UNIVERSITY OF CALIFORNIA

SANTA CRUZ

A STORY RECOMMENDATION SYSTEM FOR A KIDS CHATBOT, SLUGCHAT

A report submitted in partial satisfaction
of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Yifei Wu

December 2017

The Report of Yifei Wu
is approved:

——————————————————
Professor Yi Zhang, Chair

——————————————————
Professor Luca de Alfaro

——————————————————
Tyrus Miller
Vice Provost and Dean of Graduate Studies

A Story Recommendation System for a Kids Chatbot, SlugChat

Yifei Wu
University of California, Santa Cruz
ywu151@ucsc.edu

ABSTRACT

Listening and speaking are important skills for kids. They improve kids' intelligence and verbal ability. The booming of AI chatbot technology provides a good method for kids to practice these skills. With the increasing time children spending on smart devices, there is a growing interest in mobile chatbots for kids. In this paper, we propose a story recommendation system deployed on a kids mobile chatbot, SlugChat. The proposed system collects stories and their listening behaviors. It get preferences by using story vectors to train LSTM model. We also propose how we deploy this system to the chatbot.

1 INTRODUCTION

Childhood is the beginning of human life and plays an essential role. Some researchers show that a better childhood may provide a better development in many fields like intelligence and verbal ability. A good method to achieve these is talking more with kids especially in their first eight years[10, 13].

Most children could get a better vocabulary acquisition by listening to stories and explanations of target words from their parents[5]. However, not all parents have enough time to chat with their kids. For example, a report indicts that over 90% children complain their parents lack of time to chat with[6].

On the other hand, although some parents would not have enough time to speak to their kids, children spend increasing time on electronic devices[21]. These devices use many technologies to improve user experience, such as AI chatbot and recommendation system.

AI chatbot offers children an excellent platform to practise verbal ability. Furthermore, recent reports show that kids under 6 have great interests in interacting with electronic devices despite limited literacy[20]. Nowadays, almost all IT giants develop their own chatbot like Amazon's Alexa which provides functions over Music & Radio, Game or Trivia, and Podcast[11]. Among these chatbot, there are also excellent products for kids, such as Tyche, ibotn, BeanQ, and Apphome[1].

Recommendation systems have the effect of guiding users in a personalized way to interesting objects in a large space of possible options[17]. In today's world, recommendation systems have become more and more popular. Some recommendation algorithms are based on users' behaviors[16]. Songs, movies, news and so many things could be recommended to users based on their and others' previous behaviors. It not only saves time but also encourages people to generate more behaviors. Then algorithms would recommend more right things to users. Unfortunately, most of current recommendation systems are for all ages. And there are no public available papers about recommendation systems about children's literacy, such as short stories for kids.

In this paper, we present a short story recommendation system in Mandarin for kids between three to six years old. We deploy this system on a new mobile application for kids, Slugchat, who has the abilities to tell stories, sing children's songs, read poems, and answer some general questions for kids.

2 RELATED WORK

In order to build a story recommendation system and deploy it on a chatbot, we need technologies from different areas. We introduce these technologies in this section.

2.1 Web Crawler

A web crawler, sometimes called a spider, is a program which is able to browse the World Wide Web methodically and automatically[7]. A web crawler collects massive data from various sources. This data remains in an unstructured form. We call them raw data and we need to derive useful values from them. Thus, a web crawling is usually the first step of many work.

2.2 Word Segmentation

In languages such as English, words are separated by spaces. While there are no spaces beween words in sentences in Mandarin. It is hard to analysis word token directly. In this project, we use a python liberary *jieba* to do word segmentation. We build a dictionary for kids.

2.3 Word Vector

Word2vec is a group of related models. Each unique word is assigned a vector in a space (or location). Words contain familiar contexts are located nearby[8]. These models could be trained by giving a large corpus of text. In this project, we build word2vec models for kids' words in Mandarin.

