Lines betray you: Identifying character from lines

Anonymous ACL submission

Abstract

Will lines betray information of characters? The answer should be yes. In our work, we use discriminative model to classify characters of lines in *Bigbang* scenario. Our main contribution is to find out which feature is more important. We find that characters described in the scene information, characters before current lines and psycholinguistic features are main contributers o classifier. It means that these features of your speech or articles may betray you even if your name is anonymous.

1 Introduction

When we read novels, sometimes authors may conceal characters who say current lines. This may cause confusion sometimes. When we are blind to characters, through which we may guess correct character of current lines? If we feed them to a neural network, can machine learn characters as appropriate as human? With these questions, we start our work from *Bigbang* scenario. We want to know which feature will betray speakers, which means which feature influence most to identify character od current line. With this question we find which feature is more important and implement several state-of-the-art model to improve accuracy. Also, we implement some state-of-the-art models described in current paper. We also compare performance of these models.

2 Dataset Construction

Before we start our work, we need to construct dataset ourselves. Roughly, we have data like followings:

> [An example for *Scenario*.] *Scene*: Sheldon and Leonards apartment. Sheldon, Leonard, Howard and Raj are present.
> *Leonard*: There you go, Pad Thai, no peanuts.
> *Howard*: But does it have peanut oil?
> *Leonard*: Uh, Im not sure, everyone keep an eye on Howard in case he starts to swell up.
> *Sheldon*: Since its not bee season, you can have my epinephrine.
> *Raj*: Are there any chopsticks?

We can easily identify that each line has these background, character and scene. We may separate lines and characters who say current lines and provide them with current scenes. Second question comes to our mind is that how many characters do we have and how many lines, scenes and episodes belong to each character. In total, we have 44997 lines. For each character, ratio to total lines, total scenes and total episodes is shown in the Figure 1

We can see top 8 characters say more than 99 percent lines and appear in most of the scenes. Compared to main characters, other characters are less important. So we set all other characters as label other. It is also worth to mention that total characters are more than 300, which is too big and sparse to implement a classifier model. Therefore, we reduce character space to 9 dimensions.

3 Feature Selection

Now we can start our work. In this part we want wo introduce how we select features and how these feature works.

Scene We define two ways to extract scene information. In the first strategy we see it simply as a line. In the second strategy, we extract characters from scenes because we believe that characters mentioned in scenes are very likely to

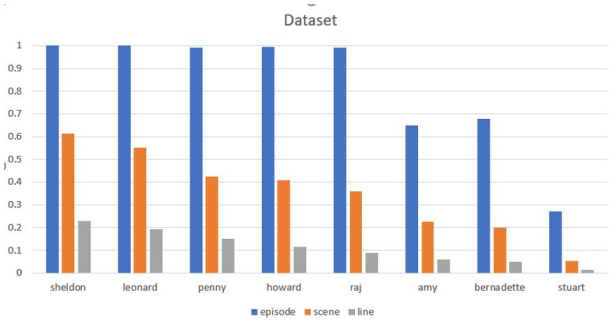

Figure 1: An example for *Dataset*.

be present in following lines. For example, raj in the scene rajs kitchen may be far more important than kitchen. In word representation, we see these two words as equal while in the character representation we highlight the character as a feature.

Lines Here we use Markov hypothesis to define line feature. We assume that the character of current line is closely related to the line before current line(before lines, BL) and of course current lines. We see lines as a feature and use word feature to extract line representation.

Word Feature we use well-trained word embedding and psycholinguistic features which can reflect personality traits. We believe that personality traits are far more important than context information provided by word embedding.

Line Feature In our model, we use LSTM layer to convert word embedding to line embedding. The math model of LSTM will be introduced in the next part.

Character Feature We define following encoding method to encode character feature. The first method is to use one hot encoding to encode nine characters. The second method is to use NN with activation "tanh" to extract feature from character instead of simple one-hot encoder.

Psycholinguistic Feature We use MRC to extract psycholinguistic feature. MRC is a psycholinguistic database which contains psychological and distributional information about words. We use 12 features in MRC by adding them to word embedding as new dimension and feed all of them to LSTM layer to train sentence embedding. The features we use are number of letters in the word (Nlet), number of phonemes in the word (Nphon), number of syllables in the word (Nsyl), Kucera and Francis written frequency (KF freq), Kucera and Francis number of categories (KF ncats), Kucera and Francis number of samples (KF nsamp), Thorndike-Lorge frequency (TL freq), Brown verbal frequency (BROWN freq), and Familiarity (Fam)[1]. MRC features used in previous studies showed that there is an important correlation between features and personality[4].

