Building a Name Masking System: From Dataset to Model

Tianqi Wang[1] Yoichiro Ogawa[1] Kazumasa Saito[1]
[1]Classi Corp.
{tianqi.wang,youichiro.ogawa}@classi.jp
kazumasa.saitoh@gmail.com

Abstract

Data is becoming a more and more crucial resource nowadays. Machine learning techniques are rapidly developed based on the analysis and leverage of Big Data. Data is collected, stored, and processed in research and product environments. However, there is a downside that the data may contain privacy information, making it a risk to process the raw data. In this paper, we are trying to extract and mask persons' names from a dataset before any further processing. We create our dataset based on KWDLC dataset and propose a simple but efficient model to match names in texts. The model trained with Japanese name dataset is able to match names in the forms of Kanji, Katakana, Hiragana and Romaji from sentences and phrases.

1 Introduction

Data is becoming a more and more important resource, especially for machine learning techniques and artificial intelligence. However, the privacy information potentially contained by the dataset is becoming a challenging risk [1, 2]. For Classi, we provide assistance systems to schools to help supervisors with daily work, including online education and student management. Analysis of data produced by users such as answers submitted by students is essential to monitor the quality of our services. However, the privacy information(such as addresses, names, student numbers, etc.) potentially contained by the data make it a risky task. Thus, a system to mask privacy information is necessary. There are generally two challenges making this task difficult. Firstly, the dataset for privacy information, especially in Japanese, is scarce. Thus it is difficult to train or fine-tune a model. Secondly, the existing model does not meet the request, especially for name masking.

In this paper, we create a dataset containing Japanese names based on an existing dataset. We then build a model to detect names from natural language text and train the model with the proposal dataset. The experimental results indicate that our proposed model outperforms the existing NER model, and is able to process names in various forms.

Full name	Hiragana	Romaji
菅原雅俊	すがわらまさとし	sugawaramasatoshi
藤井麗	ふじいれい	fujiirei
島博文	おおしまひろふみ	oshimahirofumi

Table 1 Some examples of collected names. The katakana is omitted for the limited space.

2 Related Work

The research on privacy-preserving has been widely studied by the community. Generally speaking, there are two approaches, naming anonymization and randomization. Anonymization is a practical solution for preserving users' privacy in data publishing[3]. Data owners such as hospitals, banks, social network (SN) service providers, and insurance companies anonymize their user's data before publishing it to protect the privacy of users whereas anonymous data remains useful for legitimate information consumers. Randomization refers to methods that perturb data by multiplicative and addive noise or other randomized processes[4].

In our case, the most important privacy information is names, which is possible to be detected by NER models. Named entity recognition (NER) is the task to identify mentions of rigid designators from text belonging to predefined semantic types such as person, location, organization etc[5]. In this paper, we take the pre-trained GiNZA v5[6] as the baseline. GiNZA is a Japanese NLP library based on Universal Dependencies[7], pre-trained with several datasets for different tasks. The named entity recognition model of GiNZA is trained on a part of GSK2014-A (2019) BCCWJ edition[8].

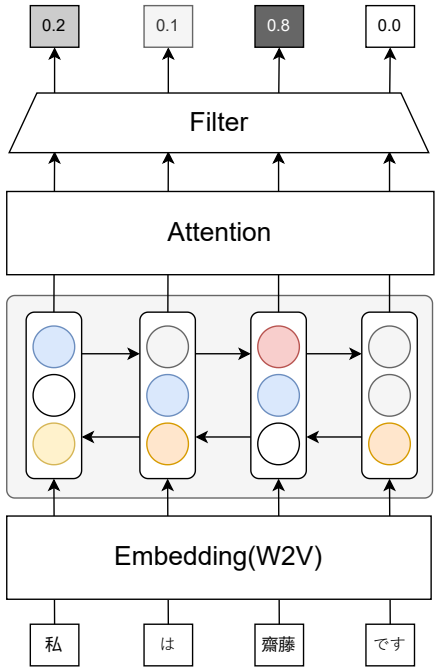

Figure 1 The architecture of proposal model.

3 Proposal

3.1 Dataset

There is not a dataset for name-masking task according to our knowledge, thus we build the dataset by inserting names into an existing public dataset.