2.4 Story Vector

There are several ways to represent a story, such as Word Mover's Distance[14], Bag of Words and Paragraph Vector[15]. In this project, we use Bag of Words to represent stories. And in future work, we will use the states in RNN to present each story.

2.5 Recommendation System

Recommendation systems filter useful information to predict the preference of a user by giving items[9, 19]. Generally, there are two kinds of recommendation algorithms, collaborative filtering and content-based[12]. Collaborative filtering algorithms are based on users' data, such as behaviors, activities and preferences. They predict the results based on not only the user's own data but also data from similar users[4]. Content-based filtering algorithms predict the relation between features or labels of items and preferences of users[3]. While, it is also feasible to combine these two kinds of algorithms in order to predict more accurately and effectively. They are hybrid recommendation systems/citeadomavicius2005toward. In this project, we develop a story recommendation system based on children's listening behaviors.

2.6 Dialogue System

The developers of dialogue systems need to design conversation flows and rules to make systems work well[2]. Luckily, there are some wonderful platforms, such as Dialogflow, wit.ai, IBM Watson, Microsoft LUIS and Rulai. They help developers build systems automatically and provide codeless interfaces for designing dialogue flows[18]. In this project, we use to Dialogflow. We build a set of intents, entities and contexts to implement our dialogue and we use webhook to post responses to our servers.

3 PROJECT DESIGN

In this section, we show how we design our project. For each step, we list all possible methods, compare their practicalities and performances and choose the suitable one.

3.1 Data Collection

The first step to build this recommendation system is collecting suitable stories for kids. We can not just download the text set of stories from the Internet directly because of the following reasons:

(1) Our users are kids between three to six years old. Parts of them may have limited literacy. Thus, we need to provide stories with audios.
(2) Our recommendation algorithm is based on the content of stories. Thus, our stories should also have text.
(3) For the cold start, we need to know the styles and target ages for each story.

In summary, we need stories with exactly the same text and auto type, target ages, styles and in Mandarin. After searching, we decided to use stories from a no copyright restrictions story website, Beva. We collect 700 stories in Beva.

The second step is collecting the listening behaviors of our stories. There are four ways: collecting real users' listening behaviors through our chatbot, simulation kids' listening behaviors, searching existing data and crawling real data from the Internet. However, we only have 2 months to finish this project, there is not enough time for us to collect data by ourselves. Simulated data are lack of convincing and do not contain the implied relation among stories. And as described in Chapter 1, there is no story recommendation system for kids. Thus, the only way for us is crawling data from other websites.

We collect the comments and their usernames under each story in Beva. Then we order stories with comments in time order for each username to get their listening lists. However, there are only 23 users who have listened to more than one story. And these users only listen to 54 stories among 700 stories. This data set is too small.

Then, we crawl data from NetEase Cloud Music, which is a music and video streaming service website. There are stories for kids. And users are able to create their own playlists and share with others without copyright restrictions. The playlists are also listening lists of users. In this website, I collect 48908 listening lists. Their lengths distribution is shown in Figure 1. This dataset is enough to be training data.

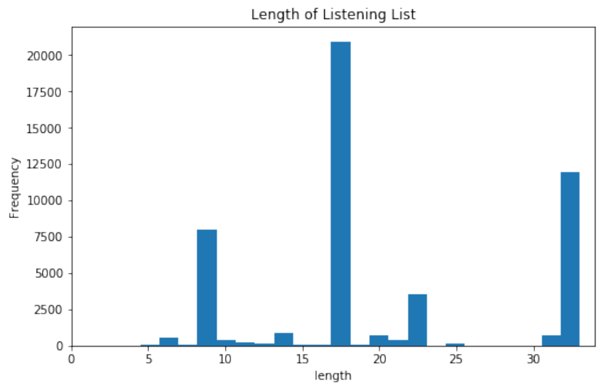


Figure 1: Distribution of Listening List Length

3.2 Data Preprocessing

After getting stories and listening lists. We abstract words and stories to vectors.

