrac{m}{|s_Y|} + \tfrac{m-t}{m}\right) \end{aligned} \tag{6}$$

Levenshtein distance measures the edit distance between two sequences. It computes the minimum cost to transform all edit procedures. Transforming a string is to delete a character, insert a character, and substitute one character for another.

The last similarity metric is the similarity of the probability of a predicted phoneme for a term, which we proposed. RNN model predicts phoneme probability, and we applied this metric for English-Phoneme and Katakana-Phoneme models. The output layer is the sequence, and the output of a predicted phoneme at the time t in our model are calculated as {”ei” :0.37, “e” :0.31, “æ” :0.23, $\cdots$} using the vector of the hidden layer of the decoder at time t. In this metric, we calculate the distance of dynamic time warping (Müller, 2007) between two temporal sequences. Dynamic time warping calculates an optimal match between two-time series sequences with certain restrictions. We regard the probability vector as a time point and use cosine similarity as a distance between two vectors. We computed all of the probability vectors from English and Katakana, and measured the distance of dynamic time warping.

[3] https://pypi.python.org/pypi/py_stringmatching

Table 1: Hyper Parameters of the Best Model (Acc is accuracy for validation data of train data.)

Input	Output	Embedding	Hidden Layers	Input Reverse	Acc
English	Katakana	512	512	False	0.765
Katakana	English	1024	512	True	0.785
English	Phoneme	1024	512	True	0.826
Katakana	Phoneme	512	512	True	0.811
Phoneme	English	512	512	True	0.815
Phoneme	Katakana	1024	512	False	0.764

5 Experiments

5.1 Dictionary Data

We prepared corpora of English, Katakana, and phonemes for training and creating models. We found a corpus of English and phonemes, and a corpus of English and Katakana. We created the data by merging each dictionary and made the original data ourselves as Japanese speakers.

At first, we used the dataset consisting of English, Katakana, and phonemes of English from Benson's dictionary[4], which is made from JMDict dictionary (Breen, 1995) and the sum of entries is 17798. However, in this dataset, there are a lot of mistakes because the creators are not native Japanese speakers, and they made it automatically. So, we removed the noise by hand as native Japanese speakers and then succeeded in cleansing the data of which 20% was occupied by noise.

Second, we used the CMU Pronouncing Dictionary[5] that includes a large amount of English terms and pronunciation signs but not Japanese Katakana. We combined them with Katakana by mecab-ipadic-NEologd[6]. mecab-ipadic-NEologd is a customized dictionary-system for MeCab (Kudo, 2006) that is an open-source text segmentation library for text written in the Japanese language, and some terms have both expression of English and Katakana. We merged the CMU Pronouncing Dictionary and mecab-ipadic-NEologd where both of the dictionaries had terms.

At last, we created an original dictionary by randomly extracting 3000 terms from the CMU Pronouncing Dictionary and attached each Katakana term by hand. We eventually concatenated those dictionaries and deleted duplicates. The total number of entries is 26815.

5.2 Experimental Methodology

For the experiment, we prepared three validation datasets. One was test data that comprised 20% of the dictionary. Another was city names in the U.S. from Google Maps API[7]. Google Maps provide a city name and we were able to find the same U.S. city expressed by English and Japanese. We collected 1110 terms from Google Maps. The last is the restaurant store names in Japan from HOT PEPPER GOURMET[8], which is called HPG. HPG is one of the most famous search services that provides information and discount coupons for restaurants, cafes, bars, and any place to eat in Japan. We know some restaurants that have both names in alphabet characters and Katakana from HPG, and built a validation dataset. Table 2 shows all of the data we used.

In terms of measuring accuracy, we used a top-five precision. We calculated the similarity scores of all combinations of entities in the experiment data and evaluated the precision of the entities included in the top-five entities, and then compared the value of top-five's precision for each model and similarity measure. The procedure of transliterating and measuring was as follows:

(1) En2Kana

We transliterated the sequence of alphabet characters into Katakana through RNN model and computed the similarity between the transliterated Katakana and Katakana terms.

(2) Kana2En

We transliterated the sequence of Katakana into English through RNN model and computed the similarity between the transliterated English and English terms.

