You Are What You Say: Character Classification Based On TV Scripts

Shihan Ran
School of Data Science
FuDan University
15307130424@fudan.edu.cn

Abstract

Understanding the interactive structure of dialogues, such as turn taking behavior and change of speakers, as well as capturing individual characteristics, such as background information and speaking style, are a critical prerequisite for building a more natural dialogue systems. This paper presents a novel approach for modeling persona information in multi-participant open-domain dialogue systems using a sequence-to-sequence generative model with hierarchical structure which encodes personas in distributed embeddings. In experiments of speaker identification from TV scripts, our models yield qualitative performance improvements over baseline models.

1 Introduction

Whether online or offline, we have plenty of conversations in our daily life. It is commonly acknowledged that natural language text itself reflects the personality of the speaker, in addition to its semantic content (Mairesse et al., 2007). Since personality can uniquely characterizes an individual, and profoundly influences his mental status, social behaviors and speaking styles (Burger, 2008), during the process of interaction with others, more or less, our personality will be manifested through words and deeds. This leads us to the interesting open problem of, by analyzing dialogues, whether or not algorithms are able to distinguish speakers based only on small amount of text, further, to find out personalities and interests which can help people to understand themselves better.

Some work has been done on personality analysis based on social media, including the analysis of relationships, topics and interests. It can't be denied that social media provides a very convenient and widely used platform for conversations and discussions among users. With hundreds of millions of users participating on social media and sharing their self-authored content, social media can provide a tremendous opportunity for personality modeling (Arnoux et al., 2017). Also, social applications considering users personality are capable of providing more adaptive and personalized user experience (Hu et al., 2016; Liu et al., 2016). However, recent work which concentrate on predicting personality from online behaviors (Golbeck et al., 2011; Bai et al., 2013; Farnadi et al., 2016) relies heavily on handcrafted feature selection from social media posts, while they also take plenty of non-utterance features (*e.g.*, the number of people they follow on a social media) into consideration. Moreover, prior modeling methods require too much input data to be realistically used, which means we can't be confident of how well these models would work with real life scenarios. One major issue for these data-driven models is their propensity to select the answer with greatest likelihood in the training data.

For present purposes, we will define PERSONA as the personality and character that an artificial agent, as actor, plays or performs during conversational interactions. A persona can be viewed as a composite of elements of identity (background facts or user profile), language behavior, and interaction style. It is also adaptive, since an agent may need to present different facets to different human interlocutors depending on the interaction. One difference between dialogues and ordinary Twitter-like dynamically posting social media is that both parties in the dialogues may play vital roles in the speaker identification task. Understanding the interactive structure of dialogues, such as turn taking behavior and change of speak-

ers, as well as capturing individual characteristics, such as background information and speaking style, are a critical prerequisite for building a more natural dialogue systems. There has been a growing research trend in training dialogue systems from large volumes of human-to-human interactions (Ritter et al., 2011; Sordoni et al., 2015; Luong et al., 2014; Shang et al., 2015). Previous study (Li et al., 2016) has found that persona information can be successfully modeled in dialogue systems with deep neural networks. Inspired greatly from this successful methods, we propose our generative classifier, which is also proved to be more resistance to overfitting problems and noisy data in recent work (Yogatama et al., 2017). Our model aims to extract persona information from natural daily conversations which are easily obtained and to do speaker identification based on these persona information. This model can be further widely applied in filed and situation like human resource, recommendation system.

The main contribution of this paper are as follows:

- This paper manages to do speaker identification through analyzing text and context only.
- We propose a generative sequence-to-sequence (SEQ2SEQ) (Sutskever et al., 2014) classifier model with recurrent neural network (RNN) architecture (Mikolov et al., 2010) which takes persona information into consideration.
- We made some hierarchical architecture (Li et al., 2015) improvements in encoder part and can simultaneously learn latent vector representation of persona information during training.
- We build up a corpus for training and evaluating the system as well as achieved significant performance on the task compared to other models based on handcraft features.

