Bringing the Dictionary to the User: the FOKS system

Slaven Bilac[†], **Timothy Baldwin**[*] **and Hozumi Tanaka**[†]

† Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552 JAPAN
{sbilac,tanaka}@cl.cs.titech.ac.jp

∗ CSLI, Ventura Hall, Stanford University
Stanford, CA 94305-4115 USA
tbaldwin@csli.stanford.edu

Abstract

The dictionary look-up of unknown words is particularly difficult in Japanese due to the complicated writing system. We propose a system which allows learners of Japanese to look up words according to their expected, but not necessarily correct, reading. This is an improvement over previous systems which provide no handling of incorrect readings. In preprocessing, we calculate the possible readings each kanji character can take and different types of phonological and conjugational changes that can occur, and associate a probability with each. Using these probabilities and corpus-based frequencies we calculate a plausibility measure for each generated reading given a dictionary entry, based on the naive Bayes model. In response to a reading input, we calculate the plausibility of each dictionary entry corresponding to the reading and display a list of candidates for the user to choose from. We have implemented our system in a web-based environment and are currently evaluating its usefulness to learners of Japanese.

1 Introduction

Unknown words are a major bottleneck for learners of any language, due to the high overhead involved in looking them up in a dictionary. This is particularly true in non-alphabetic languages such as Japanese, as there is no easy way of looking up the component characters of new words. This research attempts to alleviate the dictionary look-up bottleneck by way of a comprehensive dictionary interface which allows Japanese learners to look up Japanese words in an efficient, robust manner. While the proposed method is directly transferable to other language pairs, for the purposes of this paper, we will focus exclusively on a Japanese–English dictionary interface.

The Japanese writing system consists of the three orthographies of hiragana, katakana and kanji, which appear intermingled in modern-day texts (NLI, 1986). The hiragana and katakana syllabaries, collectively referred to as kana, are relatively small (46 characters each), and each character takes a unique and mutually exclusive reading which can easily be memorized. Thus they do not present a major difficulty for the learner. Kanji characters (ideograms), on the other hand, present a much bigger obstacle. The high number of these characters (1,945 prescribed by the government for daily use, and up to 3,000 appearing in newspapers and formal publications) in itself presents a challenge, but the matter is further complicated by the fact that each character can and often does take on several different and frequently unrelated readings. The kanji 発, for example, has readings including *hatsu* and *ta(tsu)*, whereas 表 has readings including *omote*, *hyou* and *arawa(reru)*. Based on simple combinatorics, therefore, the kanji compound 発表 *happyou* "announcement" can take at least 6 basic readings, and when one considers phonological and conjugational variation, this number becomes much greater. Learners presented with the string 発表 for the first time will, therefore, have a possibly large number of potential readings (conditioned on the number of component character readings they know) to choose from. The problem is further complicated by the occurrence of character combinations which do not take on compositional readings. For example 風邪 *kaze* "common cold" is formed non-compositionally from 風 *kaze/fuu* "wind" and 邪 *yokoshima/ja* "evil".

With paper dictionaries, look-up typically occurs in two forms: (a) directly based on the reading of the entire word, or (b) indirectly via component kanji characters and an index of words involving those kanji. Clearly in the first case, the correct reading of the word must be known in order to look it up, which is often not the case. In the second case, the complicated radical and stroke count systems make the kanji look-up process cumbersome and time consuming.

With electronic dictionaries—both commercial and publicly available (e.g. EDICT (2000))—the options are expanded somewhat. In addition to reading- and kanji-based look-up, for electronic texts, simply copying and pasting the desired string into the dictionary look-up window gives us direct access to the word.[1]. Several reading-aid systems

[1] Although even here, life is complicated by Japanese being a non-segmenting language, putting the onus on the user to

(e.g. Reading Tutor (Kitamura and Kawamura, 2000) and Rikai[2]) provide greater assistance by segmenting longer texts and outputing individual translations for each segment (word). If the target text is available only in hard copy, it is possible to use kana-kanji conversion to manually input component kanji, assuming that at least one reading or lexical instantiation of those kanji is known by the user. Essentially, this amounts to individually inputting the readings of words the desired kanji appear in, and searching through the candidates returned by the kana-kanji conversion system. Again, this is complicated and time inefficient so the need for a more user-friendly dictionary look-up remains.