For word vectors, we use word2vec to build Chinese words models for kids. However, we only have 700 short stories whose text size is about 1.5 MB. This dataset can not provide enough implied information to train good word2vec models. After training, the average similarity between each two words is about 90% which is horrible. The distribution of top 150 most frequent words in two-dimensional space is shown in Figure 7.

Then we try to use pre-trained models in Chinese. And using principal component analysis to reduce the size of dimensions. We download several models including official Chinese model from Google and customized models which training from over 150 GB corpus. However, the Chinese language does have the similar concept as lemma and stem. It is hard to train a dictionary model which contains all Chinese words. In particular, our target users are kids, which means there are a few words are unusual. These pre-trained models fail in covering all words we need. We decided to enlarge our text corpus. We only need the suitable sentences for kids and do not need to find the same audio. Thus, we collect 24 MB children stories as our new text corpus. After training, we get acceptable Chinese word2vec models.

For story vectors, we use Bag of Words first and evaluate its performance. In the future, we will use better models to improve this part.

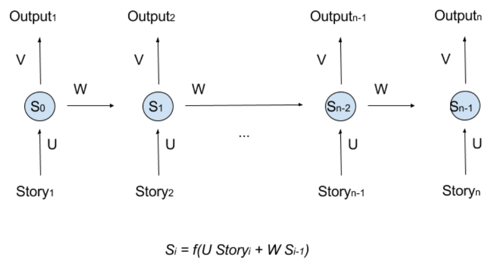


Figure 2: RNN Story Vector

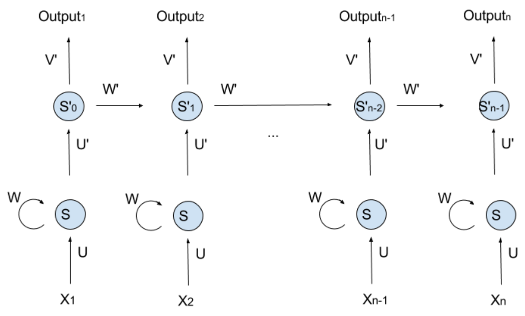


Figure 3: RNN Word Vector

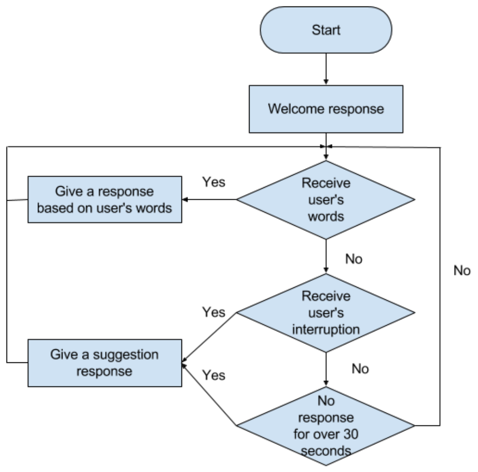
3.3 Recommendation Algorithm

There are many recommendation algorithms. For example, matrix factorization without any content features is a classic collaborative filtering method. However, we can not use it. Our listening lists fail in covering all 700 stories. Even them could, our dataset will continually add new stories. Thus, we use Recurrent Neural Network (RNN) to extract content features in listening lists.

There are two methods to build Recurrent Neural Network for our system.

(1) Use each story vector as each input, as shown in Figure 2. Story$_i$ is the ith story vector in a listening list.
(2) Build a two-level neural network. Use each word vector in each sentence in each story as an input of the first level. Using the states of each story in the first level as the inputs of the second level, as shown in Figure 3. X$_i$ is the list of word vectors of the ith story in a listening list.

In this project, we realize the first method and evaluate its performance. In future work, we will realize other methods.