(3) Both2Ph

We transliterated the sequence of alphabet characters into phonemes through RNN model, transliterated the sequence of Katakana into

[4] https://github.com/eob/englishjapanese-transliteration/blob/master/data/dictionary.txt

[5] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[6] https://github.com/neologd/mecab-ipadic-neologd

[7] https://developers.google.com/maps

[8] https://www.hotpepper-gourmet.com/en

Table 2: Dataset for Experiment

Dataset	Numbers of Entities	Data Source	Example
Train and Test Data	Total: 26815 Train: 21452 Test: 5363	Modified E. Benson’ s dictionary CMU Pronouncing Dictionary and mecab-ipadic-NEologd Our original dictionary	English: artist Katakana: アーティスト Phoneme: AA R T IX S T AX
City Names in the U.S.	1110	Google Maps	English: Phoenix Katakana: フェニックス
Restaurant Names	2458	HOT PEPPER GOURMET	English: ### Cafe Katakana: ###カフェ

phonemes through RNN model, and computed the similarity between both of the transliterated phonemes to regard one phoneme character as one index. In addition, we also used the probability vector similarity in terms of this method as we stated the end of subsection 4.4.

(4) En2Ph2Kana

We transliterated the sequence of alphabet characters into phonemes through RNN model, transliterated the transliterated phonemes into Katakana through RNN model, and computed the similarity between the transliterated Katakana and Katakana terms.

(5) Kana2Ph2En

We transliterated the sequence of Katakana into phonemes through RNN model, transliterated the transliterated phonemes into English through RNN model, and computed the similarity between the transliterated English and English terms.

(6) En2Romaji

We converted the sequence of alphabet characters into Katakana based on Romaji and computed the similarity between the Katakana of Romaji and Katakana terms.

(7) Kana2Romaji

We converted the sequence of Katakana into alphabets based on Romaji and computed the similarity between the alphabets of the Romaji and English terms.

(8)Both2Soundex

We transliterated the sequence of alphabet characters into Soundex as pronunciation, transliterated the sequence of Katakana based on Romaji into Soundex, and computed the score of both of the transliterated phonemes. We regarded one Soundex character as one index.

5.3 Results

Figure 4 shows the top-five precision graph of all methods and similarity metrics for each validation dataset, and Table 3 shows the top-five precision of predicted phoneme probability similarity.