4 Model Architecture

4.1 Neural Network Model

As described above, we separate input as several parts, scene, lines before current lines(BF), current lines, before characters. They are fed to neural network separately and finally concatenate together. Finally, we use multilayer perceptrons (MLPs) to classify each line to its character. The architecture of our baseline model is shown in Figure 2

Except our baseline model, we also implement some state-of-the-art model, which are TextCNN[2], TextRNN[6], TextRCNN[3], TextRNN+Attention[5]. We will introduce these layers in the following chapters and also we will give these network architecture in following figures.

4.1.1 Embedding layer

We use word embedding to represent a word. For word embedding, we use the pre-trained word vectors from GloVe, which is trained on Wikipedia 2014 and Gigaword 5. Each word is represented by a 50-dimension vector.

To learn some psycholinguistic features, we use MRC to extract psycholinguistic features of words. Each word is represented by a 12-dimension vector.

We fuse two representation together and finally get a 62-dimension vector to represent each word

4.1.2 LSTM layer

We use both LSTM and bidirectional LSTM layer to extract features from word embeddings. Each of bidirectional LSTM is standard LSTM, the memory cell for the t-th feature has the following com-

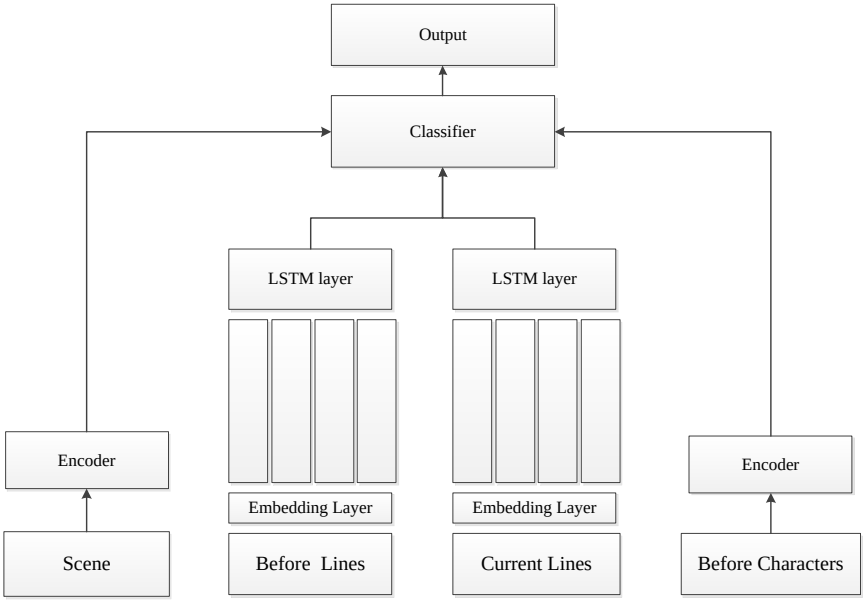

Figure 2: Baesline Model Neural Network Architecture

posite functions:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
j_t &= \sigma(W_{xj}x_t + W_{hj}h_{t-1} + b_j) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
o_t &= tanh(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
c_t &= c_{t-1} \odot f_t + i_t \odot j_t \\
h_t &= tanh(c_t) \odot o_t
\end{aligned} \tag{1}
$$

where σ is sigmoid function, i, f, o, c are the input gate, forget gate, output gate and cell activation vectors, j is for computing new c value, W, b are parameters for the LSTM layer.

4.1.3 Convolutional Layer

We use one dimension convolution to deal with texts. The advantage of using convolutional layer is that it is faster than LSTM and can also consider context information. The structure of 1D convolutional layer is:

$$y = Activation(\mathbf{W} \otimes \mathbf{X} + \mathbf{b}) \tag{2}$$

Here $\otimes$ means wide convolution calculation.