Start
Welcome response
Receive user's words
Yes
Give a response based on user's words
No
Receive user's interruption
Yes
Give a suggestion response
No
No response for over 30 seconds
Yes
No

Figure 4: The Diagram of SlugChat

3.4 Dialogue Server

We develop a mobile application, SlugChat. Its diagram is shown in Figure 4. It is able to talk with users (most of them are children). It has the following functions:

(1) Answer general questions, such as *'What is Beijing?'* and *'Who is Washington?'*.
(2) Own human features like age, name. Users can interact with SlugChat about these.
(3) Ask questions to users, such as *'Is there any interesting thing today?'*.
(4) Reading poems based on users' words.
(5) Reading stories to users.
(6) Allow users to interrupt at any time.

In this project, we only focus on the backend of reading stories function. We need finish the following work:

(1) Deploy an Amazon Relational Database Service. Create a story table to store stories and a log table to store users behaviors.
(2) Deploy an Amazon Elastic Compute Cloud platform and build a backend server by express on it. This server has three post responses. Send a recommended story, send a story by its id and send a story by its title.
(3) Use Dialogflow to create story intents corresponding to the above post responses.

4 IMPLEMENTATION

In this section, we introduce the detailed steps about how to implement the whole project in each part.

4.1 Data Collection

Frist step is crawling stories, their feature and comments from Beva.

(1) Get source code of each page for the story list for each age from 3 to 6 by python lib *requests*.

Table 1: Story Style and Age Distribution

Style	Total	Age3	Age4	Age5	Age6
Aesop's Fable	147	67	76	99	133
Celebrity Story	84	27	32	54	78
Andersen's Fairy	20	18	18	18	18
Idiom Story	114	33	45	84	105
Grimm's Fairy	77	47	60	66	63
Bedtime Story	197	160	141	99	70
Fairy Tale	61	51	56	55	47

[1] Age*i* means stories suitable for i-year-old kids.
[2] Some stores are suitable for multiple ages.

(2) Get story divs by python lib *bs4.BeautifulSoup*.
(3) Extract story titles and content URLs from story divs.
(4) Get the source code in story content page by URL.
(5) Extract the content, style, audio link and get the request URL for comments by python lib *re*.
(6) Send a get request to get comment list.
(7) Store stories and usernames of comments.
(8) Store stories to database.

The style and age distribution of stories are shown in Table 1.

The second step is crawling listening behaviors from NetEase Cloud Music. Here are the detailed steps:

(1) Get all story titles and remain all Chinese characters by *re*.
(2) Dynamically search *'children story'* in playlist part in NetEase website by python lib *selenium.webdriver*.
(3) Extract URLs of playlists by *BeautifulSoup* and add these URLs into a playlist queue Q.
(4) Get story title list in the first playlist in queue Q by *requests* and remove this list from queue Q.
(5) Remain the same stories in our story set.
(6) If this playlist has the same stories in our story set, get the URLs of its preference playlists and add them into queue Q if they are never added into queue Q before.
(7) Continue to next playlist in queue Q until queue Q is empty.

4.2 Data Preprocessing

Frist step is training word2vec models.

(1) Filtrate Chinese characters and stop punctuations of sentences from corpus by *re*.
(2) Split word tokens by python lib *jieba*.
(3) Remove stop words.
(4) Store word tokens into a two-dimensional array. First dimension store sentence words and the second dimension story words.
(5) Calculate word frequencies and rank words by frequencies.
(6) Train word2vec models by python lib *gensim*.
(7) Reduce the number of dimension by python lib *TSNE* and visualize these models by python lib *matplotlib*
(8) Evaluate the performance of these models.

As described in Chapter 3.2, the first time we use text corpus with 700 stories and get bad models. The distribution of top 150 most frequent words in 2 dimensional space is shown in Figure 7.

In order to get acceptable word2vec models, we crawl 24 MB story text file from many Chinese children story websites. Then we use this corpus to train word2vec models. We train different models with different vector sizes from 25 to 100. Finally, we choose to use models with vector size 50. The distribution of top 150 most frequent words in 2 dimensional space is shown in Figure 8.

The second step is extracting story vectors.