In this paper we describe the FOKS (Forgiving Online Kanji Search) system, that allows a learner to use his/her knowledge of kanji to the fullest extent in looking up unknown words according to their expected, but not necessarily correct, reading. Learners are exposed to certain kanji readings before others, and quickly develop a sense of the pervasiveness of different readings. We attempt to tap into this intuition, in predicting how Japanese learners will read an arbitrary kanji string based on the relative frequency of readings of the component kanji, and also the relative rates of application of phonological processes. An overall probability is attained for each candidate reading using the naive Bayes model over these component probabilities. Below, we describe how this is intended to mimic the cognitive ability of a learner, how the system interacts with a user and how it benefits a user.

The remainder of this paper is structured as follows. Section 2 describes the preprocessing steps of reading generation and ranking. Section 3 describes the actual system as is currently visible on the internet. Finally, Section 4 provides an analysis and evaluation of the system.

2 Data Preprocessing

2.1 Problem domain

Our system is intended to handle strings both in the form they appear in texts (as a combination of the three Japanese orthographies) and as they are read (with the reading expressed in hiragana). Given a reading input, the system needs to establish a relationship between the reading and one or more dictionary entries, and rate the plausibility of each entry being realized with the entered reading.

In a sense this problem is analogous to kana–kanji conversion (see, e.g., Ichimura et al. (2000) and Takahashi et al. (1996)), in that we seek to determine a ranked listing of kanji strings that could correspond to the input kana string. There is one major difference, however. Kana–kanji conversion systems are designed for native speakers of Japanese and as such expect accurate input. In cases when the correct or standardized reading is not available, kanji characters have to be converted one by one. This can be a painstaking process due to the large number of characters taking on identical readings, resulting in large lists of characters for the user to choose from.

Our system, on the other hand, does not assume 100% accurate knowledge of readings, but instead expects readings to be predictably derived from the source kanji. What we do assume is that the user is able to determine word boundaries, which is in reality a non-trivial task due to Japanese being non-segmenting (see Kurohashi et al. (1994) and Nagata (1994), among others, for details of automatic segmentation methods). In a sense, the problem of word segmentation is distinct from the dictionary look-up task, so we do not tackle it in this paper.

To be able to infer how kanji characters can be read, we first determine all possible readings a kanji character can take based on automatically-derived alignment data. Then, we machine learn phonological rules governing the formation of compound kanji strings. Given this information we are able to generate a set of readings for each dictionary entry that might be perceived as correct by a learner possessing some, potentially partial, knowledge of the character readings. Our generative method is analogous to that successfully applied by Knight and Graehl (1998) to the related problem of Japanese (back) transliteration.

2.2 Generating and grading readings

In order to generate a set of plausible readings we first extract all dictionary entries containing kanji, and for each entry perform the following steps:

1. Segment the kanji string into minimal morphophonemic units[3] and align each resulting unit with the corresponding reading. For this purpose, we modified the TF-IDF based method proposed by Baldwin and Tanaka (2000) to accept bootstrap data.
2. Perform conjugational, phonological and morphological analysis of each segment–reading pair and standardize the reading to canonical form (see Baldwin et al. (2002) for full details). In particular, we consider gemination (*onbin*) and sequential voicing (*rendaku*) as the most commonly-occurring phonological alternations in kanji compound formation (Tsujimura, 1996)[4]. The canonical reading for a given seg-

correctly identify word boundaries.

[2] `http://www.rikai.com`

[3] A unit is not limited to one character. For example, verbs and adjectives commonly have conjugating suffices that are treated as part of the same segment.

