ISMLA Multilingual Session 4: Investigating Kanji Overlap in Subtitles

Johannes Dellert

Tübingen University

November 22, 2017

Plan

1. Hanzi vs. Kanji
2. Mutual Readability?
3. Experiment Setup
4. Exercise 03: Investigating Kanji Overlap

Chinese and Japanese: A shared writing system

- Japanese has been using Chinese characters ever since writing was introduced, and is the only non-Chinese language still doing so
- 2,136 characters are in frequent use in Japan (常用漢字 Jōyō kanji), and several hundred more are in common use (mainly for names)
- 3,500 characters are in frequent use in the PRC (现代汉语常用字表 Xiàndài Hànyǔ Chángyòng Zìbiǎo), and twice that number is commonly used (现代汉语通用字表 Xiàndài Hànyǔ Tōngyòng Zìbiǎo)
- Chinese expresses everything in Hanzi (as we have seen)
- Japanese has two syllabaries in addition:
  - Hiragana (ひらがな) for native endings and small words
  - Katakana (カタカナ) for foreign terms

Chinese and Japanese: A shared writing system

The shared Kanji provide a semantic bridge between unrelated languages:

手	te	shǒu	“hand”
月	tsuki	yuè	“moon”
山	yama	shān	“mountain”
体	karada	tǐ	“body”
力	chikara	lì	“power”

In addition, Japanese has borrowed many **Chinese compounds**:

世界	sekai	shìjiè	“world”
自然	shizen	zìrán	“nature”
明白	meihaku	míngbai	“obvious”
注意	chūi	zhùyì	“attention”
漢字	kanji	hànzì	“Chinese characters”

Issue: Simplification

The overlap is drastically reduced by divergent simplifications:

PRC	Taiwan	Japan	meaning
东	東	東	“east”
头	頭	頭	“head”
体	體	体	“body”
国	國	国	“country”
假	假	仮	“fake”
乘	乘	乗	“multiplication”
关	關	関	“to close”
处	處	処	“place”

Out of the 2,136 Jōyō kanji,

- only 688 are identical in the PRC and Japan,
- only 844 are identical in Taiwan and Japan.

Issue: Different Choices for Simple Concepts

In addition, many simple concepts are expressed with different characters due to more than a thousand years of semantic change:

Japanese	Chinese	meaning
村 mura	乡 xiāng	"village"
川 kawa	河 hé	"river"
町 machi	城 chéng	"town"
赤 aka	红 hóng	"red"
丸 maru	圆 yuán	"circle"
姉 ane	姐 jiě	"elder sister"

Are enough Kanji shared for understanding?

For some sentences, the Kanji overlap is rather high:

JPN	魔法	大臣	死んだ	。	
phon	mahō	daijin	shi-nda		
gloss	magic	minister	die-PRF		
CMN	魔法	部长	死	了	。
phon	mófǎ	bùzhǎng	sǐ	le	
gloss	magic	minister	dead	[change]	
ENG	"The minister of magic is dead."				

This sentence is perfectly mutually readable. (Is it?)

Are enough Kanji shared for understanding?

For others, there is no overlap whatsoever:

JPN	杖	が	私	を	選んだ	わ	。
phon	tsue	ga	watashi	o	era-nda	wa	
gloss	stick	SBJ	1SG	OBJ	choose-PRF	[emotion]	
CMN	它	选择	了	我	。		
phon	tā	xuǎnzé	le	wǒ			
gloss	it	choose	[perfective]	1SG			
ENG	"It chose me."						

No monolingual reader will understand the other version. (Or not?)

This is just a first impression; we want a quantitative answer!

Open Subtitles

Idea: use movie subtitles to investigate Kanji overlap!

- a massive freely available parallel corpus of fan-made subtitles (thousands of movies across dozens of languages)
- attribution: `http://www.opensubtitles.org/`
- rather close to everyday language,
  derived frequency information is very informative of oral usage
- we can mechanically compute e.g. the word overlap in different movies to find movies about similar topics
- Q: Do the Kanji occurring in subtitles characterize the movies across the (Simplified) Chinese and Japanese language versions?

Exercise 03: Our Testset

We prepared roughly aligned parallel subtitles for the following very popular 2010 movies:

- `ALI`: Alice in Wonderland
- `HAR`: Harry Potter and the Deathly Hallows Part 1
- `DES`: Despicable Me
- `TRO`: Tron: Legacy

Question: Can the shared Kanji help us to tell which is which?

