CS322 - Final Project **Tegan and Jack**

Final Project

Part 1: The Question

Japanese is made up of three different character sets: hiragana, katakana, and kanji. Hiragana are characters with a 1-1 sounds correspondence. For each possible morae, the Japanese version of a syllable which includes an optional consonant followed by a vowel, there is exactly one hiragana associated with it. Hiragana are also most often used as particles, in verb conjugations, and other functional contexts. Katakana also have 1-1 sound mapping, but are used specifically for foreign words. And kanji are Chinese characters. They have 1-1 meaning correspondence, but can have multiple different pronunciations depending on context and combinations with other kanji or hiragana characters.

The problem we aimed to solve is: given a sentence in Japanese, can we determine the readings (pronunciation) of the kanji within the sentence?

We used the paper "Kanji to Hiragana Based on a Length-Constrained N-Gram Analysis", published in 1999 by Joseph Picone, Tom Staples, Kazuhiro Kondo, and Nozomi Arai.

Part 2: The Algorithm

The algorithm we used relies on two parts: building a dictionary of valid n-grams, and determining word boundaries.

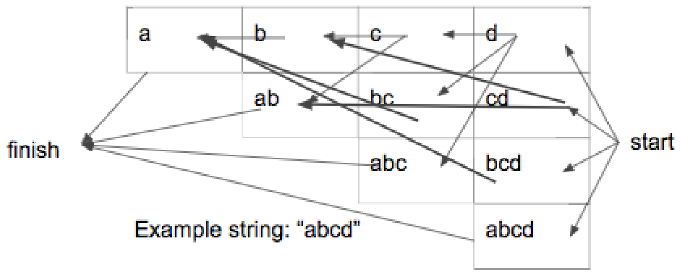
Building a dictionary of valid n-grams

In this section, we identify valid n-gram combinations from a set of training data, and store them into a dictionary with associated hiragana pronunciation and weights. Originally Picone et. al. thought to use frequency probabilities of these n-grams to calculate their weight, but instead found that a simpler method produces the same accuracy and was more computationally efficient: the weight assigned directly correlates to the length of the n-gram in question. We also give longer n-grams a slight boost so that, when given the option of reading a string of k characters as a single k-gram, or a combination of smaller grams, the algorithm favors the longer k-gram. This is meant to favor using as much context as possible to determine each kanji reading.

We used jmdict/EDICT, a freely available Japanese-to-English dictionary compiled by Jim Breen, which includes both pronunciation and meaning of thousands of Japanese words.

Determining word boundaries

The next step of the algorithm is, given the dictionary and a Japanese sentence, we must determine word boundaries–in other words, we must parse the sentence to determine which n-grams to run through the dictionary, and we'd like to do it in an optimal (maximum weight) way. To do this, we translate the sentence to a correlated graph: vertex (i, j) corresponds to the j-gram which end at character i in the sentence, and each vertex is given the weight of its associated j-gram in the dictionary. If the j-gram doesn't exist in the dictionary, it's given a weight of 0. Two vertices connect if their associated grams are found next to each other in the sentence. We also add a start vertex *start*, which connects to all vertices (last gram in sentence, j), and an end vertex $finish$, which connects to all vertices $(0, j)$. Finally, we traverse the graph find the longest path from *start* to $finish$.