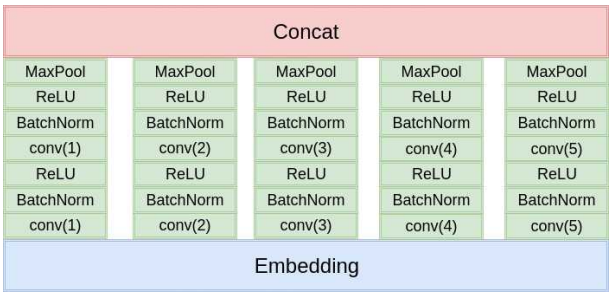

Figure 3: Structure of TextCNN

4.1.4 Pooling Layer

After convolution layer, we still have many features. To reduce feature dimension and choose the best feature, we use maximum pooling layer to select features. The 1D maxpooling layer can be described in the following form:

$$Y_m = \max_{i \in m} x_i \tag{3}$$

Here m denotes pool size.

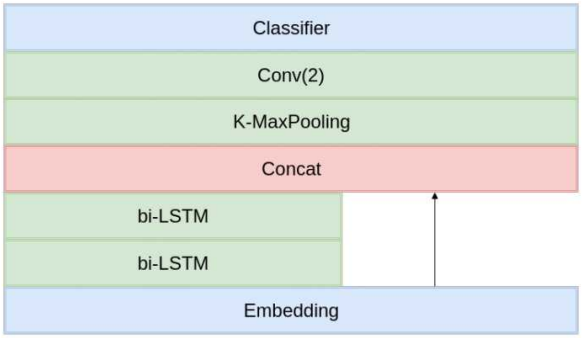

Figure 4: Structure of TextRCNN

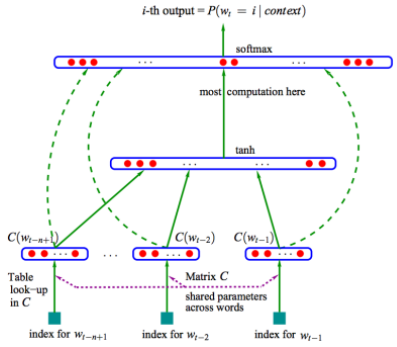

Figure 5: Structure of Attention Model

4.1.5 Concatenation Layer

Finally we concatenate these activations of the BiLSTM layers and generate the final output:

$$\sigma(W_o[h^1, h^2, h^3] + b_o) \tag{4}$$

where W_o and b_o are the parameters in this layer, and h^i is the i-th part of the model.

4.1.6 Attention Model

To deal with texts, we often have an attention mechanism to concentrate on most important parts of the texts and ignore less important context. The attention model we use in our model is a soft attention, which is:

$$s(x_i, q) = v^T tanh(Wx_i + Uq) \tag{5}$$

Also, we can use several layers to enhance the performance of attention model.

5 Experiment

5.1 Dataset

As stated above, for word embedding, we use the pre-trained word vectors from GloVe, which is trained on Wikipedia 2014 and Gigaword 5. Each word is represented by a 50-dimension vector. And we use MRC to get psycholinguistic features, the MRC can be found online. [1]

5.2 metric

To evaluate performance of the model and features. We use *accuracy* on test data to evaluate performance. First we choose the highest probability given by neural network as the predicted label. Then accuracy denotes that the ratio of right labels compared to all labels.

5.3 Feature Selection

To compare performance of each feature and each encoder, we feed them to our baseline model gradually to see how performance changes.Our baseline model uses only one perceptron layer to classify labels which is simple but still powerful. The results are shown in the Table 1

First, we define feature and strategy name as following:

Scene(S) we define two strategies: Simple Word Representation(SWR) or Character Encoding(CE) Lines Lines have two kinds of features: Before Lines(BL),Current Lines(CL).

Word Embedding(WE) We define two strategies: Nave Pre-trained Embedding(NPE) and Psycholinguistic Embedding(PE).

Characters(C) We define a single feature: Before Characters(BC(1)). The number in the bracket represents how many characters before current characters we use.

Character Embedding(CE) We define two strategies: One-hot Representation(OR) and MLP Encodere(MLPE).

Test Data Conctrction Also, we define two strategies to split dataset. The first is to Split by Episodes(SE) and the second is to Split by Scenes(SS). The first strategy uses first 8 seasons as training data and last one season as test data. The second strategy uses first 80% of a scene as training data and last one scene as test data.