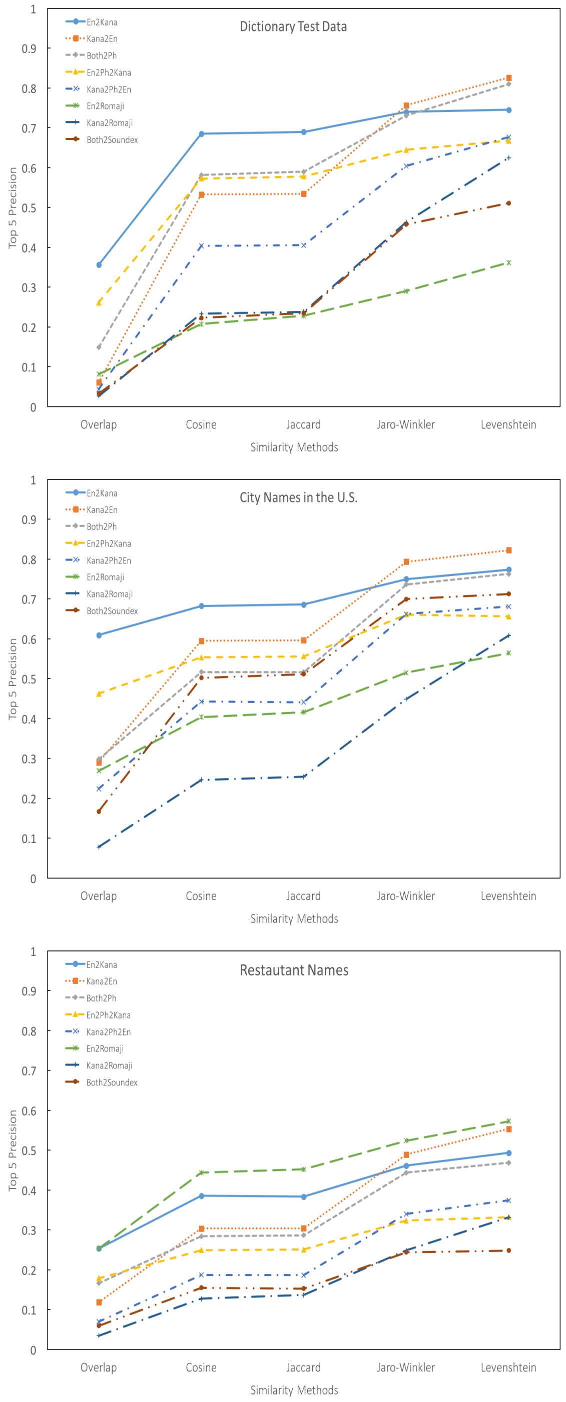

Figure 4: Top-Five Precision for Each Method and Similarity Measure.

First, we compared each of the eight methods with the similarity scores of Overlap coefficient, Cosine, Jacquard, Jaro-Winkler, and Levenshtein. The tendency of accuracy is similar in almost all methods in each dataset. In terms of the dictionary test data and the city name data, Kana2En and Both2Ph have the highest accuracy at 0.83 and 0.81 in Levenshtein distance. En2Kana was the next highest with an accuracy of 0.74 in Levenshtein. Regarding En2Ph2Kana, Kana2Ph2En, and Kana2Romaji, the accuracy was not high in any data. Likewise, even in the restaurant data, the accuracy of Kana2En and Both2Ph was relatively high at 0.55 and 0.46 in Levenshtein. However, regarding En2Romaji, it clarified that the accuracy was quite different depending on the dataset. In the dictionary test data, the accuracy was the lowest at 0.36 in Levenshtein distance, and in the city name data, the accuracy of En2Romaji is relatively low. On the other hand, in the restaurant data, En2Romaji was the highest accuracy at 0.57 in Levenshtein.

Second, focusing on the similarity measure, the accuracy trends are similar in almost all methods in each data set although the accuracy differs depending on the dataset. Levenshtein and Jaro-Winkler are the highest in almost all methods, and Overlap is the lowest. In other words, it shows that it is better to use the editing distance for word similarity in entity matching than other distance to compute the same character string.

Lastly, we considered the similarity of probability vectors. In fact, this similarity achieved the highest score. The test data of the dictionary and the data of city names' accuracy was over 0.9 and that is about 10% higher than the second highest score. The restaurant data accuracy was 0.62, which is 5% higher than the second highest score. These results insist that pronunciation is crucial for entity matching.

Table 3: Top 5 Precision of similarity of predicted Phoneme Probability Vector

Test Data of Dict	City Names	Restaurant Names
0.91	0.92	0.62

5.4 Error Analysis and Discussion

Regarding the dictionary test data and the city name data, the accuracy was over 80%, whereas the accuracy of the restaurant data was under 70%. Focusing on this phenomena, we analyzed how words were converted in each dataset. Table 4 shows some examples of each conversion.

While short words such as “switch” and “Benton” succeeded in being transformed cleanly, long words such as “coconut milk” and “South San Francisco” could not be predicted accurately. This is caused by the lack of long words in the training data. As the solution, it would be beneficial to adjust the algorithm to long terms and extend the dataset. However, in the case of Japanese, it is difficult to divide a long word into two or more words, because in comparison to English terms, Japanese terms are not separated by spaces.

Furthermore, in the restaurant data, we mined the reason why the accuracy was extremely low and found three considerable reasons in addition to the reason of word length. One reason is that there were many alphabetical representations of foreign languages other than English. For example, “Amore” is Italian, and “MAI-THAI” is Thai. Since our training data consisted only of pure English words, and the model was created for English, we can treat pure English terms as dictionary data or American city name data, but terms of other foreign languages cannot be converted accurately. To solve the problem, it is essential to create a model besides English for each language, and to create a model to recognize what language is written in words. The second is that there are some shops written in Romaji characters such as “AKICHI”, which is not English but the Romaji representation of Katakana. Therefore, there should also be a model to determine whether the word is a foreign word or Romaji. Third is the mistakes of the datasets. In this study, we extracted both alphabetical and Katakana stores automatically from HPG database, but there was a pattern in which English and Katakana combinations did not correspond completely. One of them is an abbreviation such as “Cafe X” and “X”. Prefixes are sometimes omitted in the data.