[4] In the previous example of 発表 *happyou* "announcement" the underlying reading of individual characters are *hatsu* and *hyou* respectively. When the compound is formed, *hatsu* seg-

ment is the basic reading to which conjugational and phonological processes apply.

3. Calculate the probability of a given segment being realized with each reading ($P(r|k)$), and of phonological ($P_{phon}(r)$) or conjugational ($P_{conj}(r)$) alternation occurring. The set of reading probabilities is specific to each (kanji) segment, whereas the phonological and conjugational probabilities are calculated based on the reading only. After obtaining the composite probabilities of all readings for a segment, we normalize them to sum to 1.

4. Create an exhaustive listing of reading candidates for each dictionary entry s and calculate the probability $P(r|s)$ for each reading, based on evidence from step 3 and the naive Bayes model (assuming independence between all parameters).

$$P(r|s) = P(r_{1..n}|k_{1..n}) \tag{1}$$

$$P(r_{1..n}|k_{1..n}) = \prod_{i=1}^{n} P(r_i|k_i) \times \\ \times P_{phon}(r_i) \times P_{conj}(r_i) \tag{2}$$

5. Calculate the corpus-based frequency $F(s)$ of each dictionary entry s in the corpus and then the string probability $P(s)$, according to equation (3). Notice that the term $\sum_i F(s_i)$ depends on the given corpus and is constant for all strings s.

$$P(s) = \frac{F(s)}{\sum_i F(s_i)} \tag{3}$$

6. Use Bayes rule to calculate the probability $P(s|r)$ of each resulting reading according to equation (4).

$$\frac{P(s|r)}{P(s)} = \frac{P(r|s)}{P(r)} \tag{4}$$

Here, as we are only interested in the relative score for each s given an input r, we can ignore $P(r)$ and the constant $\sum_i F(s_i)$. The final plausibility grade is thus estimated as in equation (5).

$$Grade(s|r) = P(r|s) \times F(s) \tag{5}$$

The resulting readings and their scores are stored in the system database to be queried as necessary. Note that the above processing is fully automated, a valuable quality when dealing with a volatile dictionary such as EDICT.

ment undergoes gemination and *hyou* segment undergoes sequential voicing resulting in *happyou* surface form reading.

3 System Description

The section above described the preprocessing steps necessary for our system. In this section we describe the actual implementation.

3.1 System overview

The base dictionary for our system is the publicly-available EDICT Japanese–English electronic dictionary.[5] We extracted all entries containing at least one kanji character and executed the steps described above for each. Corpus frequencies were calculated over the EDR Japanese corpus (EDR, 1995).

During the generation step we ran into problems with extremely large numbers of generated readings, particularly for strings containing large numbers of kanji. Therefore, to reduce the size of generated data, we only generated readings for entries with less than 5 kanji segments, and discarded any readings not satisfying $P(r|s) \geq 5 \times 10^{-5}$. Finally, to complete the set, we inserted correct readings for all dictionary entries s_{kana} that did not contain any kanji characters (for which no readings were generated above), with plausibility grade calculated by equation (6).[6]

$$Grade(s_{kana}|r) = F(s_{kana}) \tag{6}$$

This resulted in the following data:

Total entries: 97,399
Entries containing kanji: 82,961
Average number of segments: 2.30
Total readings: 2,646,137
Unique readings: 2,194,159
Average entries per reading: 1.21
Average readings per entry: 27.24
Maximum entries per reading: 112
Maximum readings per entry: 471

The above set is stored in a MySQL relational database and queried through a CGI script. Since the readings and scores are precalculated, there is no time overhead in response to a user query. Figure 1 depicts the system output for the query *atamajou*.[7]

The system is easily accessible through any Japanese language-enabled web browser. Currently we include only a Japanese–English dictionary but it would be a trivial task to add links to translations in alternative languages.

3.2 Search facility

The system supports two major search modes: **simple** and **intelligent**. **Simple** search emulates a conventional electronic dictionary search (see, e.g.,

[5] `http://www.csse.monash.edu.au/~jwb/edict.html`

[6] Here, $P(r|s_{kana})$ is assumed to be 1, as there is only one possible reading (i.e. r).