5.3.1 Discussion

Simple pre-trained word embedding works rather poor in the first line of the table. This is easy to understand. Sometimes we say simple words like Hello or Nice to meet you. It is rather hard for even human beings to recognize who says these lines

[1] *websites.psychology.uwa.edu.au*

Features	Strategies	Accuracy
S+BL+CL	SWR+NPE+OR+SE	30.27
S+BL+CL	SWR+NPE+OR+SS	34.36
S+BL+CL+BC(1)	SWR+NPE+OR+SS	42.54
S+BL+CL+BC(3)	SWR+NPE+OR+SS	54.27
S+BL+CL+BC(3)	CE+NPE+OR+SS	62.29
S+BL+CL+BC(3)	CE+PE+OR+SS	66.65
S+BL+CL+BC(3)	CE+PE+MLPE+SS	**67.91**

Table 1: Performance of different features and strategies

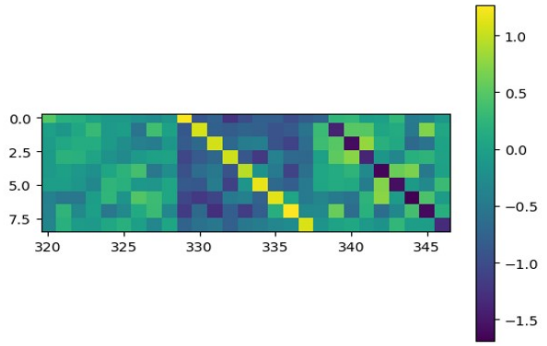

Figure 6: Weights of perceptrons on character features

since they happen everywhere. Compared to this weak feature, we can see Before Characters tend to betray speakers since in a scene, we often have dialogue or conversations between several certain persons. And one person cant say two continuous lines. To visualize this feature, we extract single perceptron layers of the forth model in Table 1 . The weights are shown in Figure 6

From left to right the weights are third, second and first character before current line. In the first character area, the diagonal of weights is minus, which denotes last character wont be the character of this line. This corresponds to our intuition. Similarly, in the second character before current line area weights are large in diagonal, which denotes that conversation often happens between two characters. This is rather useful for predicting characters.

The encoding of character is rather useful compared to one-hot encoding. Since one-hot encoding is too sparse, using neural network to encode it makes the input less sparse and improve performance.

It is worth to mention the way to separate train and test data. Separating by episodes is not reasonable since it is hard even for a person to guess character without any context. So separating by scenes is more reasonable and works better.

To sum up, we can conclude following features are prone to betray you(order is ranked): Before Characters, Characters in current scene, Lines with psychological features.

5.4 Neural Network Models

In our paper, we implement several state-of-the-art model to compare performance of each model. The models are described below: Baseline Model(LSTM + single perceptron), LSTM + Multi Layer Perceptrons(MLP), TextCNN(CNN + MLP), TextRNN(BiLSTM + MLP), TextRCNN(BiLSTM+CNN+MLP), Attention(LSTM+Attention+MLP). The features we use are the last one in Table 1. The results of the experiment are shown in Table 2

Feature Extraction	Classifier	Accuracy
Baseline	SP	67.91
Baseline	MLP	69.39
TextCNN	MLP	67.05
TextRNN	MLP	70.10
TextRCNN	MLP	69.75
BiLSTM+Attention	MLP	**71.35**

Table 2: Performance of different models

For text classification, LSTM works far better than CNN since CNN aggregates features simply by flatten layer but LSTM aggregates them together. Also we can see the deeper the classifier is the better the performance of the model is. This is intuitive because deeper neural network can contain more information. It is worth to note that BiLSTM works better than LSTM since they consider text with two sequence, both proper order and return order. Finally, Attention Model can make neural network concentrate on what may

contribute more to classification task. So we finally get 71.35 percent on our best model.

From this section, we can see that model is not the most important part. Model cant betray you, since different models can only have small influence on same task. Identifying features is the most important part. Even the construction of dataset can be much more difficult than accumulating models. So in future work, we may consider more about features or how to use features efficiently instead of models.

References

[1] Pierre Hadrien Arnoux, Anbang Xu, Neil Boyette, Jalal Mahmud, Rama Akkiraju, and Vibha Sinha. 25 tweets to know you: A new model to predict personality with social media. 2017.

[2] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *Eprint Arxiv*, 1, 2014.

[3] Yoon Kim. Convolutional neural networks for sentence classification. *Eprint Arxiv*, 2014.

[4] Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. 2016.

[5] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2017.

[6] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. A c-lstm neural network for text classification. *Computer Science*, 1(4):39–44, 2015.