Exercise 03: Movie Overlap

If we count characters, not compounds, this gives us the following matrix:

cmn\ jpn	ALI	HAR	DES	TRO
ALI	**907**	**1058**	750	959
HAR	840	**1478**	796	1071
DES	720	**940**	**826**	889
TRO	738	1113	752	**1138**

Exercise 03: Measure of Movie Separation

Ad-hoc measure 扱 (ancient character, means "collect") of how well movies are cross-linguistically identified by Kanji overlap, inspired by χ^2 statistic:

- let c_{ij} be the summed token overlap count for movie i in Chinese and movie j in Japanese (occurrences, one Kanji can count multiple times!)
- let n be the sum of all c_{ij} (= the total number of overlaps we counted)
- $E_{ij} := (\sum_k c_{kj} + \sum_l c_{il})/n$, the expected overlap if Kanji were random
- then, summarize the diagonal and off-diagonal entries as two observations, and compute a χ^2-like measure of dependence:

$$扱 := \frac{\left(\sum_{i=j}(E_{ij}-c_{ij})\right)^2}{\sum_{i=j} E_{ij}} + \frac{\left(\sum_{i\neq j}(E_{ij}-c_{ij})\right)^2}{\sum_{i\neq j} E_{ij}}$$

Exercise 03: Measure of Movie Separation

Illustration of 扱 computation in the example:

cmn\ jpn	ALI	HAR	DES	TRO	Σ
ALI	**907**	**1058**	750	959	3674
HAR	840	**1478**	796	1071	4185
DES	720	**940**	**826**	889	3375
TRO	738	1113	752	**1138**	3741
Σ	3205	4589	3124	4057	14975

Exercise 03: Measure of Movie Separation

Observed overlaps:

cmn\ jpn	ALI	HAR	DES	TRO
ALI	907	1058	750	959
HAR	840	1478	796	1071
DES	720	940	826	889
TRO	738	1113	752	1138

Expected overlaps:

cmn\ jpn	ALI	HAR	DES	TRO
ALI	786	1126	766	995
HAR	896	1282	873	1134
DES	722	1034	704	914
TRO	801	1146	780	1014

Exercise 03: Measure of Movie Separation

Observed vs. Expected overlaps:

cmn\ jpn	ALI	HAR	DES	TRO
ALI	907 - 786	1058 - 1126	750 - 766	959 - 995
HAR	840 - 896	1478 - 1282	796 - 873	1071 - 1134
DES	720 - 722	940 - 1034	826 - 704	889 - 914
TRO	738 - 801	1113 - 1146	752 - 780	1138 - 1014

Add the squares of the red entries, divide them by the sum of red entries.
Same for the blue entries, add both numbers together ⇒ 扱

Most frequent characters in subtitles

Extracted from the entire open subtitles corpus by Tiedemann (2009):

cmn	meaning	jpn	meaning
我	I	い	[i]/ADJ
的	REL	の	[no]/GEN
你	you	な	[na]/ADJ
是	to be	た	[ta]/PST
了	PRF	て	[te]/PROG
不	not	は	[ha]/TOP
一	one	に	[ni]/LOC
们	PL	る	[ru]/INF
这	this	だ	[da]/is
他	he	し	[shi]/do-

Most frequent Hanzi in subtitles

cmn	meaning	jpn	meaning
我	I	私	I
的	REL	何	what
你	you	彼	he
是	to be	人	person
了	PRF	見	see
不	not	事	thing
一	one	行	go
们	PL	前	in front
这	this	言	say
他	he	分	part/minute

Frequent Hanzi: comparison

cmn	meaning	jpn rank	meaning
我	I	163	(part of) we
的	REL	180	-ish
你	you	1679	thou [archaic]
是	to be	1360	justice
了	PRF	471	end
不	not	220	un-
一	one	82	one/[length sign]
们	PL	2277	[uncommon]
这	this	–	–
他	he	234	other

Frequent Kanji: comparison

jpn	meaning	cmn rank	meaning
私	I	907	private
何	what	166	which
彼	he	1032	other
人	person	15	person
見	see	–	(became simplified to 见)
事	thing	43	matter
行	go	103	walk/OK
前	in front	111	in front
言	say	538	speak
分	part/minute	183	part

Exercise 03: Measure of Movie Separation

Your tasks:

- implement the 扱 measure as the output of a UIMAFit pipeline
- investigate the overlap, look up the relevant characters on the Wiktionary, and build an opinion of how well it works
- experiment with various cutoffs (ignoring the k most frequent Hanzi in Chinese and the l most frequent characters in Japanese) to find out how many of the most frequent characters we need to ignore to optimize the separation measure
- repeat the experiment with binary Kanji compounds ($\approx$ loans)

Exercise 03: The Frequency Dictionary

Frequencies (and two-kanji compounds) are accessible as TSV files which can be used to build instances of our `FrequencyDictionary` class

