TDU 東京電機大学

An Experimental Evaluation of Japanese Tokenizers for Sentiment-Based Text Classification

Andre Rusli, Makoto Shishido
Graduate School of Advanced Science and Technology
Tokyo Denki University

@ NLP 2021

Agenda

Introduction
- Backgrounds
- Related works
- Research objectives

Experimental setup
- Dataset
- System specification
- Methods

Results and analysis
- Tokenization
- Vectorization
- Classification

Conclusion and future works

Introduction

Backgrounds

Machine Learning + Text Classification

- Research works on languages with less resource are still considerably low
- Challenges on non-alphabetic language

Japanese scripts pose some challenges, such as:

- No whitespace between words
- Word combinations in a sentence could vary and have different meanings

Morphological analysis tools for word-based tokenization

- For text classification, word n-grams is often more preferable than character n-grams
- Several existing MA tools: MeCab, Juman, KyTea, GiNZA, Sudachi, SentencePiece

Related Works

Japanese morphological analysis tools

MeCab (Kudo et al., 2004)	Not exactly new but one of the most **popular** and **fast** tools for Japanese MA (also available for other languages, such as Korean).
Sudachi (Takaoka et al., 2018)	**Continuous maintenance** and **feature richness**, focus to support business use. Also used by spaCy.
SentencePiece (Kudo & Richardson, 2018)	**Language-independent** subword tokenizer and detokenizer designed for Neural-based text processing. Also used by XLM-RoBERTa.

Text classification algorithm: Multinomial Naïve Bayes and Logistic Regression

- Baseline models for text classification (Bataa & Wu, 2019; Zhang & LeCun, 2017; Sun et al., 2018;)
- Ongoing improvements (Qu et al., 2018)

Research Objectives

1. Experiment with tokenizers provided by MeCab, Sudachi, and SentencePiece
2. Implement the above tokenization tools for binary sentiment-based text classification using TF-IDF with Multinomial Naïve Bayes and Logistic Regression
3. Evaluate the performance results in terms of elapsed time and error percentages

Experimental Setup

Dataset

We use the binary sentiment label and review text of a Japanese Rakuten product review sentiment dataset[1]

10% of the original data: **340,000** reviews for training and **40,000** reviews for testing

Sentiment Label	Review Text
Negative (0)	余りにも、匂いがきつく安物み たいです。\\n 安いから仕方ないかな?
Positive (1)	毎回利用しています。納品が早 いし何よりお安く大変便利です。また利用し ます