2 Identification Task

Given several consecutive sentences, denote speaker (with a certain personality) as S, preceding text (sentences without the last one) as X, response (the last sentence) as Y. It can be illustrated as Table 1.

Discriminative models are trained to distinguish the correct label among possible choices, which means these models are trained to maximize the conditional probability of the labels given the documents:

$$\arg\max_s P(s \mid X, Y) \tag{1}$$

Generative models, on the other hand, are trained to maximize the joint probability of labels and documents. After transforming it with the Bayes rule, we got

$$\begin{aligned} \arg\max_s P(s, X, Y) =& P(s \mid X, Y)P(X, Y) \\ P(s \mid X, Y) =& \frac{P(Y \mid s, X)P(s)}{P(Y)} \end{aligned} \tag{2}$$

3 Sequence-to-Sequence Models

In this section we give a quick overview of sequence-to-sequence models. Sequence-to-sequence models (Sutskever et al., 2014) are defined as follows: Given a sequence of inputs $X = \{x_1, x_2, \cdots, x_{n_X}\}$, a long-short term memory (LSTM) associates each time step with an input gate, a memory gate and an output gate, respectively denoted as i_t, f_t and o_t. For notations, we disambiguate e and h where e_t denotes the vector for an individual text unit (*e.g.*, a word or sentence) at time step t while h_t denotes the vector computed by the LSTM model at time t by combining e_t and h_{t-1}. c_t is the cell state vector at time t, and σ denotes the sigmoid function. Then, the vector representation h_t for each time step t is given by:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^s \end{bmatrix} \tag{3}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \tag{4}$$

$$h_t^s = o_t \cdot \tanh(c_t) \tag{5}$$

where W_i, W_f, W_o, $W_l \in \mathbb{R}^{K \times 2K}$. In SEQ2SEQ generation tasks, each input X is paired with a sequence of outputs to predict: $Y = \{y_1, y_2, \cdots, y_{n_Y}\}$. The LSTM defines a distribution over outputs and sequentially predicts tokens using a softmax function:

$$\begin{aligned} p(Y|X) &= \prod_{t=1}^{n_y} p(y_t|x_1, x_2, \cdots, x_t, y_1, y_2, \cdots, y_{t-1}) \\ &= \prod_{t=1}^{n_y} \frac{\exp(f(h_{t-1}, e_{y_t}))}{\sum_{y'} \exp(f(h_{t-1}, e_y))} \end{aligned} \tag{6}$$

Setting	Character	Text
Preceding Text	Penny	So, what do you guys do for fun around here?
	Sheldon	Well, today we tried masturbating for money.
	Leonard	Okay, well, make yourself at home.
Response	Penny	Okay, thank you.

Table 1: Given four consecutive sentences, we denote sentences without the last sentence as preceding text and the last one as response.

where $f(h_{t-1}, e_{y_t})$ denotes the activation function between h_{t-1} and e_{y_t}, where h_{t-1} is the representation outputted from the LSTM at time $t-1$. Each sentence terminates with a special end-of-sentence symbol *EOS*. In keeping with common practices, inputs and outputs use different LSTMs with separate parameters to capture different compositional patterns.

In the decoding procedure, the algorithm terminates when an *EOS* token is predicted. At each time step, either a greedy approach or beam search can be adopted for word prediction. Greedy search selects the token with the largest conditional probability, the embedding of which is then combined with preceding output for next step token prediction. For beam search, Sutskever et al. (2014) discovered that a beam size of 2 suffices to provide most of benefits of beam search.

4 Persona Information Extraction

Our objective is to build an automatic classifier for the identification task defined above. We consider two baseline models and then introduce two proposed models: Discriminative Classifier Models, which takes preceding text and response into consideration, and Generative Classifier Models which models the personality of the speaker. In the discriminative model the RNN reads the text and uses its hidden representation to model the class posterior; In the generative model, text are generated word by word, conditioned on a learned speaker persona embedding.