(1) Get word multiset for each story.
(2) Remove stop words. In order to increase the differences among story vectors. The size of our stop word set is 844.
(3) Remove words whose frequencies are less than 10. We remain 2130 words.
(4) Build an one hot vector for each story. Each story is represented by a 2130-dimensional vector.
(5) Get 80 stories randomly and visualize them in two-dimensional space. We use different colors to represent different styles. We can see them from Figure 9

From Figure 9, we can find a trend that stories with the same styles are more like to locate in the same clusters. It shows that our story vector model is good.

4.3 Recommendation System

In this section, we use LSTM to predict whether a story should be recommended when we know the previous listening behaviors of users.

The data we collect is users' playlists. Based on our data, we have the following assumptions:

(1) A user likes all stories in his/her playlist and does not like stories out of his/her playlist. We try to use LSTM and story vector to predict the next stories a user would like when we know the subset of his/her playlist.
(2) The orders in playlists don't have any impact on users' next preference.

Based on these assumptions, we use the following rules to generate our training set and testing set:

(1) All input items are lists of story vectors.
(2) All input items have labels, either $[0., 1.]$ which represents the user like all stories in this list or $[1., 0.]$ which represents the user like all stories except the last one in this list. We call $[0., 1.]$ $user - like$ and $[1., 0.]$ $user - dislike$.
(3) Each time we get a subset from a playlist, we just return an unordered list.
(4) For an input item whose length is n. If it is $user - like$, we need to choose n stories from this playlist randomly. If it is $user - dislike$, we need to choose n-1 stories from this playlist randomly, then choose 1 story out of this playlist randomly and add it to the end of its input item.
(5) In the training set, the size of $user - like$ and $user - dislike$ should be balanced, which means when we generate an $user - like$ list, we must generate an $user - dislike$ list.
(6) In testing set, the size of $user - like$ and $user - dislike$ should be in accord with the real situation. For example, if the length

of a playlist is 100 and we want the input items with a length of 21, the proportion of $user - like$ lists to $user - dislike$ lists should be $100 - (21 - 1) : 700 - 100$, which is equal to $2 : 15$.

(7) The all input items generated from one playlist should be added to either training data or testing data together without separation.

With these rules, we set the length of each input item 5 and we generate 1139430 training items, 215480 testing items. The proportion of $user - like$ and $user - dislike$ lists in testing data is $167 : 113776$. We use python lib $tensorflow$ to training our data. The detailed parameters are:

(1) Set $n\_hidden = 50$, $n\_classes = 2$.
(2) Use $BasicLSTMCell$ and $static_rnn$.
(3) $Logits = Output * W + B$, where $Output$ is the last one in $static\_rnn$ return output list.
(4) Use $softmax\_cross\_entropy\_with\_logits$ and $reduce\_mean$ to calculate loss.
(5) Use $AdamOptimizer$ and set $learning\_rate = 0.0001$ to optimize loss.
(6) Set number of cycle steps 5000.
(7) Set batch size for each step is 1024.
(8) Use $argmax$ to predict the label.

After training this LSTM network, we get a trained graph. Each time we get a previous listenning list with length n, we just need to add all other $700 - n$ story to the list. We use the $700 - n$ new lists as inputs of this trained graph. We compare the $logits$ of each list and get the preference story which is the closest to $[0., 1.]$ and farthest to $[1., 0.]$.

4.4 SlugChat Server

In this section, we list the steps about how to deploy SlugChat Server for story recommondation part.

(1) Deploy a RDS in AWS.
(2) Create two tables $tbl\_stories$ and tbl_logs. Their schemas are shown in Table 2 and 3.
(3) Deploy a EC2 in AWS.
(4) Install an express server in EC2.
(5) Create $tbl\_stories$ post request.
(6) Create a webhook and set the URL of our post request in Dialogflow. Figure 5.
(7) Create story intents in Dialogflow
(8) Add user's says such as *'I want to listen to a story'* or *'I am so boring'*.
(9) Add actions to these intents and use thess actions to call the webhook we create. Figure 6.

