Efficient Grapheme-phoneme Alignment for Japanese

Lars Yencken and **Timothy Baldwin**

Dept. of Computer Science and Software Engineering
University of Melbourne
Victoria 3010 Australia

NICTA Victoria Research Lab
University of Melbourne
Victoria 3010 Australia

{lars,tim}@csse.unimelb.edu.au

Abstract

Current approaches to the grapheme-phoneme alignment problem for Japanese achieve good accuracy, but are extremely computationally expensive. In this paper we evaluate various modifications to previous algorithms for both the alignment and okurigana detection subtasks. The best algorithm achieved accuracy of 96.2% for the combined task on a limited data set, and was significantly more efficient than previous approaches.

1 Introduction

Alignment is the task of, for two streams of data which represent alternate construals of the same basic information content, identifying corresponding segments within the two streams. A common alignment task in computational linguistics is word alignment, whereby given an English sentence and its French translation, say, each English word n-gram is aligned with its French translation (Brown et al., 1993; Manning and Schütze, 2000). The combined set of such alignments derived from a parallel corpus is generally used to train the translation model in statistical machine translation systems. Other alignment tasks in computational linguistics include sentence alignment, structural alignment (e.g. as a means of grammar inference), and grapheme-phoneme alignment.

The grapheme-phoneme ("GP") alignment task aims to *maximally* segment the orthographic form of an utterance into morphophonemic units, and align these units to a phonetic transcription of the utterance. Maximal indicates the desire to segment grapheme strings into the smallest meaningful units possible. Taking the English example word *battleship* and its phonetic transcription /bætlʃɪp/, one possible alignment is:

b	a	tt	le	sh	i	p
b	æ	t	l	ʃ	ɪ	p

Note that alignment in general is many-to-many. In the example above, *tt* aligns to /t/, *le* aligns to /l/ and *sh* aligns to /ʃ/. Equally it might be possible for some letters to align to an empty string. This task is challenging for any language without a one-to-one correspondence between individual graphemes and phonemes, as is the case with English (Zhang et al., 1999), Japanese (considering graphemes as kanji characters), and indeed most languages with a pre-existing writing system.

GP alignment is a prerequisite for many applications. For example, the alignment process, and the resulting aligned GP tuples, are a precursor to achieving automated grapheme-to-phoneme mappings for several text-to-speech systems (Allen et al., 1987; Sejnowski and Rosenberg, 1987; Sloat, 1996; Black et al., 1998). Further uses include accented lexicon compression (Pagel et al., 1998), identification of cognates (Kondrak, 2003), Japanese-English back-transliteration (Knight and Graehl, 1998; Bilac and Tanaka, 2005a, 2005b) and finally the FOKS dictionary system for Japanese learners (Bilac, 2002; Bilac et al., 2002), which provides the context for our work.

There are several successful approaches to Japanese GP alignment, notably the iterative rule-based approach taken by Bilac et al. (1999), later followed by an unsupervised statistical model based on TF-IDF by Baldwin and Tanaka (1999a, 1999b). Although these models were found to have high accuracy, their iterative approach had a high computational cost, making them impractical for many real-world applications. For the statistical models, this is partially a consequence of their strongly unsupervised nature. We thus explore the use of the Edict and Kanjidic electronic dictionaries (Breen, 1995) as means of constraining the alignment search space and reducing computational complexity.

The goal of this paper is to compare sev-

eral different GP alignment methods in order to achieve equivalent or better alignment accuracy to that for existing methods, at a much lower computational cost. To achieve this goal, we split the task of GP alignment into a pure alignment subtask and an okurigana detection subtask, and compare algorithm variants of pre-existing approaches for both. As our base model, we use the top performing statistical model from Baldwin and Tanaka (2000).

The remainder of this paper is structured as follows. First, we discuss the FOKS system, an important motivator for this work (Section 2). We then discuss the GP-alignment problem for Japanese in greater depth (Section 3), before giving details of the baseline statistical model and our modifications to it (Section 4). Finally, we discuss our results on a manually aligned test data set (Section 5).