4.1 Notation

Let P denote the preceding text, which is comprised of a sequence of N_P sentences, $P = \{s_1, s_2, \cdots, s_{N_P}, end_P\}$. An additional end_P token is appended to each preceding text. Each sentence s is comprised of a sequence of tokens $s = \{w_1, w_2, \cdots, w_{N_s}\}$ where N_s denotes the length of the sentence, each sentence ending with an end_s token. The word w is associated with a K-dimensional embedding e_w, $e_w = \{e_w^1, e_w^2, \cdots, e_w^K\}$. Let V denote vocabulary size. Each sentence s is associated with a K-dimensional representation e_s.

Typically, the SEQ2SEQ model first compresses the input preceding text X into a vector representation e_P using neural models (encoding), and then generates response Y based on e_P (decoding).

For simplicity, we define $LSTM(h_{t-1}, e_t)$ to be the LSTM operation on vectors h_{t-1} and e_t to achieve h_t as in Equ.3 and 4. For clarification, we describe the following notations used in encoder and decoder:

- h_t^w and h_t^s denote hidden vectors from LSTM models, the subscripts of which indicate time-step t, the superscripts of which indicate operations at word level (w) or sequence level (s).
- e_t^w and e_t^s denotes word-level and sentence-level embedding for word and sentence at position t in terms of its residing sentence or preceding text.

4.2 Model 1: Baseline Models

The first baseline model always predicts the most frequent class. We call this model the *Majority Class Predictor*. This is the most naive baseline available but it's useful for the purpose of defining the difficulty of the task.

The second baseline model is a *traditional machine learning model*, which uses naive bayes classifier and logistic regression classifier (Ng and Jordan, 2002), taking as input a set of hand-crafted features. Largely inspired by the work of Walker et al. (2012), we compute the following feature for each sentence:

- Sum of TF-IDF (term frequency times inverse document frequency) (Joachims, 1996) values for each word in the sentence, where the document is defined to be all the words in the movie script.

4.3 Model 2: RNN Discriminative Model

We propose to use recurrent neural networks (RNNs) for embedding words into a distributed vector space representation (Bengio et al., 2003; Mikolov et al., 2013). It might be useful to embed more preceding text, hence we will embed the previous k sentences. Inspired by the approach taken by Yu et al. (2014), we use two GRU RNNs (Cho et al., 2014) to transform words into a vector representation. The first RNN reads the previous k sentences word-by-word forwards producing a sequence of hidden states $h_1^p, \cdots, h_t^p$, where p stands for preceding. The last hidden state of the RNN is taken to be the vector representation of the preceding sentences: $v^p = (h_t^p)^T$. Likewise, the second RNN reads the response sentence word-by-word forwards producing another sequence of hidden states $h_1^r, \cdots, h_t^r$, where r stands for response. Its last hidden state is taken to be the vector representation of the response sentences: $v^r = h_t^r$. We constrain the RNNs to share parameters for the word embeddings. Further details on the RNN architectures are given by Cho et al. (2014). The model is illustrated in Figure 1.

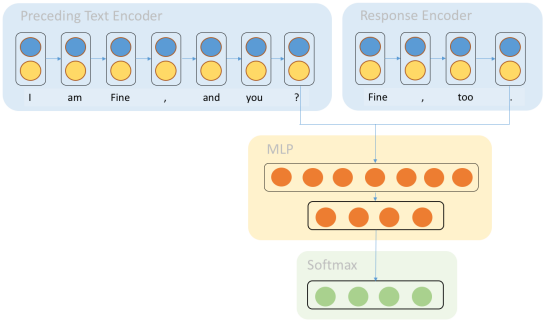

Figure 1: The computational graph of the RNN model. The vector embeddings v^p and v^r are produced respectively by running one RNN over the word sequences in the preceding sentences and one RNN over the word sequences in the response sentences.

