# Phonetic Matching in Japanese

Michiko Yasukawa* J. Shane Culpepper† Falk Scholer†

*Gunma University, Gunma, Japan
michi@cs.gunma-u.ac.jp

†RMIT University, Melbourne, Australia
{shane.culpepper, falk.scholer}
@rmit.edu.au

## ABSTRACT

This paper introduces a set of Japanese phonetic matching functions for the open source relational database PostgreSQL. Phonetic matching allows a search system to locate approximate strings according to the sound of a term. This sort of approximate string matching is often referred to as fuzzy string matching in the open source community. This approach to string matching has been well studied in English and other European languages, and open source packages for these languages are readily available. To our knowledge, there is no such module for the Japanese language. In this paper, we present a set of string matching functions based on the phonetic similarity for modern Japanese. We have prototyped the proposed functions as an open source tool in PostgreSQL, and evaluated these functions using the test collection from the NTCIR-9 INTENT task. We report our findings based on the evaluation results.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*query formulation, retrieval models, search process*; I.7.1 [**Document and Text Processing**]: Document and Text Editing—*languages, spelling*; I.7.3 [**Document and Text Processing**]: Text Processing—*index generation*

## General Terms

Open Source RDBMS, Approximate String Matching, Fuzzy String Matching, Phonetic Matching, Japanese Information Retrieval

## 1. INTRODUCTION

One interesting but cumbersome problem in IR and NLP research is mismatches between the spelling of words. A simple question for this problem is: What if two words have the same meaning but different spellings? This is the issue explored in this paper. A related but more difficult problem are homographs, or words with the same spelling. The latter requires word sense disambiguation (WSD)[7], which is the task of identifying the meaning of words in a context. In this paper, we focus only on words with the same meaning and different spellings. These words are categorized into the following two types.

- *Synonyms* – words with the same (or, nearly the same) meaning, different spellings and different pronunciation.
- *Spelling variation* – words with the same meaning, different spellings and the same (or, nearly the same) pronunciation.

Synonyms are words that are similar in a semantic sense. Approaches such as Latent Semantic Indexing (LSI) [2] and word clustering[9] can be used to alleviate this problem. Spelling variants are slightly more difficult to classify, and present a difficult problem in ranked document retrieval systems depending on keyword queries.

Recent efforts to improve the effectiveness of IR systems have included web search result diversification [10]. In order to increase search effectiveness in this task, an innovative solution to spelling variants is needed. The goal of this paper is take a first step towards providing an open source tool to help with this problem in the Japanese language.

The rest of this paper is organized as follows: Section 2 presents string matching methods for English and Japanese; Section 3 describes our proposed phonetic matching approach for Japanese; Section 4 reports our findings based on a preliminary experimental study; and Section 5 presents conclusions and future work.

## 2. RELATED WORK

Phonetic matching is a type of approximate string matching. As with other approximate pattern matching methods, edit distance[6] and character-based $n$-gram search[8] are commonly used. The seminal approach to phonetic matching is the Soundex Indexing System[5]. It can accurately assign the same codeword to two different surnames that sound the same, but are spelled differently, like SMITH and SMYTH. The basic coding rules of Soundex are shown in Figure 1. Both SMITH and SMYTH are encoded as S530 by Soundex. In English and other languages, revised versions and alternatives of Soundex have been proposed. A set of these string matching functions, generally referred to as "fuzzy string matching", are deployed in an open source programming language. These functions are assembled in an open source relational database, PostgreSQL[1] as well and are applied to the objects in the database or combined with other relational operations.

[1]http://www.postgresql.org/


SIGIR 2012 Workshop on Open Source Information Retrieval.
August 16, 2012, Portland, Oregon, USA.

**Step-1** Retain the first symbol and drop all vowels.
**Step-2** Replace consonants with the following **code**.

| | | | | | |
|---|---|---|---|---|---|
| b, f, p, v | → | **1** | l | → | **4** |
| c, g, j, k, q, s, x, z | → | **2** | m, n | → | **5** |
| d, t | → | **3** | r | → | **6** |

**Step-3** Remove the duplication of code numbers.
**Step-4** Continue until you get three code numbers.
If you run out symbols, fill in **0**'s
until there are three code numbers.