2 FOKS dictionary system

GP alignment is an important step in the pipeline that drives the FOKS ("Forgiving Online Kanji Search") dictionary interface (Bilac, 2002), our particular research interest. Whereas normal electronic dictionaries will not provide the target word if an incorrect reading is looked up, FOKS is able to compensate for learner mistakes by dynamically predicting readings for compounds, and aims to direct the user to the correct word despite possible mistakes in the entered reading.

For example, suppose the user wishes to look up 風邪 [ka-ze] "common cold". He or she may know the kanji 風 [ka-ze/fu-u] "wind", and also 邪 [yo-ko-shi-ma/jya] "evil, wicked", and thus guess that the reading for 風邪 is ka-ze-yo-ko-shi-ma, one possible combination of readings. However, the correct reading ka-ze is non-compositional. Despite the incorrect guess, FOKS still lists the target word with the correct reading in its list of candidates for the guessed reading.

The back-end data that drives FOKS is constructed as follows. Firstly, all entries in the Edict dictionary are GP aligned. The subsequent GP tuples are counted to estimate the probability $P(r|k)$ of a given reading r for a given grapheme segment k. Composing segment probabilities together gives the probability $P(r|s)$ of an entire dictionary entry s taking reading r. Using Bayes rule, we finally calculate $P(s|r)$, the probability of a dictionary entry s being the target entry given the user provided reading r. The entries s with non-zero probabilities form our list of candidates for the user's query, and the probabilities $P(s|r)$ provide the basis of the ranking (Bilac et al., 2002). GP alignment allows us to calculate each $P(r|k)$ and is thus a pivotal supporting technology which underlies the FOKS system.

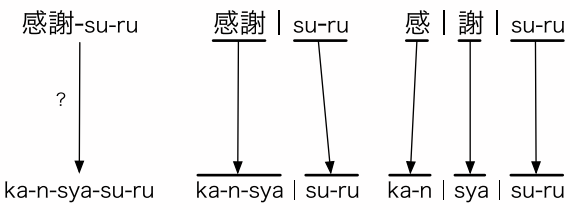

Figure 1: A typical dictionary entry requiring GP alignment, with two potential alignments shown

3 Grapheme-phoneme alignment in Japanese

In the context of Japanese, the GP-alignment task has a few peculiarities. Japanese has three scripts: *kanji*, *hiragana*, and *katakana*. Since hiragana and katakana (or *kana* collectively) are essentially phonetic, we can represent the phoneme string using either of these scripts directly. Kanji on the other hand are ideographic rather than phonetic. Each kanji may have many readings as a single unit, and may also form part of larger units which themselves take on one or more readings. To emphasize the difference between scripts, we shall use romanizations for the phonetic scripts. Figure 1 gives an example for 感謝する [ka-n-sya-su-ru] "to give thanks, be thankful".

Given that kana are phonetic, the main task is then reduced to determining how the kanji elements should be segmented, and what elements of the phoneme string they correspond to. Below, we outline four features of Japanese that impede this task.

3.1 Okurigana alternation

Individual kanji segments do not always correspond to minimal units in language. Often a hiragana suffix of some description (usually conjugational) is required, which we term *okurigana*. Verb and adjective conjugation fall under this category: for example 行-ku [i-ku] "go" in plain form changes to 行-ke [i-ke] in the imperative. Any useful segmentation should thus include such suffixes along with their kanji stem

in order to preserve the basic morpho-phonemic structure of the compound.

Although most cases of okurigana represent verb and adjective conjugation, there are many general cases such as that of the kanji 取, which occurs in compounds almost exclusively as 取-ri [to-ri], but also has an alternate where the suffix ri is conflated with the kanji stem (such as in 取分 [to-ri-bu-n] "one's share or portion"). With some lexemes, both alternants are possible, such as in 取-ri-分 [to-ri-bu-n]. It is desirable for systems to be able to capture such alternations, in order to achieve consistent segmentation behaviour and attain an accurate estimate of the frequency with which a given kanji occurs with a particular reading (independent of the exact lexical form of the word).