To help generalization further, we may fix the word embedding parameters with GloVe word embeddings (Pennington et al., 2014) trained on five corpora of varying sizes. This should improve the semantic representation of each word, since the corpora is more than a magnitude larger than our TV script dataset. We call this the *Discriminative+GloVe* model.

We train both models by optimizing the log-likelihood of the labels, where all parameters are shared between the models.

4.4 Model 3: SEQ2SEQ Generative Model

Our first generative model is the Speaker Model from Li et al. (2016), which models the respondent alone. This model represents each individual speaker as a vector or embedding, which encodes speaker-specific information (*e.g.*, dialect, register, age, gender, personal information) that influences the content and style of his responses. Note that these attributes are not explicitly annotated, you can take it as hidden variables. Instead, our model aims to cluster speakers along their speaking style and extract persona information based on the responses alone.

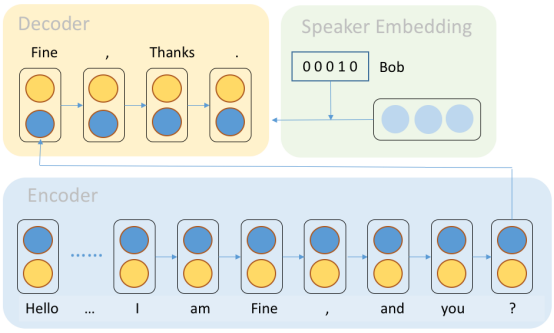

Figure 2: Illustrative example of the Generative Model introduced in this work. These speaker embeddings are learned jointly with word embeddings and all other parameters of the neural model via backpropagation.

Figure 2 gives a brief illustration of the SEQ2SEQ Generation Model. Each speaker $i \in [1, N]$ is associated with a speaker-level representation $s_i \in \mathbb{R}^{K \times 1}$. As in standard SEQ2SEQ models, we first encode preceding text into a vector representation h_t^p using the source LSTM. Then for each step in the target side, hidden units are obtained by combining the representation produced by the target LSTM at the previous time step, the word representations at the current time step, and the speaker embedding s_i:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^p \\ s_i \end{bmatrix} \tag{7}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \tag{8}$$

$$h_t^p = o_t \cdot \tanh(c_t) \tag{9}$$

where $W \in \mathbb{R}^{4K \times 3K}$. In this way, speaker information is encoded and injected into the hidden layer at each time step and thus helps predict personalized responses throughout the generation process. The Speaker embedding s_i is shared across all conversations that involve speaker i. s_i are learned by backpropagating word prediction errors to each neural component during training.

Another useful property of this model is that it helps infer answers to questions even if the evidence is not readily present in the training set. This is important as the training data does not contain explicit information about every attribute of each speaker. The model learns speaker representations based on conversational content produced by different speakers, and speakers producing similar responses tend to have similar embeddings, occupying nearby positions in the vector space. This way, the training data of speakers nearby in vector space help increase the generalization capability of the SEQ2SEQ model. For example, consider two speakers i and j who sound distinctly British, and who are therefore close in speaker embedding space. Now, suppose that, in the training data, speaker i was asked *Where do you live?* and responded *in the UK*. Even if speaker j was never asked the same question, this answer can help influence a good response from speaker j, and this without explicitly labeled geo-location information.

4.5 Model 4: Hierarchical Generative Model

In the standard SEQ2SEQ generative model, we just splice the preceding text and throw them into encoder. However, sentences said by different speaker should be treated differently and separately. The hierarchical model (Li et al., 2015) draws on the intuition that just as the juxtaposition of words creates a joint meaning of a sentence, the juxtaposition of sentences also creates a joint meaning of a paragraph or a document. Details can be shown like Figure 3.