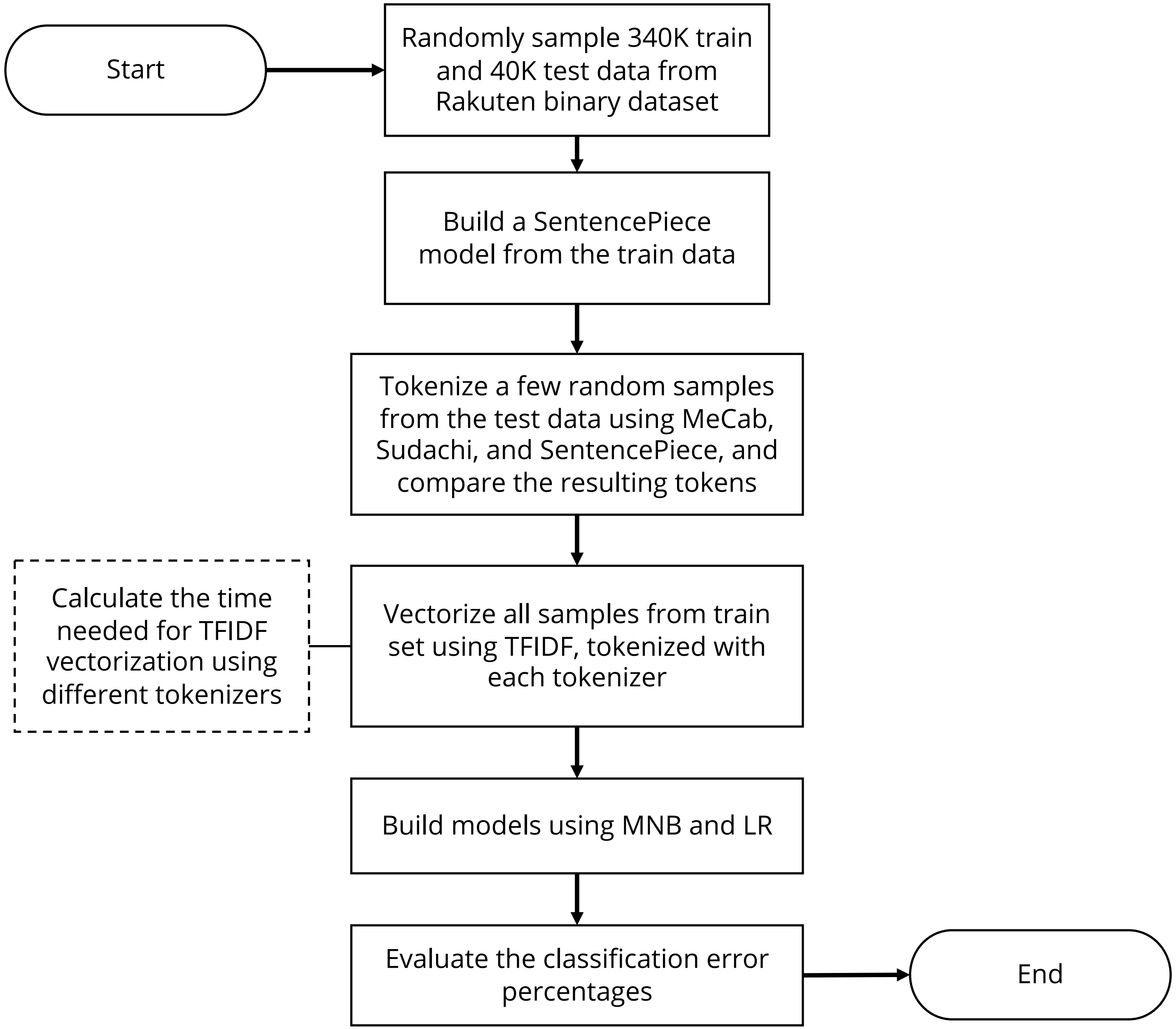
[1]X. Zhang and Y. LeCun, "Which Encoding is the Best for Text Classification in Chinese, English, Japanese and Korean?," arXiv:1708.02657v2 [cs.CL], 2017.

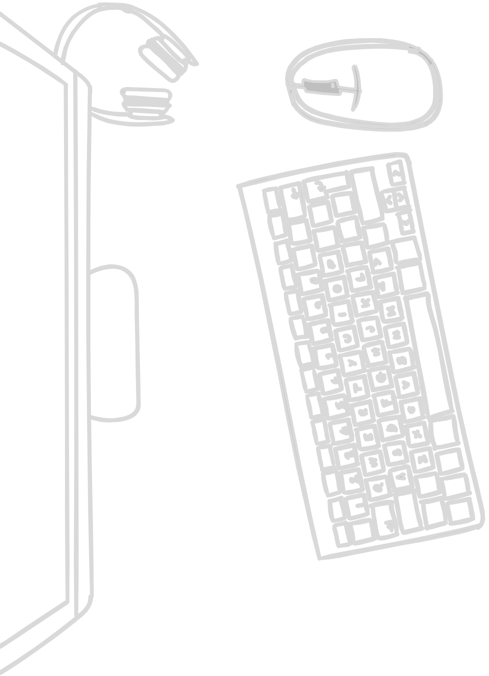
System Specifications

Python 3.7 and Jupyter Notebook on Google Compute Engine:

- Ubuntu virtual machine
- Machine type c2-standard-4
- 4 vCPUs and 16 GB memory

Methods


Start
Randomly sample 340K train and 40K test data from Rakuten binary dataset
Build a SentencePiece model from the train data
Tokenize a few random samples from the test data using MeCab, Sudachi, and SentencePiece, and compare the resulting tokens
Calculate the time needed for TFIDF vectorization using different tokenizers
Vectorize all samples from train set using TFIDF, tokenized with each tokenizer
Build models using MNB and LR
Evaluate the classification error percentages
End