[7] This is a screen shot of the system as it is visible at `http://tanaka-www.titech.ac.jp/foks/`.

Figure 1: Example of system display

Breen (2000)) over the original dictionary, taking both kanji and kana as query strings and displaying the resulting entries with their reading and translation. It also supports wild character and specified character length searches. These functions enable lookup of novel kanji combinations as long as at least one kanji is known and can be input into the dictionary interface.

Intelligent search is over the set of generated readings. It accepts only kana query strings[8] and proceeds in two steps. Initially, the user is provided with a list of candidates corresponding to the query, displayed in descending order of the score calculated from equation (5). The user must then click on the appropriate entry to get the full translation. This search mode is what separates our system from existing electronic dictionaries.

3.3 Example search

Let us explain the benefit of the system to the Japanese learner through an example. Suppose the user is interested in looking up 頭上 *zujou* "overhead" in the dictionary but does not know the correct reading. Both 頭 "head" and 上 "over/above" are quite common characters but frequently realized with different readings, namely *atama, tou*, etc. and *ue, jou*, etc., respectively. As a result, the user could interpret the string 頭上 as being read as *atamajou* or *toujou* and query the system accordingly. Tables 1 and 2 show the results of these two queries.[9] Note that the displayed readings are always the correct readings for the corresponding Japanese dictionary entry, and not the reading in the original query. For all those entries where the actual reading coincides with the user input, the reading is displayed in boldface.

[8] In order to retain the functionality offered by the simple interface, we automatically default all queries containing kanji characters and/or wild characters into simple search.

[9] Readings here are given in romanized form, whereas they appear only in kana in the actual system interface. See Figure 1 for an example of real-life system output.

Entry	Reading	Grade	Translation
頭上	*zujou*	0.40844	overhead
頭声	*tousei*	0.00271168	head voice

Table 1: Results of search for *atamajou*

Entry	Reading	Grade	Translation
登場	***toujou***	73.2344	appearance
頭上	*zujou*	1.51498	overhead
搭乗	***toujou***	1.05935	embarkation
筒状	***toujou***	0.563065	cylindrical
道場	*doujou*	0.201829	dojo
東上	***toujou***	0.126941	going to Tokyo
凍傷	*shimoyake*	0.0296326	frostbite
凍傷	*toushou*	0.0296326	frostbite
刀匠	*toushou*	0.0144911	swordsmith
頭声	*tousei*	0.0100581	head voice
刀傷	*toushou*	0.00858729	sword wound
痘瘡	*tousou*	0.00341006	smallpox
凍瘡	*tousou*	0.0012154	frostbite
東清	*toushin*	0.000638839	Eastern China

Table 2: Results of search for *toujou*

From Table 1 we see that only two results are returned for *atamajou*, and that the highest ranking candidate corresponds to the desired string 頭上. Note that *atamajou* is not a valid word in Japanese, and that a conventional dictionary search would yield no results.

Things get somewhat more complicated for the reading *toujou*, as can be seen from Table 2. A total of 14 entries is returned, for four of which *toujou* is the correct reading (as indicated in bold). The string 頭上 is second in rank, scored higher than three entries for which *toujou* is the correct reading, due to the scoring procedure not considering whether the generated readings are correct or not.

For both of these inputs, a conventional system would not provide access to the desired translation without additional user effort, while the proposed system returns the desired entry as a first-pass candidate in both cases.

4 Analysis and Evaluation

To evaluate the proposed system, we first provide a short analysis of the reading set distribution and then describe results of a preliminary experiment on real-life data.