**Figure 1: The Soundex Indexing System.**

For English, phonetic matching for IR systems has been studied extensively[1, 3, 11]. If Japanese documents are transliterated into a Latin alphabet, English methods can be applied. However, transliteration contains its inherent problems. Some input characters are lost due to the lack of correspondence between Japanese and Latin alphabets. How to reduce the transliteration errors is an interesting related problem, and will be considered in our future research. The issue of effective transliteration was explored by Karimi et al.[4]. Our research interest in this paper is to develop native matching functions for Japanese search engines without transliteration.

# 3. OUR METHODOLOGY

In this section, we present our approach to symbol grouping and phonetic matching in Japanese.

## 3.1 Japanese writing system and syllabary

In the Japanese language, the number of phonetic sounds is relatively small, and simply expressed in the $5 \times 10$ grid in Table 1. It shows the Japanese syllabary, "五十音(Gojūon)" meaning "Fifty Sounds" in English. In the table, both Hiragana symbols (the rounded syllabic symbols) and Katakana symbols (the angular syllabic symbols) are displayed. Vowel symbols, such as ア (a) and イ (i), do not have corresponding consonants, and this is represented as $\phi$ in the leftmost column. In the table, the gray cells are vacant because the symbols become lost over time. In modern Japanese, there is no symbol for Y+I (yi), Y+E (ye), W+U (wu), and the pronunciation for them are the same as $\phi$+I (i), $\phi$+E (e), $\phi$+U (u), respectively. The symbols for W+I (wi) and W+E (we) are outdated, but can still be used in modern Japanese. They are normally replaced with symbols for $\phi$+I (i) and $\phi$+E (e) because of the similarity of the pronunciation. In addition to the symbols in the table, the Japanese writing system takes an additional symbol, ン (n, a syllabic nasal), symbols with a voiced/semi-voiced sound mark, such as ガ (ga) or パ (pa), lower-case symbols, such as ッ (tsu, a double consonant), or ャ (ya, a contracted sound), and a diacritical mark for a prolonged sound, such as ー (a macron). For web queries and documents, classical spellings, such as the usage of obsolete symbols, ヱ (we) or an uncommon usage of lower-case symbols, e.g, ヶ (ke) substituting for ケ (ke), may appear in web queries and documents for stylistic reasons, or simple mistakes. Japanese phonetic matching must account for such anomalies.

**Table 1: The Japanese Syllabary (Fifty Sounds).**

| | Hiragana Symbol | | | | | Katakana Symbol | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | I | U | E | O | A | I | U | E | O | |
| $\phi$ | あ | い | う | え | お | ア | イ | ウ | エ | オ | 1 |
| | a | i | u | e | o | a | i | u | e | o | |
| K | か | き | く | け | こ | カ | キ | ク | ケ | コ | 2 |
| | ka | ki | ku | ke | ko | ka | ki | ku | ke | ko | |
| S | さ | し | す | せ | そ | サ | シ | ス | セ | ソ | 3 |
| | sa | si | su | se | so | sa | si | su | se | so | |
| T | た | ち | つ | て | と | タ | チ | ツ | テ | ト | 4 |
| | ta | ti | tu | te | to | ta | ti | tu | te | to | |
| N | な | に | ぬ | ね | の | ナ | ニ | ヌ | ネ | ノ | 5 |
| | na | ni | nu | ne | no | na | ni | nu | ne | no | |
| H | は | ひ | ふ | へ | ほ | ハ | ヒ | フ | ヘ | ホ | 6 |
| | ha | hi | hu | he | ho | ha | hi | hu | he | ho | |
| M | ま | み | む | め | も | マ | ミ | ム | メ | モ | 7 |
| | ma | mi | mu | me | mo | ma | mi | mu | me | mo | |
| Y | や | | ゆ | | よ | ヤ | | ユ | | ヨ | 8 |
| | ya | | yu | | yo | ya | | yu | | yo | |
| R | ら | り | る | れ | ろ | ラ | リ | ル | レ | ロ | 9 |
| | ra | ri | ru | re | ro | ra | ri | ru | re | ro | |
| W | わ | ゐ | | ゑ | を | ワ | ヰ | | ヱ | ヲ | 10 |
| | wa | wi | | we | wo | wa | wi | | we | wo | |
| | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | |

## 3.2 Symbol groups in Japanese

Table 2 shows symbol groups for our phonetic matching functions. The symbol groups are assembled based on the similarity of Japanese speech sounds. Each symbol group is given a unique identifier (ID). In the table, text in parentheses expresses a commentary on each group and the corresponding consonant, e.g., K for カ (ka) and S for サ (sa). Vowel symbols, such as ア (a) and イ (i), do not have corresponding consonants, and hence $\phi$ is in the parentheses.