Results and Analysis

Tokenization

Tokenize a few random samples from the test data using MeCab, Sudachi, and SentencePiece, and compare the resulting tokens

Original text:

自転車通勤用に購入。サックス を選びましたが、…

MeCab:

自転、車、通勤、用、に、購入、。、サックス、を、選び、まし、た、が、…

Sudachi (SplitMode.B):

自転車、通勤用、に、購入、。、サックス、を、選び、まし、た、が、…

SentencePiece (vocab size=32K):

_、自転車通勤、用に購入、。、サックス、を選びましたが、…

Tokenization

Tokenize a few random samples from the test data using MeCab, Sudachi, and SentencePiece, and compare the resulting tokens

Original text:

かわいいです(*^。^*)\n パソコン...

MeCab:

かわいい、です、(、*^。^*)\、n、パソコン、...

Sudachi (SplitMode.B):

かわいい、です、(*^。^*)、\\、n、パソコン、 ...

SentencePiece (vocab size=32K):

_、かわいいです、(*^。^*) 、 \\ 、 n 、パソコン、...

Vectorization

Calculate the time needed for TFIDF vectorization using different tokenizers

Vectorize all samples from train set using TFIDF, tokenized with each tokenizer

Elapsed time to vectorize the training data (340,000 review texts) using **TF-IDF and each tokenizer:**

Tokenizer	Elapsed Time (seconds)
MeCab	34.65
Sudachi	1,533.58
SentencePiece	25.84

Classification

Build models using MNB and LR

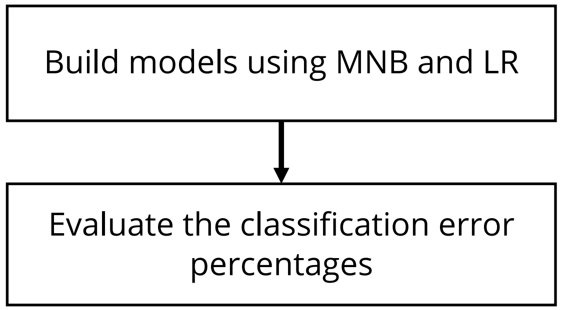
↓

Evaluate the classification error percentages

We then train two models using MNB and LR using vectorized TF-IDF values for each tokenizer, from the previous steps. Results are as follow.

Tokenizer	Classifier	Error-Train (340K)	Error-Test (40K)
MeCab	Logistic Regression	8.52	9.63
Sudachi	Logistic Regression	8.5	9.59
SentencePiece	Logistic Regression	6.54	8.02
MeCab	M. Naïve Bayes	11.24	12.52
Sudachi	M. Naïve Bayes	11.04	12.42
SentencePiece	M. Naïve Bayes	8.28	8.9

Classification



Based on the best performing combination in the previous slide, we then perform a **hyperparameter tuning** process for Logistic Regression using grid search and repeated stratified k-fold cross validator from Scikit-learn.

Tokenizer	Classifier	Error-Train (340K)	Error-Test (40K)
SentencePiece	Logistic Regression (default)	6.54	8.02
SentencePiece	Logistic Regression (C=10, penalty='l2', solver='lbfgs')	5.56	7.78

Conclusion and Future Works

Conclusion

- The generated tokens from Sudachi are more likely to match dictionary results and common words understood by human, however, MeCab and SentencePiece are significantly faster

- Even though tokens generated by SentencePiece are limited to its training data and might not match common dictionary results, they perform better for our dataset, which is a binary sentiment-based text classification task

- The combination of SentencePiece, TF-IDF, and Logistic Regression achieved the best performance with 5.56 training error percentage and 7.78 testing error percentage.

Future Works

- Experiment with various n-gram configurations and other hyperparameters
- Use multi-class and bigger datasets
- Train multi-lingual models
- Experiment with various shallow and deep learning approaches.

Thank you
ご静聴ありがとうございました