3.2 Sequential voicing

Sequential voicing occurs when a tailing segment has its initial consonant voiced. For example: 本 [ho-n] "book" + 棚 [ta-na] "shelf" → 本棚 [ho-n-da-na] "bookshelf". Although sequential voicing is notoriously unpredictable, its potential occurrence is constrained by Lyman's law, which states that sequential voicing will not occur where there are existing voiced obstruents in the tailing segment (Vance, 1987). It occurs in about 75% of cases where Lyman's law is not violated, with some systematic irregularities for noun-noun compounds as found in recent work by Rosen (2003).

Alignment methods based on precedence or frequency counts may be hindered by sequential voicing, since aligned grapheme/phoneme pairs may not be recognised as phonological variants of previously seen kanji–reading pairs. Fortunately, devoicing is relatively simple, so a common approach is to simply consider voiced and devoiced grapheme/phoneme pairs to be equivalent for counting or comparison.

3.3 Sound euphony

Sound euphony occurs when the last syllable of a leading segment is modified to match the sound of the tailing segment. This is marked uniquely by the っ kana character in Japanese. For example: 国 [ko-ku] "country" + 境 [kyo-o] "boundary" → 国境 [ko-k-kyo-o] "national border". Unlike sequential voicing, which imposes a reversible transformation, it is not clear from 国境 [ko-k-kyo-o] "national border" what the original kana ending for 国 was (possibilities include ko-ki, ko-ku, ko-su and ko-tsu).

3.4 Grapheme gapping

Occasionally a kana is omitted from the written form of a word, but does not constitute a component of the readings of the neighbouring kanji. Typically the kana can also be explicitly included in the written form of the word. For example: 山 [ya-ma] "mountain" + no [GENITIVE] + 手 [te] "hand" can be written as either 山手 or 山-no-手, both with reading [ya-ma-no-te].

Grapheme gapping is very rare, normally only occurs with the particles ga or no, and tends not to be productive, suggesting that even approaches aimed at open text are better off simply storing each case individually. The only productive case involving a kanji is 真 [ma]. For example: 真 [ma] "true/pure" + 暗闇 [ku-ra-ya-mi] "darkness" → 真暗闇 [ma-ku-ra-ya-mi] "pitch dark", or 真っ暗闇 [ma-k-ku-ra-ya-mi] for emphasis.

4 Multi-step alignment

In this section, we first describe the baseline algorithm of Baldwin and Tanaka (1999a, 1999b), before introducing the modifications we propose in this research.

4.1 Overview

A high-level depiction of the unsupervised alignment method of Baldwin and Tanaka (1999a, 1999b) is given in Figure 2. Firstly, all potential segmentations and alignments for input entries are created. Each entry will have potential segmentations and alignments per segmentation which number exponentially in the entry's orthographic length.

Some simple linguistic constraints used as forward constraints to reduce this number are strictly linear alignment, a minimum of one phoneme aligned to each grapheme, and a restriction that each alignment must successfully match any kana entry in the grapheme string with its equivalent phoneme entry. Further constraints used to prune entries include matching okurigana to pre-clustered variants and forcing script-boundaries (except kanji to hiragana boundaries) to correspond to segment boundaries.

Based on the linguistic constraints, we can reasonably expect to have uniquely determined some number of alignments for any sufficiently diverse data set.[1] The uniquely determined

[1] Notable exceptions to this are dictionaries of 4-kanji proverbs, such as the 4JWORDS electronic dictionary,

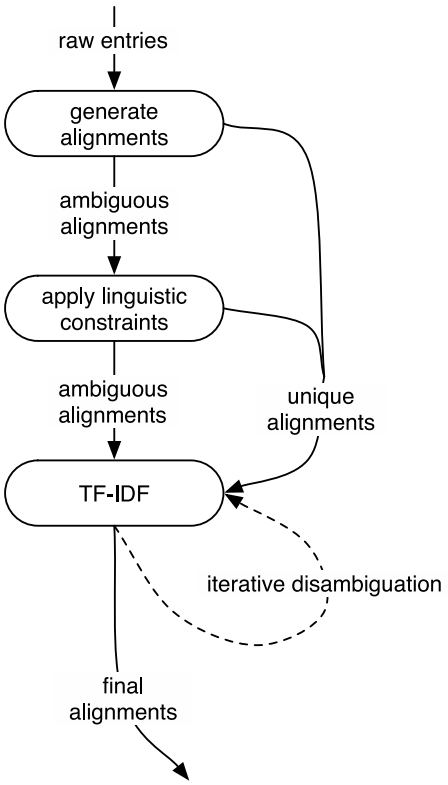

Figure 2: The TF-IDF based alignment algorithm

alignments and the remaining ambiguous alignments are both used separately to seed frequency counts for the TF-IDF model.