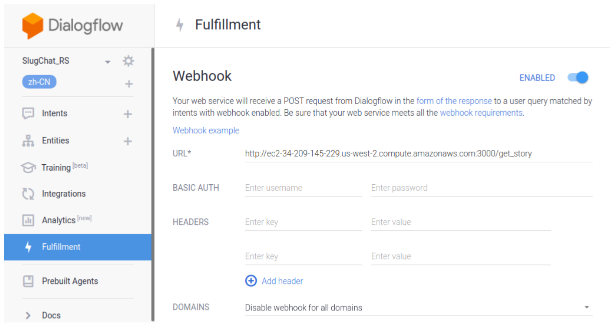
Dialogflow will send a post request to our server when a user active the webhook. This request will send a json package containing the $profileId$ of this user. When recieving this request, our server will query all logs belong to this user from tbl_logs table and get the recent listening list. Then it will return a json package with the content of a preference story to Dialogflow. At the same time, our server will write a new log into tbl_logs table.

Table 2: Schema of Table *tbl_stories*

Field	Type
storyId	int(11)
title	varchar(45)
style	0.varchar(45)
content	text
age3	tinyint(1)
age4	tinyint(1)
age5	tinyint(1)
age6	tinyint(1)
entityName	varchar(45)

Table 3: Schema of Table *tbl_logs*

Field	Type
logId	bigint(20) unsigned
profileId	bigint(20)
createTime	bigint(20)
logType	int(11)
content	tinyint(1)text



Fulfillment

Webhook ENABLED

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the webhook requirements.

Webhook example

URL* http://ec2-34-209-145-229.us-west-2.compute.amazonaws.com:3000/get_story

BASIC AUTH Enter username Enter password

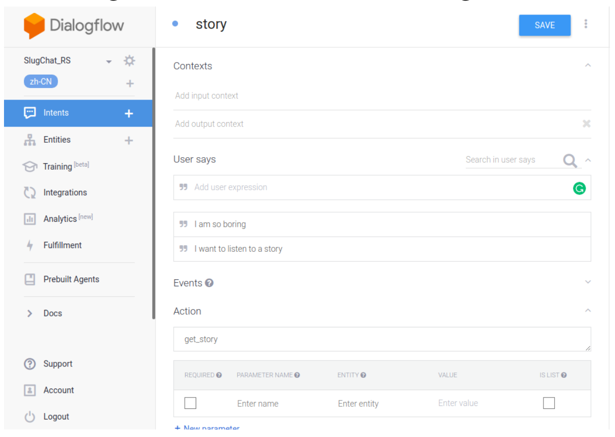
HEADERS Enter key Enter value

Enter key Enter value

Add header

DOMAINS Disable webhook for all domains

Figure 5: Create Webhook in DialogFlow



story

Contexts

Add input context

Add output context

User says

Search in user says

Add user expression

I am so boring

I want to listen to a story

Events

Action

get_story

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST
☐	Enter name	Enter entity	Enter value	☐

+ New parameter

Figure 6: Create Intent in DialogFlow

Table 4: The Top 10 Most Similar Words in Chinese to *watermelon*

Word	Similar Coefficient
big radish	0.940
pear	938
insect	0.937
vegetables	0.937
peach	0.936
gourd	0.934
nut	0.933
candy	0.925
shark	0.924
carrot	0.923

5 EXPERIMENTS & DISSCUSSIONS

5.1 Word Model

Because we don't realize the recommendation algorithm by using word2vec models. We do some experiments to test our word2vec models.
Here are some tests for the performance of our models:

(1) The most similar words to '*watermelon*' are shown in Table 4
(2) The similar coefficient between '*king*' and '*princess*' is *0.845*.
(3) The similar coefficient between '*cock*' and '*princess*' is *0.530*.