Collect names We collect Japanese names from a website[1)] providing dummy data of privacy information. There are other types of privacy information such as addresses, credit card numbers, etc. available, but we only use the data of names in this research. The data is provided as full names. Consider the different diversity of family names and given names, the difficulty may also change. Thus we split the names to family names and given names, and organize the data in forms of Kanji, Hiragana, Katakana and Romaji respectively. Some examples of collected names are shown in Table 1 As a result, we have three dataset for full names, family names and given names respectively, and 300 data for each of them.

Insert names We build the dataset by inserting names into the KWDLC corpus[9]. The KWDLC corpus contains documents with various genres and styles, such as news articles, encyclopedic articles, blogs and commercial pages. The linguistic annotations consist of annotations of morphology, named entities, dependencies, etc. We make use of the named entities' annotation in our research.

We extract only the data with names from KWDLS corpus, and remove data where the names do not refer to persons such as **薄竹善道**事務所 (Office of **Usutake Yoshimiti**). Besides, there are texts that are the same after removing name tokens. We remove these data as well to avoid repeated data. As a result, we extracted 798 texts containing names in total. We then insert the names we obtained earlier to the corresponding position of the texts based on the annotations of named entities provided by the KWDLC corpus. All of the names are inserted into the text equally.

We assign binary labels to each character and token as the ground truth. Labels to characters and tokens contained by names are set to 1 while labels to others are set to 0. The labels are used for training and evaluating models in the following experiments.

3.2 Proposed Model

We name our proposed model as NameTagger(NT) in this paper. As illustrated in Figure 1, the model consists of four layers. First, the embedding layer converts tokens to word embeddings $\mathbf{e}_i \in \mathbb{R}^d$. Then we feed the word embeddings to a Bi-LSTM layer to produce contextualized hidden states $\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}]$ for each words. Note that the forward ($\overrightarrow{\mathbf{h}_i}$) and backward ($\overleftarrow{\mathbf{h}_i}$) hidden states are concatenated as the output of Bi-LSTM layer. We set the dropout rate of Bi-LSTM layer to 0.5. The attention layer assign attention values to tokens as probability that the token belongs to a name. The attention values α_i to the i^{th} token are calculated as follows:

$$\alpha_i = \frac{1}{1 + e^{\mathbf{h}_i \mathbf{M} + \beta}}, 0 \leq \alpha_i \leq 1 \quad (1)$$

where $\mathbf{h} \in \mathbb{R}^r$ is the hidden state of the i^{th} token, $\mathbf{M} \in \mathbb{R}^{r \times 1}$ and $\beta \in \mathbb{R}$ are parameters to learn. Finally, we filter out tokens that are clearly non-names, such as numbers and punctuation.

4 Experiments and Discussion

We divide the dataset to training data (80%), dev data(10%) and test data (10%), and make sure that both the text and names are different between each subset of the dataset, so that all of the input data is new to the model in the test phase. We tokenize the input data to tokens

1) https://testdata.userlocal.jp/

by MeCab[10] with IPA dictionary. We use the word2vec model pre-trained with Japanese Wikipedia, distributed by the Inui Lab at Tohoku University[2], to initialize the embedding layer. The word embedding dimension is $d = 100$. Considering the limited size of training data, we freeze the embedding layer during training phase. The dimension of Bi-LSTM is set to 250, so the dimension of hidden states is $r = 500$ after concatenation.

The performance is evaluated on character level. To evaluate the performance, We first binarize the predicts of our model based on a threshold value:

$$p_i = \begin{cases} 1, & \alpha_i > T \\ 0, & otherwise \end{cases} \tag{2}$$

where p_i is the predicted label to the i^{th} token, and $T = 0.7$ is a hyper-parameter. We then calculate recall, precision and F1 scores between the predicted labels and correct labels.

We train the model for 20 epochs with a learning rate of 0.001, and take Cross-Entropy as the objective function. We save the learned parameters when the model achieves the best performance on dev data, and the final performance is evaluated with the test data.

We explore the following topics in our experiments. We first test the performance of the pre-trained GiNZA model on our dataset, and discuss the possible reason limiting the performance. We then test the performance of our proposed model. Furthermore, We explore various settings to train our model, and discuss the effect on processing different name forms.