TF-IDF is a family of models originally developed for IR tasks, combining the TF (term frequency) and IDF (inverse document frequency) heuristics (Baeza-Yates and Ribiero-Neto, 1999). In the GP alignment task, they mediate the tension between oversegmenting and undersegmenting. The TF value is largest for the most frequently occurring GP pair given any grapheme; an oversegmented alignment produces rarer segments with lower frequency, penalizing the TF score. The IDF value on the other hand is largest for segments which occur in a wide variety of contexts, and penalises undersegmenting.

4.2 TF-IDF Alignment

We use a modified version of the TF-IDF model which takes into account the differing level of confidence we have in our frequency counts between solved ($freq_s$) and ambiguous ($freq_u$) alignments (Baldwin and Tanaka, 2000). For each alignment, we count the occurrence of each grapheme segment $\langle g \rangle$, of each aligned grapheme/phoneme segment pair $\langle g, p \rangle$, and of the same pair with one additional pair of context on either side $\langle g, p, ctxt \rangle$. For any frequency lookup, the w_s and w_u constants provide a weighting between information from solved and ambiguous alignments:

$$wtf(x) = w_s \times freq_s(x) + w_u \times freq_u(x) \quad (1)$$

To score a potential alignment, we calculate the *tf* and *idf* scores for each grapheme/phoneme segment pair and multiply them together as in Equations 2-4. The score for the whole alignment is the average of the scores for every pair which contains a kanji character, since these are the non-trivial pairs. The constant α is intended as a smoothing factor for the TF and IDF scores. It must be assigned such that $0 < \alpha < w_u \leq w_s$.

$$tf(g,p) = \frac{wtf(\langle g,p \rangle) - w_u + \alpha}{wtf(\langle g \rangle)} \quad (2)$$

$$idf(g,p,ctxt) = \log \frac{wtf(\langle g,p \rangle)}{wtf(\langle g,p,ctxt \rangle) - w_u + \alpha} \quad (3)$$

$$score(g,p,ctxt) = tf(g,p) \times idf(g,p,ctxt) \quad (4)$$

Once all potential alignments have been scored, the highest-scoring alignment is chosen to disambiguate its entry. Its counts are removed from the unsolved pool and added to the solved pool, and algorithm reiterates with updated counts. In this way entries are iteratively disambiguated until no more remain, and the algorithm is complete.

Although effective, the iterative algorithm is extremely expensive, with two main costs. Firstly, as with any alignment task where two strings of length l and m respectively need to be aligned, there are 2^{lm} possible alignments before applying constraints (Brown et al., 1993). In our task, kanji essentially form free variables in the alignment, whereas kana align to themselves, constraining the search space. Entries with many kanji and no kana to constrain them thus have prohibitively large numbers of possible alignments. These cases bloat the number of potential alignments to be rescored on each iteration so much that including them makes our main algorithm infeasibly expensive.

whose entries' grapheme forms lack kana to help eliminate possible alignments.

The second bottleneck is in the average case. Suppose there are n alignments pairs, each with p possible alignments. Then the cost of the iterative rescoring loop is $O((np)^2)$. Even having removed the problem cases above, if p is still high on average, the problem will prove intractable for suitably large n. As a comparison, the evaluation set we use has 5000 elements, yet the Edict dictionary has over 110,000 entries, representing a near 500 fold expected increase in computation time. Although this could be mitigated by simply breaking the input down into smaller subsets for processing, it is desirable to process all the data in the same iterative loop, since this gives greatest consistency of alignment.

Strategies to reduce the average case for p and to eliminate the worst case for p thus form the basis for our attempts at modifying the algorithm.