Table 2 composed of 3 distinct parts: "Fifty Sounds," "Voiced Sounds," and "Additional Sounds." As a whole, the table covers all speech sounds in modern Japanese that are writable with Katakana symbols in UTF-8 character encoding. Different from English phonetic matching that uses the Latin alphabet or Arabic numeral for input/output strings, our matching functions take Katakana symbols for input strings and use Hiragana symbols for output strings. In UTF-8 character encoding, Katakana symbols are from ァ (E382A1) to ヶ (E383B6), and the number of Katakana symbols is 86. Hiragana symbols are from ぁ (E38181) to ん (E38293), and the number of Hiragana symbols is 83. The three symbols, ヴ (E383B4), ヵ (E383B5), ヶ (E383B6) are special symbols. They are defined in the Katakana part only, and there is no corresponding Hiragana symbol for these symbols.

In Table 2, Katakana symbols for "Fifty Sounds" (F-01 to F-11) are mostly the same as those in Table 1 with the exception of Katakana symbols for *wi*, *we* and *wo* because the Katakana symbols, ヰ (wi), ヱ (we), and ヲ (wo) have the same pronunciation as イ (i), エ (e), and オ (o), respectively, in modern Japanese. Hence, F-02 are incorporated into the same group as F-01 in Table 2. Similarly, V-03 are separated from V-04, and incorporated into V-02 because the Katakana symbols, ヂ (di) and ヅ (du) in modern

**Table 2: The Symbol Groups for Japanese Phonetic Matching.**

| ID | Fifty Sounds [in] | Code [out] | ID | Voiced Sounds [in] | Code [out] | ID | Additional Sounds [in] | Code [out] |
|---|---|---|---|---|---|---|---|---|
| F-01 | アイウエオ ($\phi$) → | E38182 あ | | | | A-01 | ァィゥェォ (lower-case, $\phi$) → | E38182 あ |
| F-02 | ヰエヲ (obs., $\phi$) → | E38182 あ | | | | A-02 | ー (macron, $\phi$) → | E38182 あ |
| F-03 | カキクケコ (K) → | E3818B か | V-01 | ガギグゲゴ (G) → | E3818C が | A-03 | ヵヶ (lower-case, K) → | E3818B か |
| F-04 | サシスセソ (S) → | E38195 さ | V-02 | ザジズゼゾ (Z) → | E38196 ざ | | | |
| | | | V-03 | ヂヅ (obs., Z) → | E38196 ざ | | | |
| F-05 | タチツテト (T) → | E3819F た | V-04 | ダデド (D) → | E381A0 だ | A-04 | ッ (lower-case, T) → | E381A3 っ |
| F-06 | ナニヌネノ (N) → | E381AA な | | | | A-05 | ン (syllabic nasal, N) → | E38293 ん |
| F-07 | ハヒフヘホ (H) → | E381AF は | V-05 | バビブベボ (B) → | E381B0 ば | | | |
| | | | V-06 | ヴ (V) → | E381B0 ば | | | |
| | | | V-07 | パピプペポ (P) → | E381B1 ぱ | | | |
| F-08 | マミムメモ (M) → | E381BE ま | | | | | | |
| F-09 | ヤユヨ (Y) → | E38284 や | | | | A-06 | ャュョ (lower-case, Y) → | E38283 や |
| F-10 | ラリルレロ (R) → | E38289 ら | | | | | | |
| F-11 | ワ (W) → | E3828F わ | | | | A-07 | ヮ (lower-case, W) → | E3828F わ |

Japanese have the same pronunciation as ジ (zi) and ズ (zu), respectively.

Each Katakana symbol for "Voiced Sounds" (V-01 to V-07) is a symbol with a voiced/semi-voiced sound mark. In the table, voiceless and voiced (e.g., K and G), voiced and semi-voiced (e.g., B and P), and Japanese voiced and foreign voiced (e.g., B and V) are all distinguished and separated into different symbol groups. "Additional Sounds" (A-01 to A-07) cover the rest of Katakana symbols and the diacritical mark, ー (a macron).