Considering similarities, the predicted phoneme probability had the highest score. Katakana is segmental scripts, and each term is based on sounds, but some exceptions are changed somehow by implicit Japanese rules. Therefore, we can not predict the pronunciation perfectly. However, because we can predict candidates of the pronunciation as phoneme probability vector like {"ei":0.37, “e”:0.31, “æ”:0.23, ···}, we could match enti-

Dataset	English	Katakana	En2Ph	Kana2Ph	En2Kana	Kana2En	En2Ph2Kana	Kana2Ph2En	En2Romaji	Kana2Romaji
Dictionary Test Data	switch	スイッチ	s w ih ch	s w ih ch	スウィッチ	swicch	スイッチ	switch	sウィtch	suicchi
	chicken	チキン	ch ih k ah n	ch ih k ah n	チェッンン	chicken	チカン	chickon	チcケン	chikin
	coconut milk	ココナッツミルク	k ax k y n t t t ih ih k ax	k ax k ax k ax n t m m ih ax ax	ココトノミトックク	coconantiiill	コャントトトトッッ	cococononmmam	ココヌtミlk	kokonattsumiruku
	airport	エアポート	eh r p ao r t	eh r p ow r t	エアポート	airpott	エアポート	airooort	アイrポrt	eapoto
City Names in the U.S.	Phoenix	フェニックス	f iy n ih k s	f eh n ih k s	フェニニクク	phenixs	フェニックス	phnix	pホエニx	fenikkusu
	Benton	ベントン	b eh n t ax n	b eh n t ah n	ベントン	benton	ベントン	benton	ベントン	benton
	Mountain View	マウンテン・ビュー	m aw n t t n n y uw	m aw n t ix n v y uw	マウンンンビイビー	mouttinnbvve	マウトトンニュー	mouttinveww	モウンタインヴィエw	mauntenbyu
	San Jose	サンノゼ	s ae n jh ow z	s ae n n ow z	サンジェス	sannoss	サンジジーズ	sannose	サンジョセ	sannoze
	South San Francisco	サウス・サンフランシスコ	s ow f f ae n s s s s k k k ax	s er s s ax s s ae n s ax k r s s	サススラランンンニカス	soussccnnnnssccoo	フフファスンススククク	ssrssasaacancucces	ソウthサンfランcイscオ	saususanfuransisko
Restaurant Names	Adesso	アデッソ	ax d eh s ow	ae d eh s ow	アドッソ	adesso	アデッ	adeso	アデッソ	adesso
	Amore	アモーレ	ah m ao r	ah m ow r ey	アモー	amorel	アモー	amora	アモレ	amore
	MAI-THAI	マイタイ	m ey th ey	m ay t ay	メイア	mitite	メイイ	mithte	マイ-テャイ	maitai
	Ristorante	リストランテ	r ih s t ow t t	r ih s t r ax n t eh hh eh r ow	リストラインアア	resteraatttttooo	リトトトシシシアア	resternaaarr	リストランテ	risutorante
	AKICHI	アキチ	ae k iy ch iy	ae k ch iy	アキチ	aciih	アキチー	acchi	アキチ	akichi

Table 4: Example Results of Transliteration.

ties using vector similarity more accurately. This is why the probability vector metric outperforms all other metrics. We also computed the other top-N precision, and the result was precision (N=10) at 67% and precision (N=30) at 74%. This opens the way for narrowing down by entity matching with the name alone.

For future work, in order to improve accuracy on restaurant data, we could implement a few procedures. The first would be cleaning the datasets because there are some pairs that do not correspond. This time we checked the non-matching pairs quickly but because of large volume of the datasets, there could remain some amount of noise. It could be solved by crowdsourcing of many Japanese speakers. The second would be polishing the transliteration model. We tried to use an attention mechanism for RNN model, but the accuracy was not good. We may polish the model by adjusting the best hyper parameter. Moreover, we could train pair data of English and Katakana at the same time instead of independently to create the model. In the entity matching task of the actual restaurant data, if we detect 60% through the top-five precision, entity matching would be easy through using additional data such as postal code level addresses.