4.1 Performance of GiNZA

We evaluate the performance of GiNZA in detecting names from our dataset in forms of Kanji, Katakana, Hiragana and RomaJi respectively. Table 2 shows the results.

It is easy to tell that the performance of GiNZA is highly related to the names forms. GiNZA performs pretty well on names in Kanji, but almost does not work for names in Katakana and RomaJi. It is reasonable because Japanese names are in the forms of Kanji and Hiragana in most cases, and is more possible contained in the training data used by GiNZA. Considering the performance irrelevant to the frequency of name forms, we assume that the pre-trained model relies on vocabulary of names.

2) http://www.cl.ecei.tohoku.ac.jp/~m-suzuki/jawiki_vector/

		Kanji	Hiragana	Katakana	Romaji
Precision	FLN	0.907	0.683	0.763	0.000
	FN	0.811	0.786	0.248	0.096
	GN	0.830	0.846	0.824	0.33
Recall	FLN	0.962	0.918	0.195	0.000
	FN	0.928	0.403	0.682	0.601
	GN	0.971	0.746	0.572	0.33
F1	FLN	0.934	0.783	0.311	0.000
	FN	0.866	0.533	0.364	0.165
	GN	0.895	0.794	0.676	0.61

Table 2 The performance of GiNZA on full names (FLN), family names (FN) and given names (GN) respectively.

4.2 Performance of Proposed Model

We train our model with the dataset introduced in Section 3.1. The model is trained for different name forms independently. The comparison between F1 scores produced by GiNZA and our proposed model(NT) is shown in Table 4. Note that because the pre-trained GiNZA model is not fine-tuned with the dataset, the comparison is only for reference. Our model outperforms the pre-trained GiNZA model on all name forms, especially on non-Kanji forms. The performance on names in Romaji is even better than names in Hiragana and Katakana. It indicates that the model can be overfitted. Because names in Romaji are unknown tokens to the pre-trained word embeddings, the model may prefer to detect all of the unknown tokens as names. To inspect the hypothesis, we train the model with names in Kanji, and apply it to names in other forms. The results are shown as NT(Kanji) at Table 4. The performance on different name forms supports our hypothesis. Because the non-Kanji name tokens are unseen by the model during the training phase, the model barely matches the names correctly. Especially when the names in Romaji are unknown to the model, namely no semantic information is available, the model can not detect the names totally. In order to train one model and apply it to various name forms, we explore two approaches to solve this issue.

4.3 Training with Mixed Name Forms

To obtain a model capable of matching various name forms, it is natural to train the model with multiple name forms instead of single ones. To achieve this goal, we build the data set with mixed name forms.

When inserting names into texts, we insert all four forms

Text	NT	NT(Mix)
１０周年記念特別号前編を担当致します inuiizumi と申します。	１０, inuiizumi	inuiizumi
１０周年記念特別号前編を担当致しますスズキユキと申します。	１０, スズキユキ	スズキユキ
紙パックの飲み物であり、otsukahideo の好物。	紙パック, otsukahideo	紙パック, otsukahideo
紙パックの飲み物であり、マツモトシゲハルの好物。	紙パック, マツモトシゲハル	紙パック, マツモトシゲハル

Table 3 Example of names extracted by NT and NT(Mix) settings.

	Kanji	Hiragana	Katakana	Romaji
GiNZA	0.883	0.348	0.217	0.145
NT	0.993	0.933	0.900	0.955
NT(Kanji)	0.993	0.620	0.188	0.000
NT(Mix)	0.954	0.923	0.932	0.965
NT(Random)	0.467	0.608	0.665	0.795

Table 4 Performance of proposed model trained with various settings. NT is trained on and applied to datasets of each name forms independently. NT(Kanji) is trained with names in Kanji and applied to each name forms. NT(Mix) is trained with mixed name forms and applied to each name forms. NT(Random) is trained with randomly selected name tokens and applied to each name forms..

to generate the dataset. Namely we created four data for each text. The performance of the model trained with mixed name forms is shown as NT(Mix) in Table 4. The F1 score is close to models trained for each name forms independently, and much better than the setting of NT(Kanji). That indicates that training with mixed name forms clearly improves the performance. We noticed that the NT(Mix) performs even better than NT on Katakana and Romaji, which is out of our expectations, because for the NT setting the name forms are the same in training and test data, while the name forms are different for the NT(Mix) setting. Some further analysis shows that the improvement is from precision, meaning the NT(Mix) produces fewer false-positive results. Table 3 shows some examples of names extracted by the model. The first two examples show that compared to NT(Mix), NT is easier to incorrectly detect full-width numbers as names. As we mentioned above, the model may be overfitted when trained with names in Romaji, tending to predict unknown tokens as names. However, trained with mixed name forms, the model process unknown tokens better. At the same time, the 3rd and 4th examples in Table 3 show that the mixed name forms do not contribute to the processing on known tokens.