## 3.3 Japanese Phonetic Matching

In the same way as the Soundex Coding System in English, the matching function in Japanese also encodes symbols according to symbol groups. Essential steps in the matching function are described as follows:

**Step-1** Encode all strings in text DB in advance.

**Step-2** Encode a query string on arrival.

**Step-3** Output a matching set of encoded symbols.

The greatest challenge in designing phonetic matching functions is deciding how to group similar phonetic symbols. Our approach accomplishes this challenge empirically rather than theoretically by using the actual speech sound in modern Japanese. Our first phonetic matching function (Japanese phonetic matching; jppm) is as follows:

**jppm1** Retain the first symbol, and encodes the rest as encoded symbols according to Table 2. The encoded symbols are a sequence of Hiragana symbols in UTF-8 character encoding. While it categorizes Japanese speech sounds in a rigorous manner, the output codewords tend to be too verbose, and consequently cause mismatches with similar strings.

In order to reduce the number of codewords used, we derive three revised versions from **jppm1**. They are altered from the initial version as follows:

**jppm2** In order to simplify output code symbols, drop all symbols if they are vowels (F-01 and F-02) or an "Additional Sound" (A-01 to A-07) in Table 2. The first symbol is always retained in this function.

**jppm3** In order to simplify encoded symbols, merge groups in the same line (the same row) in Table 2. To be more specific, incorporate V-01 into F-03, incorporate V-02 and V-03 into F-04, incorporate V-04 and A-04 into F-05, incorporate V-05, V-06 and V-07 into F-07, incorporate A-06 into F-09. The first symbol is always retained in this function.

**jppm4** In order to simplify output codewords, drop A-01, A-02, A-04 and A-06 in Table 2. The first symbol is still retained as is.

## 3.4 Implementation

Our aim is to provide an open source component of Japanese phonetic matching. We prototyped the phonetic matching functions in Japanese as an extended version of the `fuzzystrmatch` module in the open source relational database, PostgreSQL. We call the new module `jpfuzzystrmatch` and provide its source code under an open source license[2]. To help understanding, an example of the each of the 4 functions (jppm[1,2,3,4]) and an example of English phonetic encoding systems with a transliteration system are attached with the source code. The module contains user-defined C-Language functions, and is compiled into dynamically loadable objects (shared libraries). It is distinguished from PostgreSQL internal functions, and can be loaded by the server on demand.

# 4. EXPERIMENTS

Evaluation of phonetic matching is an important problem to consider. In contrast to ad hoc evaluation in IR systems, there are no standard test collections for phonetic matching functions. In our experiment, we adapted a standard test collection for IR systems, the NTCIR-9 Japanese Intent task, for evaluating phonetic matching functions in Japanese. To build a text database to query, 84 million index terms were extracted from 67 million Japanese documents in the `ClueWeb09-JA` collection. For document processing, we employed MeCab[3] as a morphological an-

[2]http://www.daisy.cs.gunma-u.ac.jp/jpfsm/

[3]http://mecab.sourceforge.net/

Table 3: Experimental Results (Average).

| | CodeLen | #Results | EditDist | $P_{score}$ |
|---|---|---|---|---|
| jppm1 | 8.2 | 6.0 | 1.88 | 3.06 |
| jppm2 | 5.3 | 131.2 | 3.73 | 24.00 |
| jppm3 | 8.2 | 9.2 | 2.06 | 4.29 |
| jppm4 | 6.3 | 25.5 | 2.74 | 7.19 |

alyzer in Japanese, and Indri[4] as an indexing module to extract index terms from documents. In order to analyze how well the new functions work, sufficiently long Katakana words from topic words in the test collection were used as query strings. Specifically, we used all Katakana words which consists of 7 or more Katakana symbols; there were 10 such words in the test set.

Since the judgments in the test collection were not developed to carry out phonetic matching, how to evaluate the developed functions needs to be considered. Generally speaking, it is difficult for human assessors to judge if two strings are phonetically similar. For example, Zobel and Dart[11] report a high level of inconsistency among assessors in such a judgment proces. We therefore adopted an automatic evaluation approach.