4.1 Reading set analysis

Since we create a large number of plausible readings, one potential problem is that a large number

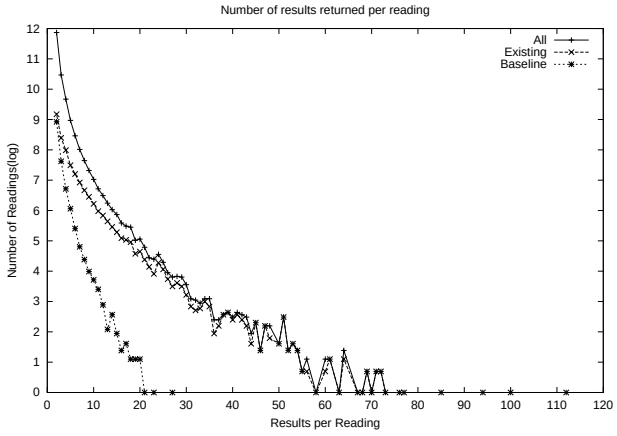

Figure 2: Distribution of results returned per reading

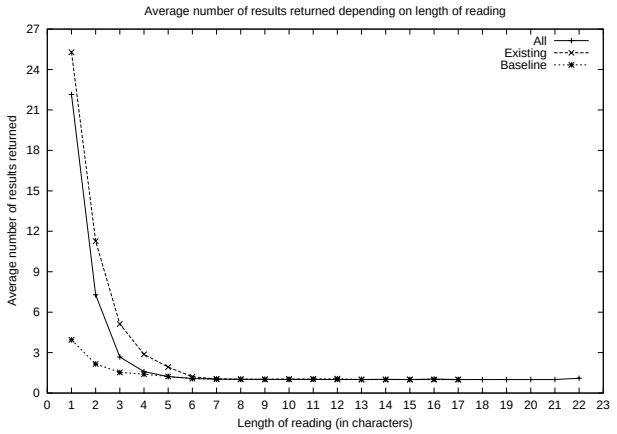

Figure 3: Distribution of results for different query string lengths

of candidates would be returned for each reading, obscuring dictionary entries for which the input is the correct reading. This could result in a high penalty for competent users who mostly search the dictionary with correct readings, potentially making the system unusable.

To verify this, we tried to establish how many candidates are returned for user queries over readings the system has knowledge of, and also tested whether the number of results depends on the length of the query.

The distribution of results returned for different queries is given in Figure 2, and the average number of results returned for different-length queries is given in Figure 3. In both figures, **Baseline** is calculated over only those readings in the original dictionary (i.e. correct readings); **Existing** is the subset of readings in the generated set that existed in the original dictionary; and **All** is all readings in the generated set. The distribution of the latter two sets is calculated over the generated set of readings.

In Figure 2 the x-axis represents the number of results returned for the given reading and the y-axis represents the natural log of the number of readings returning that number of results. It can be seen that only a few readings return a high number of entries. 308 out of 2,194,159 or 0.014% readings return over 30 results. As it happens, most of the readings returning a high number of results are readings that existed in the original dictionary, as can be seen from the fact that **Existing** and **All** are almost identical for x values over 30. Note that the average number of dictionary entries returned per reading is 1.21 for the complete set of generated readings.

Moreover, as seen from Figure 3 the number of results depends heavily on the length of the reading. In this figure, the x-axis gives the length of the reading in characters and the y-axis the average number of entries returned. It can be seen that queries containing 4 characters or more are likely to return 3 results or less on average. Here again, the **Existing** readings have the highest average of 2.88 results returned for 4 character queries. The 308 readings mentioned above were on average 2.80 characters in length.

From these results, it would appear that the returned number of entries is ordinarily not overwhelming, and provided that the desired entries are included in the list of candidates, the system should prove itself useful to a learner. Furthermore, if a user is able to postulate several readings for a target string, s/he is more likely to obtain the translation with less effort by querying with the longer of the two postulates.