4.3 Modified algorithm

The modified algorithm diverges from the unsupervised algorithm in three main respects. Firstly, we separate out okurigana handling into a separate step after alignment, benefiting both efficiency and error measurement. Secondly, a reading model is introduced based on the Kanjidic electronic dictionary[2] and is used to disambiguate the majority of remaining cases before the TF-IDF model is reached. Thirdly, we provide a maximum alignment size cutoff above which we use a simplified non-iterative alignment algorithm which meets resource constraints for problem cases. We discuss these changes below.

4.3.1 Separating okurigana handling

The okurigana handling in the original algorithm involves pre-clustering okurigana alternates, and attempting to restrict alignments to match these alternates wherever possible. Whilst this constraint does help reduce potential alignments, it also limits the application of the stronger constraint that script boundaries in the grapheme string must correspond to segment boundaries (i.e. every occurrence of a kanji–hiragana script boundary must be considered as a potential okurigana site). If okurigana detection is left as a post-processing task, we can strengthen this constraint to include all script boundaries, instead of omitting kanji-to-hiragana boundaries. This in turn provides a larger gain than the original okurigana constraint, since more entries are fully disambiguated.

The GP-alignment task is then split into two parts: a pure alignment task, which can be carried out as per the original algorithm, and a separate okurigana detection task. This redesign also allows us to separately evaluate the error introduced during alignment, and that introduced during okurigana detection, and thus allows us to experiment more freely with possible models.

4.3.2 Short and long entries

Ultimately, any method which considers all possible alignments for a long entry will not scale well, since potential alignments increase exponentially with input length. We can however extend the applicability of the algorithms considered by simply disambiguating long entries in a non-iterative manner.

The number of potential alignments for an entry can be estimated directly from the number of consecutive kanji. Our approach is to simply to count the number of consecutive kanji in the grapheme string. If this number is above a given threshold, we delay alignment until all the short entries have been aligned. We then use the richer statistical model to align all the long entries in a single pass, without holding their potential alignments in memory.

Although long entries were not an issue in our evaluation set, a threshold set experimentally to 5 consecutive kanji worked well using the Edict dictionary as input, where such entries can prove difficult.

4.3.3 Reading model

For the pure alignment task, we added an additional reading model which disambiguates entries by eliminating alignments whose single kanji readings do not correspond to those in the Kanjidic and KANJD212 electronic dictionaries. These dictionaries list common readings for all kanji in the JIS X 0208-1990 and JIS X 0212-1990 standards respectively, covering 12154 kanji in total. Effectively, we are applying the closed world assumption and allowing only those alignment candidates for which each grapheme unit is associated with a known reading. Only in the instance of over-constraint, i.e. every GP alignment containing at least one unattested reading for a grapheme unit, do we relax this constraint over the overall alignment candidate space for the given grapheme string.

A simple example of disambiguation using the reading model is that of 一両 [i-chi-ryo-u] "one

[2] http://www.csse.monash.edu.au/~jwb/kanjidic.html

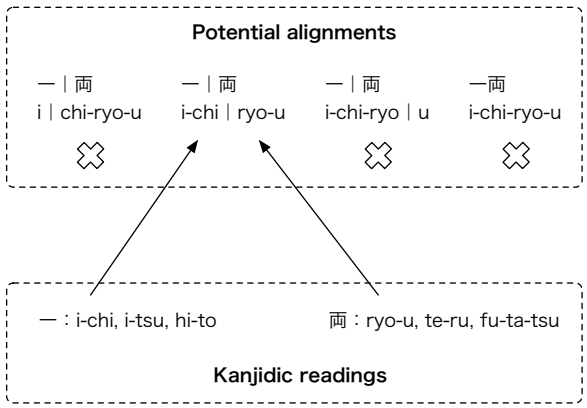

Figure 3: Disambiguation using the reading model

vehicle" as shown in Figure 3. Since only one of the potential alignments is compatible with the known readings, we then select it as the correct alignment. As an indication of the effectiveness of the reading model, our initial constraints uniquely determine 31.1% of the entries in the Edict dictionary.[3] The reading model disambiguates a further 60.6% of entries, effectively decreasing the input to the iterative alignment algorithm by an order of magnitude, to the remaining 8.3%.