In order to evaluate phonetic matching in an automatic manner, we make the following assumptions.

- If the discrepancy between two spellings is small, phonetic similarity is naturally large.
- If many matching strings are returned, the shortened sequence of symbols is sufficiently generic.

Based on the above assumptions, we define the following score, $P_{score}$ to measure the performance of phonetic matching functions.

$$P_{score} = \frac{\#Results}{1 + avg(EditDist)}$$

Here, $\#Results$ is the number of strings returned by a phonetic matching function, and $avg(EditDist)$ is the average edit distance between the query string and each returned string. Table 3 shows the experimental results. The average values are across all 10 test queries. In the table, **jppm2** has the highest $P_{score}$. However, manual inspection of the results showed that this function returned many strings that are phonetically similar, but are not spelling variants of the query strings. For example, **jppm2** obtained "matui ryousuke" (a male's name in Japanese) and "mattress queen" (a product name) for the query "matoriyosika" ("Matryoshka doll"). Since **jppm[1,3]** have different grouping features from **jppm2**, some obtained strings are different among these functions. The results of **jppm4** are a subset of those of **jppm2** because the grouping features are common. Putting together obtained strings, phonetic matching can be a viable solution to find potential spelling variants. However, the obtained strings still need to be filtered using further semantic analysis. In future work, we intend to evaluate these functions in the context of a full IR task.

[4]http://sourceforge.net/apps/trac/lemur/

# 5. CONCLUSION

In this paper, we have presented a set of new Japanese phonetic matching functions. We do not attempt to address the bigger issue of "word sense disambiguation" (WSD) and synonyms, but rather present a simple approach to capturing similar Japanese terms for ad hoc queries. A phonetic matching function takes an input sequence of symbols and converts it into a generic shortened sequence of symbols in order to find as many fuzzy matches as possible. Some of the generic codewords may gather too many matches, including correct ones (the same meaning) and wrong ones (different meaning). However, this approach can be used as a first pass filter to find potentially relevant documents in large Japanese document collections. This subset of documents can then be further refined using relevance feedback or more computationally expensive ranking metrics in subsequent retrieval steps. In future work, we intend to investigate the broader issue of WSD and synonyms using phonetic matching, and explore other applications of these simple matching functions.

# 6. ACKNOWLEDGMENTS

The first author was supported by JSPS KAKENHI Grant-in-Aid for Young Scientists (B) 21700273. The second author was supported by the Australian Research Council.

# 7. REFERENCES

[1] Amir, A., Efrat, A., and Srinivasan, S. Advances in phonetic word spotting. In *CIKM '01 Proceedings* (2001), ACM, pp. 580–582.

[2] Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. Indexing by latent semantic analysis. *JASIS 41*, 6 (1990), 391–407.

[3] French, J. C., Powell, L., and Schulman, E. Applications of approximate word matching in information retrieval. In *CIKM '97 Proceedings* (1997), ACM, pp. 9–15.

[4] Karimi, S., Scholer, F., and Turpin, A. Machine transliteration survey. *ACM Comput. Surv. 43*, 3 (2011), 17:1–17:46.

[5] Knuth, D. E. *The Art of Computer Programming: Volume 3*. Addison-Wesley, 1973.

[6] Navarro, G. A guided tour to approximate string matching. *ACM Comput. Surv. 33*, 1 (2001), 31–88.

[7] Navigli, R. Word sense disambiguation: A survey. *ACM Comput. Surv. 41*, 2 (2009), 10:1–10:69.

[8] Okazaki, N., and Tsujii, J. Simple and efficient algorithm for approximate dictionary matching. In *Coling Proceedings* (2010), pp. 851–859.

[9] Slonim, N., and Tishby, N. Document clustering using word clusters via the information bottleneck method. In *SIGIR '00 Proceedings* (2000), pp. 208–215.

[10] Song, R., Zhang, M., Sakai, T., Kato, M., Liu, Y., Sugimoto, M., Wang, Q., and Orii, N. Overview of the NTCIR-9 INTENT task. In *Proceedings of NTCIR-9 Workshop Meeting* (2011), pp. 82–105.

[11] Zobel, J., and Dart, P. W. Phonetic string matching: Lessons from information retrieval. In *SIGIR '96 Proceedings* (1996), pp. 166–172.