Encoder We first obtain representation vectors at the sentence level by putting one layer of LSTM (denoted as $LSTM^{word}_{encode}$) on top of its containing words:

$$h^w_t = LSTM^{word}_{encode}(e^w_t, h^w_{t-1}) \tag{10}$$

The vector output at the ending time-step is used to represent the entire sentence as

$$e_s = h^w_{end_s} \tag{11}$$

To build representation e_P for the current preceding context P, another layer of LSTM (denoted as $LSTM^{sentence}_{encode}$) is placed on top of all sentences, computing representations sequentially for each time-step:

$$h^s_t = LSTM^{sentence}_{encode}(e^s_t, h^s_{t-1}) \tag{12}$$

Representation $e^s_{end_P}$ computed at the final time-step is used to represent the entire document: $e_P = h^s_{end_P}$.

Thus one LSTM operates at the token level, leading to the acquisition of sentence-level representations that are then used as inputs into the second LSTM that acquires document-level representations, in a hierarchical structure.

Decoder The decoder part is the same as in standard SEQ2SEQ Generative Model.

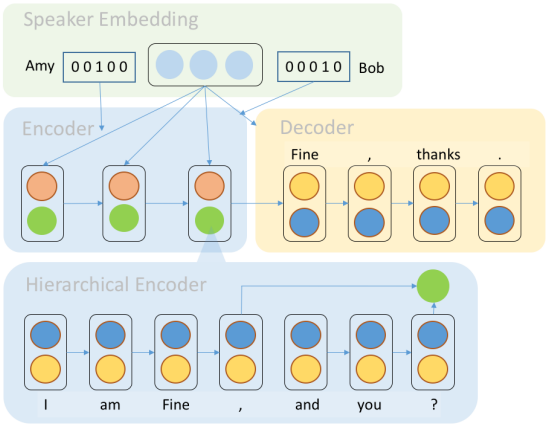

Figure 3: Illustrative example of the Hierarchical Generative Model introduced in this work.

4.6 Training and Testing

Parameters are estimated by maximizing likelihood of outputs given inputs, similar to standard SEQ2SEQ models. A softmax function is adopted for predicting each token within output response sentence. Stochastic gradient descent with mini-batches is adopted.

Like we defined before, during our prediction, we feed each speaker label to the model. For decoding process, N possible next-word candidates list is generated. We figure out

$$P(Y|s_i, X) = P(y_1|s_i, X) \cdots P(y_n|s_i, X, y_{i<n}) \tag{13}$$

Speaker	Training	Dev	Test
Sheldon	33146	3881	3875(21.12%)
Leonard	28460	2799	3559(19.40%)
Penny	21726	2432	3050(16.62%)
Howard	17071	1628	1903(10.37%)
Raj	13110	1328	1492(8.13%)
Amy	8503	1098	1123(6.12%)
Others	22260	3522	3346(18.24%)

Table 2: Labeled instances for identification tasks.

where $p(Y|s_i, X)$ denotes the probability of the generated response Y given the preceding text X and the respondents speaker ID i. The speaker ID which gets highest probability is the prediction result in the final output.

5 Experiments

5.1 Dataset

We used scripts from the American television comedies *The Big Bang Theory*[1], available from Internet Movie Script Database (IMSDb)[2]. We split the corpus into training/development/testing sets, at ratio 8:1:1. Statistics for each class in the task are outlined in Table 2.

We use a slide window to do dataset augmentation (*e.g*, suppose the number of sentences of preceding text is three, and there are eight consecutive sentences numbered from 1 to 8, we generate data group like 1234, 2345, 3456, 4567, 5678 rather than 1234, 5678. In each data group, the first three sentences are preceding text and the last one is response) which can help us expand the dataset enormously. We only consider sentences longer than five words as well as keep a vocabulary set consisting of the 20,000 most frequent words. A special *UNK* token is used to denote all the remaining less frequent tokens.