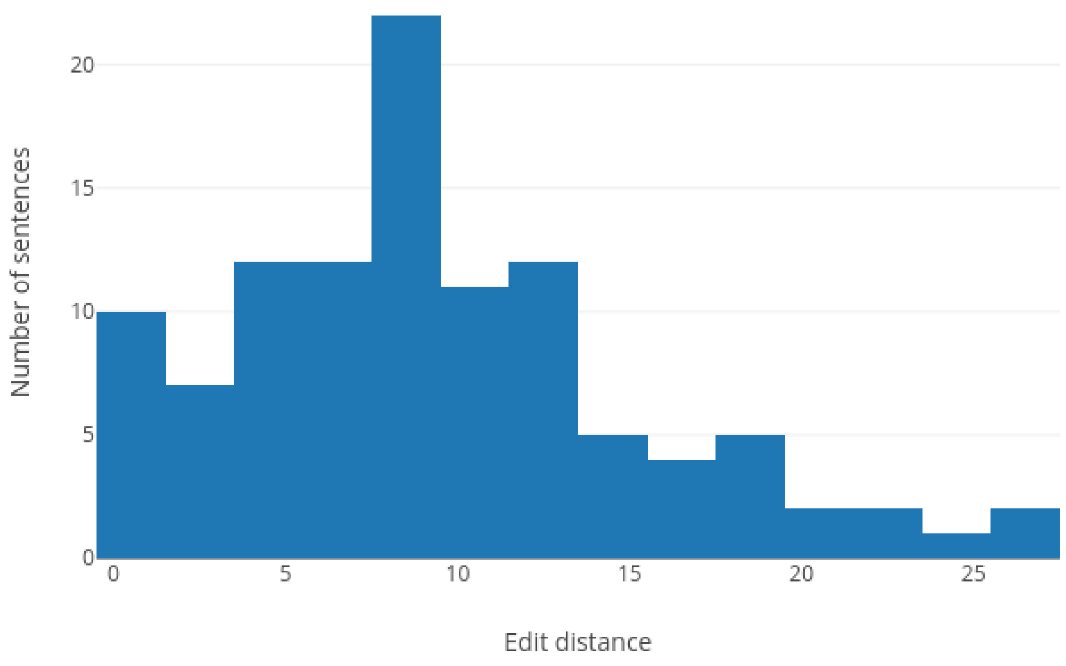
Because of the way in which we create this graph, D, it's guaranteed to be an acyclic digraph. Therefore, there exists a linear time $O(|V| + |E|)$ algorithm to find the longest path. We first create a topological ordering on the graph, an ordering of the vertices such that for any vertex v in the ordering, any other vertices w that have edges coming into v must necessarily come before v in the ordering. We can then iterate through the ordering, computing longest path lengths from *start* as we go, until we arrive at $finish$. We then backtrack to find the exact longest path. This corresponds directly with a maximum weight parse of the sentence. The for each vertex (i, j) in the longest path, we run the associated j-gram through the dictionary if possible to retrieve its pronunciation.

To get test data, we found a Japanese news site, http://www3.nhk.or.jp/news/easy/, which offers readings of each kanji in their articles. We were able to scrape the html of the site to pull both sentences with kanji, and a full sentence of their associated readings. The only issue we ran into here was that the readings of numbers was not given, so we had to adjust that within the test data by hand. We ended up with 108 total sentence pairs after scraping 14 articles.

Part 3: Results and Analysis

Our implementation was not as successful as the one used by the paper that provided the algorithm, but surpassed our expectations given the quality of the training data. We used edit distance across a whole sentence as our metric for evaluation. Figure 1 represents the results of our algorithm across all of our test data. We used penalties of 1, 1, and 2 for insertion, deletion, and substitution respectively. Originally, we encountered issues with number translation, and results improved dramatically once we added a first pass that checked if the number was in the dictionary, and if not, translated the number directly using an algorithm of our own design.

Figure 1: Accuracy of results as determined by edit distance



Below we provide an example of the worst translated sentence in our corpus. Correct translations are in green, incorrect are in red, and characters that were originally hiragana are in black.

Original

朝鮮通信使は日本にじゅうに回来て、多いときは学者や医者、画家など500人ぐらいが日本人と交流しました

Output

ちょうせんつうしんしはにほんにじゅうにかいらいて、おおいときはがくしゃやいしゃ、がかなどっぴゃくひとぐらいがにほんじんとこもごもながしました

Answer

ちょうせんつうしんしはにっぽんにじゅうにかいきて、おおいときはがくしゃやいしゃ、がかなどごひゃくにんぐらいがにっぽんじんとこうりゅうしました

It's worth considering that our test data came from the translations of a Japanese language news site. They don't say how they produce these translations, and it's possible that they aren't done by hand. Furthermore, news sites represent a specific style of speaking Japanese. It's possible that the algorithm would do better (or worse) on transcripts of spoken Japanese or works of fiction.