4.2 Comparison with a conventional system

As the second part of evaluation, we tested to see whether the set of candidates returned for a query over the wrong reading, includes the desired entry. We ran the following experiment. As a data set we used a collection of 139 entries taken from a web site displaying real-world reading errors made by native speakers of Japanese.[10] For each entry, we queried our system with the erroneous reading to see whether the intended entry was returned among the system output. To transform this collection of items into a form suitable for dictionary querying, we converted all readings into hiragana, sometimes removing context words in the process. Table 3 gives a comparison of results returned in **simple** (conventional) and **intelligent** (proposed system) search modes. 62 entries, mostly proper names[11] and 4-

[10] `http://www.sutv.zaq.ne.jp/shirokuma/godoku.html`

[11] We have also implemented the proposed system with the ENAMDICT, a name dictionary in the EDICT distribution,

	Conventional	Our System
In dictionary	77	77
Ave. # Results	1.53	5.42
Successful	10	34
Mean Rank	1.4	4.71

Table 3: Comparison between a conventional dictionary look-up and our system

character proverbs, were not contained in the dictionary and have been excluded from evaluation. The erroneous readings of the 77 entries that were contained in the dictionary averaged 4.39 characters in length.

From Table 3 we can see that our system is able to handle more than 3 times more erroneous readings then the conventional system, representing an error rate reduction of 35.8%. However, the average number of results returned (5.42) and mean rank of the desired entry (4.71 — calculated only for successful queries) are still sufficiently small to make the system practically useful.

That the conventional system covers any erroneous readings at all is due to the fact that those readings are appropriate in alternative contexts, and as such both readings appear in the dictionary. Whereas our system is generally able to return all reading-variants for a given kanji string and therefore provide the full set of translations for the kanji string, conventional systems return only the translation for the given reading. That is, with our system, the learner will be able to determine which of the readings is appropriate for the given context based on the translation, whereas with conventional systems, they will be forced into attempting to contextualize a (potentially) wrong translation.

Out of 42 entries that our system did not handle, the majority of misreadings were due to the usage of incorrect character readings in compounds (17) and graphical similarity-induced error (16). Another 5 errors were a result of substituting the reading of a semantically-similar word, and the remaining 5 a result of interpreting words as personal names.

Finally, for the same data set we compared the relative rank of the correct and erroneous readings to see which was scored higher by our grading procedure. Given that the data set is intended to exemplify cases where the expected reading is different from the actual reading, we would expect the erroneous readings to rank higher than the actual readings. An average of 76.7 readings was created for the 34 entries. The average relative rank was 12.8 for erroneous readings and 19.6 for correct readings. Thus, on average, erroneous readings were ranked higher than the correct readings, in line with our prediction above.

Admittedly, this evaluation was over a data set of limited size, largely because of the difficulty in gaining access to naturally-occurring kanji–reading confusion data. The results are, however, promising.

4.3 Discussion

In order to emulate the limited cognitive abilities of a language learner, we have opted for a simplistic view of how individual kanji characters combine in compounds. In step 4 of preprocessing, we use the naive Bayes model to generate an overall probability for each reading, and in doing so assume that component readings are independent of each other, and that phonological and conjugational alternation in readings does not depend on lexical context. Clearly this is not the case. For example, kanji readings deriving from Chinese and native Japanese sources (*on* and *kun* readings, respectively) tend not to co-occur in compounds. Furthermore, phonological and conjugational alternations interact in subtle ways and are subject to a number of constraints (Vance, 1987).

However, depending on the proficiency level of the learner, s/he may not be aware of these rules, and thus may try to derive compound readings in a more straightforward fashion, which is adequately modeled through our simplistic independence-based model. As can be seen from preliminary experimentation, our model is effective in handling a large number of reading errors but can be improved further. We intend to modify it to incorporate further constraints in the generation process after observing the correlation between the search inputs and selected dictionary entries.

Furthermore, the current cognitive model does not include any notion of possible errors due to graphic or semantic similarity. But as seen from our preliminary experiment these error types are also common. For example, 墓地 *bochi* "graveyard" and 基地 *kichi* "base" are graphically very similar but read differently, and 物 *mono* "thing" and 事 *koto* "thing" are semantically similar but take different readings. This leads to the potential for cross-borrowing of errant readings between these kanji pairs.