5.2 Training Details and Implementation

Previous research has shown that deep LSTMs work better than shallow ones for SEQ2SEQ tasks (Sutskever et al., 2014). We adopt a LSTM structure with 1 layer for encoding and 1 layer for decoding, each of which is comprised of a different set of parameters. Each LSTM layer consists of 300 hidden neurons and the dimensionality of word embeddings is set to 300. Other training details are given below, some of which follow Sutskever et al. (2014).

- LSTM parameters and word embeddings are initialized from a uniform distribution between [-0.08, 0.08].
- Stochastic gradient decent is implemented without momentum using a fixed learning rate of 0.005. We trained our models for a total of 10 epochs.
- Batch size is set to 32.
- 0.2 dropout rate.

[1] https://en.wikipedia.org/wiki/The_Big_Bang_Theory

[2] http://www.imsdb.com

Model	accuracy
Majority Class Predictor	21.12%
Logistic Regression	25.79%
Multinomial Naive Bayes	23.28%
Random Forest	22.60%
Discriminative	27.03%
Discriminative+GloVe	**27.78%**(+31.5%)
Standard Generative	20.93%
Hierarchical Generative	22.26%

Table 3: Speaker Identification accuracies.

5.3 Evaluations

Since it's a classification problem, we use accuracies as evaluation metric. Also, we adapt confusion matrix and weights visualization to help us adjust parameters.

5.4 Results

A summary of our experimental results is given in Table 3. First, we observe that traditional machine learning classifiers (Logistic Regression, Naive Bayes, Random Forest) is able to outperform the Majority Class Predictor baseline on the task. With more handcraft features and rules, there is a great chance that performance will be better. Second, we see that the RNN-based Discriminative models clearly outperform the traditional machine learning models on this tasks. The highest of Discriminative models reach 27.78%, which is about 31.5% increase than Majority Class baseline. This suggests that the RNNs have learned sentence-level embeddings that capture speaker's characteristics. In particular, the observation that the Discriminative+GloVe model performs slightly better than the one without GloVe word embedding suggests that additional labeled training data may improve performance substantially. Third, we can see from the results that Hierarchical structure outperform than standard SEQ2SEQ structure, which

is reasonable and indicates it learns document-level embeddings that captures the interactive information of dialogues. By comparing Discriminative models and Generative models, we demonstrate empirically that our discriminative model obtains a lower asymptotic error rate than its generative counterpart.

5.5 Qualitative Analysis

Seriously Overfitting Problem After training for 7 epoch, there exists a tremendous gap between performances of training set and development set. Take Discriminative model as example, accuracies of training set can achieve 75% while the one of development remains 24%. Confusion matrix of which can be plotted like Figure 4.

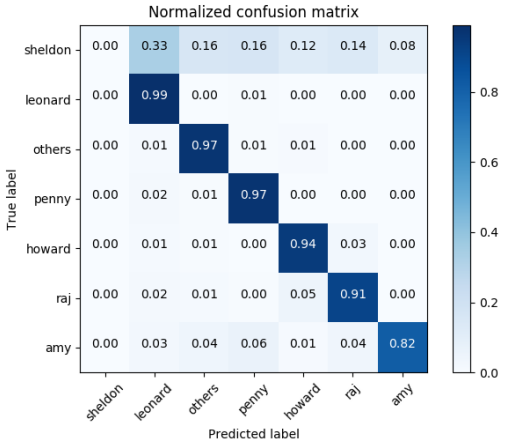

Figure 4: Normalized confusion matrix of Training set from Discriminative model with accuracies 75%

The visualization of speaker embedding learned in Generative models are like Figure 5. As we can see in the picture, seven speakers can be separated well. However, there is no obvious clustering trend, perhaps because of insufficient training data.

Since the relatively small size of the dataset does not allow for training an open domain dialog model, except from doing dataset augmentation, we can also adopted a domain adaption strategy where we first trained a standard SEQ2SEQ models using a much larger OpenSubtitles (OSDb) dataset (Tiedemann, 2009), and then adapting the pre-trained model to the TV series dataset.