4.3.4 Heuristic variants

We could continue to use the original TF-IDF model over the residue which is not disambiguated by the reading model, although the type of input has changed considerably after passing through the reading model. Since the reading model is likely to fully disambiguate any entry containing only single kanji segments, the only remaining ambiguous models are likely to be those with solutions containing multi-kanji segments (which do not occur in either Kanjidic or KANJD212); an instance of a multi-kanji segment is our earlier example 風邪 [ka-ze] "common cold". With this in mind, we compare the original TF-IDF model (our baseline) with similar models using TF only, IDF only, or random selection to choose which entry/alignment to disambiguate next.

4.3.5 Okurigana detection

We similarly wish to determine what form of okurigana detection and realignment model is most appropriate. Since the majority of entries in the Edict dictionary (our main experimental data set) which contain potential okurigana sites (i.e. kanji followed by hiragana) do contain okurigana in some form, we use as our baseline the simple assumption that every such site is an instance of okurigana. In this manner, the baseline simply removes every kanji-to-kana segment boundary. As a small enhancement, the boundary is not removed if the tailing kana segment is one of the hiragana particles no, ga or ni, which frequently occur alone.

We consider three alternative okurigana models to compare to our baseline, of increasing complexity and expected coverage. Firstly, the Kanjidic dictionary contains common okurigana suffixes for some kanji with conjugating entries. Thus our first model uses these suffixes verbatim for okurigana detection. The coverage of okurigana suffixes in Kanjidic is somewhat patchy, so in our second model, in addition to Kanjidic suffixes, we also perform a frequency count over all potential okurigana sites in the Edict dictionary, and include any occurrences above a set threshold as okurigana.

Finally, most instances of okurigana are due to verb conjugation. As well as taking straight suffixes from the previous models, this final model harvests verbs from Edict. Most verb entries in Edict have a tag marking them as *ichidan*, *godan* or *suru* verbs.[4] The verb type and stem allow us to conjugate regular verbs variously, giving us a large number of new okurigana suffixes not present in the previous models. In order to improve accuracy, all three methods fall back to the baseline method if they do not detect any okurigana.

5 Evaluation

Having teased apart the alignment and okurigana detection algorithms, we are in a position to separately evaluate their performance. Our test set for the combined task consists of 5000 randomly chosen and manually aligned examples from Edict, from which we then separated out an individual evaluation set for each subtask.

Since we are also interested in efficiency, we provide execution time as measured by elapsed time on a standard Pentium 4 desktop PC. Our emphasis however is on the *relative* time taken by different algorithms rather than the exact

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

[4] The tagset for Edict verbs is larger than this, but the additional tags largely mark subclasses and exceptions of the three main classes, which we ignore for the sake of simplicity.

time as measured.

In the following section we first evaluate alignment and okurigana detection separately, then we evaluate okurigana detection, and finally we assess performance over the combined task.

5.1 Alignment

We first compare the accuracy of the various alignment algorithm variants, as given in Table 1. After some experimentation, parameter values of 0.05 for α, and 2.5 for w_s and w_u were found to yield the best results, and were hence used to generate the results we discuss here.

For each of the non-random heuristics, we expect that the iterative version will achieve higher accuracy than the non-iterative version, since the statistical model is rebuilt each iteration adding the best example from the last. As such, this represents a time/accuracy trade-off, a fact confirmed by our data (see Table 2). The gain | 2% in the case of TF-IDF, 4% for IDF alone | comes at the cost of an order of magnitude larger execution time, which also increases exponentially with the number of input entries.

In contrast, the Kanjidic model consistently achieves a very high accuracy regardless of the heuristic chosen. A large number of entries are immediately disambiguated by the Kanjidic model, thus initially improving accuracy and then facilitating use of more accurate statistics in the iterative algorithm without significant penalty to efficiency. We also expect the Kanjidic model's execution time to scale more moderately with the number of input entries than the original iterative algorithm, since a far lesser proportion of the entries require iterative disambiguation.