Finally, we are working under the assumption that the target string is contained in the original dictionary and thus base all reading generation on the existing entries, assuming that the user will only attempt to look up words we have knowledge of. We also provide no immediate solution for random reading errors or for cases where user has no intuition as to how to read the characters in the target string.

allowing for name searches on the same basic methodology. We feel that this part of the system should prove itself useful even to the native speakers of Japanese who often experience problems reading uncommon personal or place names. However, as of yet, we have not evaluated this part of the system and will not discuss it in detail.

4.4 Future work

So far we have conducted only limited tests of correlation between the results returned and the target words. In order to truly evaluate the effectiveness of our system we need to perform experiments with a larger data set, ideally from actual user inputs (coupled with the desired dictionary entry). The reading generation and scoring procedure can be adjusted by adding and modifying various weight parameters to modify calculated probabilities and thus affect the results displayed.

Also, to get a full coverage of predictable errors, we would like to expand our model further to incorporate consideration of errors due to graphic or semantic similarity of kanji.

5 Conclusion

In this paper we have proposed a system design which accommodates user reading errors and supplements partial knowledge of the readings of Japanese words. Our method takes dictionary entries containing kanji characters and generates a number of readings for each. Readings are scored depending on their likeliness and stored in a system database accessed through a web interface. In response to a user query, the system displays dictionary entries likely to correspond to the reading entered. Initial evaluation indicates that the proposed system significantly increases error-resilience in dictionary searches.

Acknowledgements

This research was supported in part by the Research Collaboration between NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation and CSLI, Stanford University. We would particularly like to thank Ryo Okumura for help in the development of the FOKS system, Prof. Nishina Kikuko of the International Student Center (TITech) for initially hosting the FOKS system, and Francis Bond, Christoph Neumann and two anonymous referees for providing valuable feedback during the writing of this paper.

References

Timothy Baldwin and Hozumi Tanaka. 2000. A comparative study of unsupervised grapheme-phoneme alignment methods. In *Proc. of the 22nd Annual Meeting of the Cognitive Science Society (CogSci 2000)*, pages 597–602, Philadelphia.

Timothy Baldwin, Slaven Bilac, Ryo Okumura, Takenobu Tokunaga, and Hozumi Tanaka. 2002. Enhanced Japanese electronic dictionary look-up. In *Proc. of LREC*. (to appear).

Jim Breen. 2000. A WWW Japanese Dictionary. *Japanese Studies*, 20:313–317.

EDICT. 2000. EDICT Japanese-English Dictionary File. `ftp://ftp.cc.monash.edu.au/pub/nihongo/`.

EDR. 1995. *EDR Electronic Dictionary Technical Guide*. Japan Electronic Dictionary Research Institute, Ltd. In Japanese.

Yumi Ichimura, Yoshimi Saito, Kazuhiro Kimura, and Hideki Hirakawa. 2000. Kana-kanji conversion system with input support based on prediction. In *Proc. of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 341–347.

Tatuya Kitamura and Yoshiko Kawamura. 2000. Improving the dictionary display in a reading support system. International Symposium of Japanese Language Education. (In Japanese).

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

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *SNLR*, pages 22–28.

Masaaki Nagata. 1994. A stochastic Japanese morphological analyzer using a forward-DP backward-A^* N-best search algorithm. In *Proc. of the 15th International Conference on Computational Linguistics)(COLING 1994*, pages 201–207.

NLI. 1986. *Character and Writing system Education*, volume 14 of *Japanese Language Education Reference*. National Language Institute. (in Japanese).

Masahito Takahashi, Tsuyoshi Shichu, Kenji Yoshimura, and Kosho Shudo. 1996. Processing homonyms in the kana-to-kanji conversion. In *Proc. of the 16th International Conference on Computational Linguistics (COLING 1996)*, pages 1135–1138.

Natsuko Tsujimura. 1996. *An Introduction to Japanese Linguistics*. Blackwell, Cambridge, Massachusetts, first edition.

Timothy J. Vance. 1987. *Introduction to Japanese Phonology*. SUNY Press, New York.