- initialization:
  `new FrequencyDictionary(InputStream tokenFrequencyFileAsStream, InputStream charFrequencyFileAsStream)`
- `characterLookup(String orthForm)` returns character frequency
- `compoundLookup(String orthForm)` returns compound frequency

Return type `FrequencyDictionaryEntry` contains the following fields:

- `lemma`: the orthographic form
- `logPercentage`: the logarithm of the percentage of the form in the subtitles corpus (not relevant for this exercise, but very interesting)
- `rank`: rank of the respective form either in the character or compound list defined by the files fed to the constructor

Exercise 03: The Frequency Dictionary

You have two options to get the `InputStream` objects necessary to construct the `FrequencyDictionary`:

1. extract the files from the JAR (they are within the source hierarchy, and are packaged directly next to the `FrequencyDictionary`) after extraction to a location of your choice, you can build `File` objects and then `FileInputStream` objects
2. leave the files inside the JAR, and retrieve the resources as streams (providing this option is the motivation for `InputStream` arguments) `FrequencyDictionary.class.getResourceAsStream(` *fileName* `)`

The relevant filenames in both cases are:

- `opensubtitles-freq-tokens-cmn.tsv`
- `opensubtitles-freq-chars-cmn.tsv`
- `opensubtitles-freq-tokens-jpn.tsv`
- `opensubtitles-freq-chars-jpn.tsv`

Exercise 03: The Subtitles Collection Reader

We provide a ParallelSubtitlesReader which builds multi-view JCases from the aligned subtitle files (the TSV files in the same JAR):

- only parameter `"dataDirName"`: to tell it where the TSVs are
- creates a JCas for each movie which contains two views with names `"cmn"` and `"jpn"`, which contain the Chinese and Japanese versions of the subtitles for that movie as document texts

Usage:

- do not create an instance, but add it to the beginning of the pipeline
- your components will retrieve the language version using `cas.getView("cmn")` and `cas.getView("jpn")`

Exercise 03: Thresholded Frequency Annotation

To process the subtitles, you need to write a configurable analysis engine:

- your type system only needs an annotation type with an integer field for the rank of the respective kanji (compound)
- task: annotate kanji (compounds) above a certain frequency rank
- make your annotator configurable by
    - a language parameter (`"cmn"` or `"jpn"`), telling it which view to process
    - path(s) to the frequency files necessary to initialize the `FrequencyDictionary` objects
    - a minimum rank r_{min}, it should only build annotations for tokens with frequency rank $\geq r_{min}$
- for kanji compounds (the last task), you need to either have a parameter for specifying optional greediness, or write a second variant of your annotator that calls `compoundLookup(String orthForm)` for chunks of length 2

Exercise 03: Setting up the Pipeline

Use your knowledge of UIMAFit to set up a pipeline consisting of the following components:

- the `ParallelSubtitlesReader` we are providing (configured to your data directory)
- one instance of your `CharFrequencyAnnotator` configured to operate on the Chinese view
- one instance of your `CharFrequencyAnnotator` configured to operate on the Japanese view

At this stage, you might want to do some debugging by printing out parts of the annotation and validating a small part of the results against the raw data.

Exercise 03: Extracting the Statistic

After annotating the kanji of a certain frequency:

- get access to the individual `JCases` for each movie by calling the pipeline using `SimplePipeline.iteratePipeline(...)`, this gives you an iterator over the `JCas` objects for each movie
- store the kanji (compounds) counts for each movie in a data structure that allows you to compare the counts for different movies
- implement a method for computing the c_{ij} values out of the stored counts
- for both rank thresholds at zero, ensure that the result table makes sense (you can compare it to the example table on the slides)
- implement the computation of the 扱 measure

Exercise 03: Investigating the Overlap

To understand what is happening, we need to inspect the actual overlap:

- take a look at the ten most frequent shared Kanji for two language versions of an arbitary movie, and two versions of different movies (i.e. one diagonal and one off-diagonal table cell)
- look up the meaning of each Kanji in the Wiktionary
- do the meanings tell you anything about the movies involved?
- argue why this finding implies using a threshold value; which concept of computational linguistics corresponds to using such a threshold?

Exercise 03: Setting up the Experiments

To set up the experiment with the threshold values,

- wrap two loops around your pipeline and statistics code, modifying separate lookups for Chinese and Japanese steps in steps of 100 from 0 to 2,000 (this is just the minimal setup, you can do more complex things if you want)
- in this way, find the combination of thresholds which maximizes 扱
- what are the most salient Kanji for each movie at this threshold? do they make more sense semantically?

Finally,

- build an alternative pipeline which only annotates compounds
- repeat the entire experiment (including determining an optimal combination of thresholds)

Exercise 03: Questions

Questions?