4.4 Training with Random Name Tokens

As we discussed in Section 4.1, one of the reason limiting the performance of GiNZA may be the dependence on name vocabulary. To train a model independent to name vocabulary, we insert random tokens instead of collected names to the texts to build the dataset. Given a text containing names, we randomly select a token from the text, and replace the name token with the random token. In this way, the selected token appears at least twice in one sentence, one as name, and the another one as non-name. By training the model with this dataset, we expect the model to learn the difference between name tokens and non-name tokens depending on the context instead of the token itself.

NT(Random) in Table 4 shows that the performance is limited overall. Considering the name tokens are randomly selected from the text, there are almost no name data in Katakana and Romaji, but the model works well on names in these two forms, especially compared to the performance of NT(Kanji) and GiNZA. We notice that the performance is limited by precision (0.343 - 0.709), while the recall scores are still high enough (0.718 - 0.905). It may caused by the semantic features of the pre-trained word embeddings. We consider to fine-tune the word embedding layer with larger dataset in the future.

5 Conclusion

With the popular technique based on Big Data, masking privacy information is becoming a crucial challenge task. In this paper, we build a system to detect names in Japanese from texts. We first build a dataset for the name masking task by inserting Japanese names into an existing dataset. The dataset covers names in various forms. We train a simple but efficient model with the dataset to detect Japanese names. The model outperforms the pre-trained NER model on the name-masking task. We also discuss various settings to train and apply the model. However, because of the scarcity of data, we could not evaluate the model with data in the real product context. After applying the model to the product environment, masked data in the real world will be available. Then the model can be better trained and evaluated in the future. We shall also train the model to mask privacy information other than names such as addresses etc. in the future.

References

[1] W Nicholson Price and I Glenn Cohen. Privacy in the age of medical big data. **Nature medicine**, Vol. 25, No. 1, pp. 37–43, 2019.

[2] Sangchul Park, Gina Jeehyun Choi, and Haksoo Ko. Information technology–based tracing strategy in response to covid-19 in south korea—privacy controversies. **Jama**, Vol. 323, No. 21, pp. 2129–2130, 2020.

[3] Abdul Majeed and Sungchang Lee. Anonymization techniques for privacy preserving data publishing: A comprehensive survey. **IEEE access**, Vol. 9, pp. 8512–8545, 2020.

[4] Rasim M Alguliyev, Ramiz M Aliguliyev, and Fargana J Abdullayeva. Privacy-preserving deep learning algorithm for big personal data analysis. **Journal of Industrial Information Integration**, Vol. 15, pp. 1–14, 2019.

[5] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. **IEEE Transactions on Knowledge and Data Engineering**, Vol. 34, No. 1, pp. 50–70, 2020.

[6] 松田寛. Ginza-universal dependencies による実用的日本語解析. 自然言語処理, Vol. 27, No. 3, pp. 695–701, 2020.

[7] Marie-Catherine De Marneffe, Christopher D Manning, Joakim Nivre, and Daniel Zeman. Universal dependencies. **Computational linguistics**, Vol. 47, No. 2, pp. 255–308, 2021.

[8] 橋本泰一, 乾孝司, 村上浩司ほか. 拡張固有表現タグ付きコーパスの構築. 情報処理学会研究報告自然言語処理 (NL), Vol. 2008, No. 113 (2008-NL-188), pp. 113–120, 2008.

[9] Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. Building a diverse document leads corpus annotated with semantic relations. In **Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation**, pp. 535–544, 2012.

[10] Taku Kudo. Mecab: Yet another part-of-speech and morphological analyzer. **http://mecab. sourceforge. net/**, 2005.