Comparing the individual heuristics at this stage, a surprise is that the IDF heuristic attains equivalent results to the TF-IDF heuristic, suggesting that broad occurrence of $\langle g, p \rangle$ pairs is a good indicator of their alignment probability. The TF heuristic in comparison performs worse than simply choosing randomly, suggesting that the proportion of times a grapheme occurs as the current $\langle g, p \rangle$ pair is a very poor indication of its alignment probability.

5.2 Okurigana detection

We now compare the performance of our okurigana detection algorithms. All the algorithms we compare are linear in the size of the input and thus run in much less time than the alignment phase, thus efficiency is not a significant criteria in choosing between them. The accuracy found by each model is shown in Table 3.

Model	**Accuracy**
Simple	98.1%
Kanjidic	98.3%
Co-occurrence	97.7%
Verb conjugation	97.7%

Table 3: Okurigana detection accuracy across models

Interestingly, the simple model which assumes that every potential case of okurigana *is* okurigana performs extremely well, beaten only by the addition of the Kanjidic common okurigana stems. Adding more information to the model about valid okurigana occurrences even reduces the accuracy slightly over our test data.

Rather than indicating blanket properties of these models, the results suggest properties of our testing data. Since it consists entirely of dictionary entries without the common hiragana particles which would occur in open text, this greedy approach is very suitable, and suffers few of the shortcomings which it would normally face.

In open text, we would consistently expect additional language features between lexical items which would break the assumptions made by our simple model, and thus reduce its accuracy dramatically. In contrast, the full verb conjugation model would then be expected to perform best, since it has the most information to accurately detect cases of okurigana even in the presence of other features.

5.3 Combined task

Selecting the two models which performed best on our test data, we can now evaluate the pair on the combined task. For the alignment subtask, the IDF heuristic with Kanjidic was used. For the okurigana detection subtask, the simple algorithm is used. The results are shown in Table 4.

A final accuracy of 96.2% was achieved, with the errors caused mostly in the alignment subtask. As predicted, grapheme gapping was a source of errors only in a small percentage of cases, justifying its exclusion from our model. This level of accuracy if equivalent to that of earlier models, yet it has been achieved with a much lower computational cost. Examples of incorrect alignment are given in Figure 4 below.

Example (a) shows a grapheme gapping error,

ACCURACY (%)	Random	TF	IDF	TF-IDF
Iterative	47.8	23.7	94.7	93.4
Single-pass	47.3	23.6	90.5	90.8
Kanjidic	94.4	92.9	98.0	97.9

Table 1: Alignment accuracy across models

TIME (M:S)	Random	TF	IDF	TF-IDF
Iterative	0:10	24:10	22:47	21:54
Single-pass	0:10	0:11	0:09	0:10
Kanjidic	0:12	0:27	0:24	0:24

Table 2: Alignment execution time across models

Status	Count	Percentage
Correct	4809	96.2%
Incorrect	191	3.8%
→ Gapping	6	0.1%
→ Alignment	163	3.3%
→ Okurigana	22	0.4%

Table 4: Best model accuracy for the combined task

a.	*Output*	真 | 盛	ma-s | sa-ka-ri
	Correct	真 | 盛	ma-(s) | sa-ka-ri
		"full bloom"	
b.	*Output*	挟 | 撃 | chi	ha-sa | mi-u | chi
	Correct	挟 | 撃 | chi	ha-sa-mi | u | chi
		"pincer attack"	
c.	*Output*	赤-n | 坊	a-ka-n | bo-u
	Correct	赤 | n | 坊	a-ka | n | bo-u
		"baby"	

Figure 4: Examples of incorrect alignment in the combined task

where the output, although correctly segmented and aligned, attributes the additional *s* sound to the 真 kanji instead of detecting it as a gapped grapheme. In example (b) we see a typical alignment error, where one kanji has been attributed part of the reading of another. Finally, example (c) gives an error in okurigana detection, where the *n* kana is erroneously detected as an okurigana suffix of the 赤 kanji.

6 Extensions

Although current work is suitable for use with the FOKS system, it is still untested on open text. The lack of suitable aligned data is the main obstacle to creating a system with wider applicability. Of the two subtasks, alignment should remain relatively unchanged in the move to open text, and we expect the IDF algorithm with Kanjidic to continue to perform well.