The OSDb dataset is a large, noisy, open-domain dataset containing roughly 60M-70M scripted lines spoken by movie characters. This

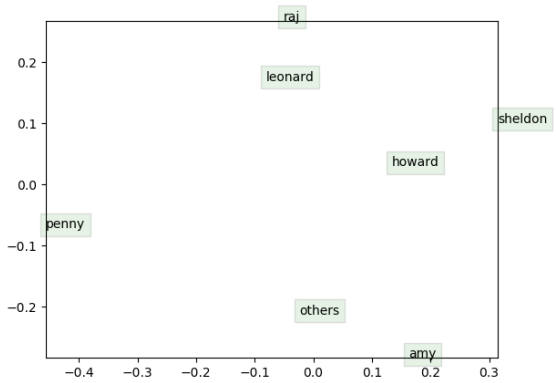

Figure 5: Visualization of speaker embedding learned in Generative models

dataset does not specify which character speaks each subtitle line, which prevents us from inferring speaker turns. However, we can initialize word embeddings and LSTM parameters using parameters learned from OpenSubtitles datasets. It may help to ease the overfitting phenomenon.

6 Conclusions and Future Work

This work presents several models to automatically infer speakers from scripted dialogues. We showed that discriminative models are better than generative models in this task. The models considered show promising performances for automatically identifying speaker in multi-participant open-domain dialogues. However, in Generative models, we've only considered persona information. There are many other dimensions of speaker behavior, such as mood and emotion, that are beyond the scope of the current paper and must be left to future work.

References

Pierre-Hadrien Arnoux, Anbang Xu, Neil Boyette, Jalal Mahmud, Rama Akkiraju, and Vibha Sinha. 2017. 25 tweets to know you: A new model to predict personality with social media. *arXiv preprint arXiv:1704.05513* .

Shuotian Bai, Bibo Hao, Ang Li, Sha Yuan, Rui Gao, and Tingshao Zhu. 2013. Predicting big five personality traits of microblog users. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013 IEEE/WIC/ACM International Joint Conferences on*. IEEE, volume 1, pages 501–508.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic lan-

guage model. *Journal of machine learning research* 3(Feb):1137–1155.

JM Burger. 2008. Personality . australia: Belmont, ca: Wadsworth: Thomson learning.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Golnoosh Farnadi, Geetha Sitaraman, Shanu Sushmita, Fabio Celli, Michal Kosinski, David Stillwell, Sergio Davalos, Marie-Francine Moens, and Martine De Cock. 2016. Computational personality recognition in social media. *User modeling and user-adapted interaction* 26(2-3):109–142.

Jennifer Golbeck, Cristina Robles, Michon Edmondson, and Karen Turner. 2011. Predicting personality from twitter. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*. IEEE, pages 149–156.

Tianran Hu, Haoyuan Xiao, Jiebo Luo, and Thuy-vy Thi Nguyen. 2016. What the language you tweet says about your occupation. In *ICWSM*. pages 181–190.

Thorsten Joachims. 1996. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155* .

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057* .

Leqi Liu, Daniel Preotiuc-Pietro, Zahra Riahi Samani, Mohsen Ebrahimi Moghaddam, and Lyle H Ungar. 2016. Analyzing personality through social media profile picture choice. In *ICWSM*. pages 211–220.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206* .

François Mairesse, Marilyn A Walker, Matthias R Mehl, and Roger K Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of artificial intelligence research* 30:457–500.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Andrew Y Ng and Michael I Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*. pages 841–848.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 583–593. http://dl.acm.org/citation.cfm?id=2145432.2145500.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364* .

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Jörg Tiedemann. 2009. News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*. volume 5, pages 237–248.

Marilyn A Walker, Grace I Lin, and Jennifer Sawyer. 2012. An annotated corpus of film dialogue for learning and characterizing character style. In *LREC*. pages 1373–1378.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898* .

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632* .