6 CONCLUSION

In this paper we built two research questions to clarify how to solve heterogeneous entity matching. The first question was what should be used for conversion in entity matching between English and Japanese Katakana. We proposed an entity matching method that considers phoneme symbols, and compared models to convert a term into English, Katakana, and phonemes. The second question was what should be used for the similarity metric. We proposed a similarity method that uses a phoneme probability vector predicted by sequence to sequence models and compared six metrics, which are Overlap Coefficient, Cosine, Jacquard, Jaro-Winkler, Levenshtein, and the similarity of phonemes.

Our experiments in the three real datasets showed that 1) a phoneme matching system works better than other methods, and 2) the similarity of phoneme in the phoneme method and Levenshtein similarity in other methods suit the entity matching problems. We insist that phoneme information is crucial for heterogeneous entity matching.

Based on these results, we will build an entity matching system between English and Japanese Katakana, and publish a part of it as an open-source software. We can apply the system to extension of a dataset from a vast amount of data on the web and to rare query expansion on a search engine. Almost of our services have a search engine and in some services, we are using a part of the entity matching system to reduce outputs of 0 hits result. In the future, the model is expected to be polished and applied to other languages. In this paper it is tested on Japanese and English, but potentially could be used on other languages. We believe that our method could apply to any language that has phonetic characters.

Acknowledgments

In this research, we would like to thank Wang-Chiew Tan and Akiko Ito for crucial advice and supports of this work. We also thank the reviewers for valuable feedbacks.

References

Edward Benson, Stephen Pueblo, Fuming Shih, and Robert Berwick. 2009. English-japanese transliteration.

Jim Breen. 1995. Building an electronic japanese-english dictionary. In *Japanese Studies Association of Australia Conference*. Citeseer.

Zhiyuan Cai, Kaiqi Zhao, Kenny Q Zhu, and Haixun Wang. 2013. Wikification via link co-occurrence. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1087–1096. ACM.

David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June Paul Hsu, and Kuansan Wang. 2014. Erd'14: entity recognition and disambiguation challenge. In *ACM SIGIR Forum*, volume 48, pages 63–77. ACM.

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*, volume 3, pages 73–78.

John DeFrancis. 1984. Digraphia. *Word*, 35(1):59–66.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Taku Kudo. 2006. Mecab: Yet another part-of-speech and morphological analyzer. *http://mecab.sourceforge.net*.

Assunta Martin. 2004. The 'katakana effect' and teaching english in japan. *English Today*, 20(1):50–55.

Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34. ACM.

Meinard Müller. 2007. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84.

Emir Muñoz, Aidan Hogan, and Alessandra Mileo. 2014. Using linked data to mine rdf from wikipedia's tables. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 533–542. ACM.

Noriko Nagata. 1998. Input vs. output practice in educational software for second language acquisition. *Language Learning and Technology*, 1:23–40.

Pramod Pandey and Somnath Roy. 2017. A generative model of a pronunciation lexicon for hindi. *arXiv preprint arXiv:1705.02452*.

Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4225–4229. IEEE.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Yoones A Sekhavat, Francesco Di Paolo, Denilson Barbosa, and Paolo Merialdo. 2014. Knowledge base augmentation using tabular data. In *LDOW*.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Janet S Smith. 1996. Japanese writing. *The world' s writing systems*, pages 209–217.

Ye Kyaw Thu, Win Pa Pa, Yoshinori Sagisaka, and Naoto Iwahashi. 2016. Comparison of grapheme-to-phoneme conversion methods on a myanmar pronunciation dictionary. In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 11–22.

Shubham Toshniwal and Karen Livescu. 2016. Jointly learning to align and convert graphemes to phonemes with neural attention models. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 76–82. IEEE.

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

Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. In *INTERSPEECH 2015*, pages 3330–3334. ISCA.