Okurigana detection remains the harder problem, for tasks which require it. The verb-conjugation model, despite its relatively poor performance for dictionary entries, suggests itself as the most fruitful approach to accurate detection for open text, and could easily be extended. In particular, the addition of conjugation suffixes of high-frequency irregular verbs would be a straightforward way to boost accuracy.

7 Conclusion

We have decomposed the GP alignment task into an alignment subtask and an okurigana detection subtask, and explored various algorithm variants for use in both. In particular, the iterative IDF heuristic with a Kanjidic reading model provided the best accuracy in significantly less time than the original algorithm. For the okurigana detection subtask, a simple model outperformed more complicated models of conjugation due to peculiarities of dictionary entries as input to alignment.

References

Jonathan Allen, Sheri Hunnicut, and Dennis Klatt. 1987. *From Text To Speech, The MITTALK System.* Cambridge University Press, Cambridge, UK.

Ricardo Baeza-Yates and Berthier Ribiero-Neto. 1999. *Modern Information Retrieval.* Addison Wesley / ACM press.

Timothy Baldwin and Hozumi Tanaka. 1999a. The applications of unsupervised learning to Japanese grapheme-phoneme alignment. In *Proc. ACL*

Workshop on Unsupervised Learning in Natural Language, College Park, USA.

Timothy Baldwin and Hozumi Tanaka. 1999b. Automated Japanese grapheme-phoneme alignment. In *Proc. International Conference on Cognitive Science*, pages 349–354, Tokyo, Japan.

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

Slaven Bilac and Hozumi Tanaka. 2005a. Direct combination of spelling and pronunciation information for robust back-transliteration. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 413–424. January.

Slaven Bilac and Hozumi Tanaka. 2005b. Improving back-transliteration by combining information sources. In Keh-Yih Su, Jun'ichi Tsujii, Jong-Hyeok Lee, and Oi Yee Kwong, editors, *Proc. 1st International Joint Conference on Natural Language Processing*, pages 216–223, January.

Slaven Bilac, Timothy Baldwin, and Hozumi Tanaka. 1999. Incremental Japanese grapheme-phoneme alignment. In *Information Processing Society of Japan SIG Notes*, volume 99-NL-209, pages 47–54.

Slaven Bilac, Timothy Baldwin, and Hozumi Tanaka. 2002. Bringing the dictionary to the user: the FOKS system. In *Proc. 19th International Conference on Computational Linguistics*, pages 85–91, Taipei, Taiwan.

Slaven Bilac. 2002. Intelligent dictionary interface for learners of Japanese. Master's thesis, Tokyo Institute of Technology.

Alan W. Black, Kevin A. Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *Proc. 3rd ESCA Workshop on Speech Synthesis*, pages 77–80, Jenolan Caves, Australia.

Jim Breen. 1995. *Building an electronic Japanese-English dictionary.* Japanese Studies Association of Australia Conference (`http://www.csse.monash.edu.au/~jwb/jsaa_paper/hpaper.html`).

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

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

Grzegorz Kondrak. 2003. Identifying complex sound correspondences in bilingual wordlists. In Alexander Gelbukh, editor, *Proc. 4th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 432–443, Berlin. Springer-Verlag.

Christopher D. Manning and Hinrich Schütze. 2000. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, Massachusetts.

V. Pagel, K. Lenzo, and A.W. Black. 1998. Letter to sound rules for accented lexicon compression. In *Proc. 5th International Conference on Spoken Language Processsing*, pages 252–255.

Eric Rosen. 2003. Systematic irregularity in Japanese rendaku: How the grammarmediates patterned lexical exceptions. *Canadian Journal of Linguistics*, (48):1–37.

T. Sejnowski and C. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

Richard Sloat. 1996. Multilingual text analysis for text-to-speech synthesis. *Natural Language Engineering*, 4(2).

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

Jianna Jian Zhang, Howard J. Hamilton, and Nick J. Cercone. 1999. Learning english grapheme segmentation using the iterated version space algorithm. In Andrzej Skowron and Zbigniew W. Raś, editors, *Proc. 11th International Symposium on Methodologies for Intelligent Systems*, Warsaw, Poland. Springer-Verlag.