5.2 Recommendation Algorithm

Then we calculate the accuracy of our LSTM model when the length of input items is 5. The accuracy of predictions is 92.1%. Then we separate our testing data into 2 groups by their labels. The user-like group has the accuracy with 99.4% and the user-dislike group has the accuracy with 92.0%. It is too high to believe.

We change the length of input items to 6. The overall accuracy is 17.3%, the accuracy of the user-like group is 100% and the accuracy of the user-dislike group is 17.2%. We change the length of input items to 4. The overall accuracy is 99.8%, the accuracy of the user-like group is 5.8% and the accuracy of the user-dislike group is 99.9%.

5.3 Disscussions

The result of recommendation algorithm is really bad. We use the stable length of inputs to train the model. If we user longer inputs, it will put all inputs into the user-like group. It seems that all inputs already have 5 user-like stories and there is no need to consider whether the 6th is user-like or not. If we use shorter inputs, it will put all inputs into the user-dislike group. It seems that even the user likes the last story. No inputs in our data contain enough(5) user-like stories.

6 FUTURE WORK

For data part, we will use SlugChat to collect children's behaviors and use these to replace our old data.
For algorithm part, we will realize the LSTM with unstable lengths of inputs and evaluate its performance. Then we will use Word Mover's Distance to replace Bag of Words to generate new story vectors. Finally, we will use the state of LSTM to replace story vectors.
For SlugChat, we will replace reading stories from a text by an audio. We will also add cold start function before reading stories in order to collect users' information.

REFERENCES

[1] Chinabyte. 2017. 5 robots for kids worth paying for. (2017). http://news.chinabyte.com/295/14160795.shtml.
[2] Sameera A Abdul-Kader and John Woods. Survey on chatbot design techniques in speech conversation systems. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, 6(7), 2015.
[3] Charu C Aggarwal. *Recommender systems*. Springer, 2016.
[4] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.
[5] Arlene Brett, Liz Rothlein, and Michael Hurley. Vocabulary acquisition from listening to stories and explanations of target words. *The Elementary School Journal*, 96(4):415–422, 1996.
[6] West China City Daily. Survey: Over 90% children claim that their parents do not have enough time to accompany them. 2014. http://edu.sina.com.cn/zxx/2014-12-26/0813450854.shtml.
[7] SS Dhenakaran and K Thirugnana Sambanthan. Web crawler-an overview. *International Journal of Computer Science and Communication*, 2(1):265–267, 2011.
[8] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.
[9] Lev Grossman. How computers know what we wantâĂŤbefore we do. *TIME magazine*, 27, 2010.
[10] Francesca GE Happe. The role of age and verbal ability in the theory of mind task performance of subjects with autism. *Child development*, 66(3):843–855, 1995.
[11] Rayna Hollander. Amazon is looking to drive user retention for alexa skills. 2017. http://www.businessinsider.com/ amazon-is-looking-to-drive-user-retention-for-alexa-skills-2017-11.
[12] Hosein Jafarkarimi, Alex Tze Hiang Sim, and Robab Saadatdoost. A naive recommendation model for large databases. *International Journal of Information and Education Technology*, 2(3):216, 2012.
[13] Pauline A Jones. Home environment and the development of verbal ability. *Child Development*, pages 1081–1086, 1972.
[14] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966, 2015.
[15] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.
[16] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.
[17] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.
[18] Michael McTear, Zoraida Callejas, and David Griol. Introducing the conversational interface. In *The Conversational Interface*, pages 1–7. Springer, 2016.
[19] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.
[20] Alexandra Sifferlin. 6-month-old babies are now using tablets and smartphones. 2015. http://time.com/3834978/babies-use-devices/.
[21] Elizabeth A Vandewater, Victoria J Rideout, Ellen A Wartella, Xuan Huang, June H Lee, and Mi-suk Shim. Digital childhood: electronic media and technology use among infants, toddlers, and preschoolers. *Pediatrics*, 119(5):e1006–e1015, 2007.

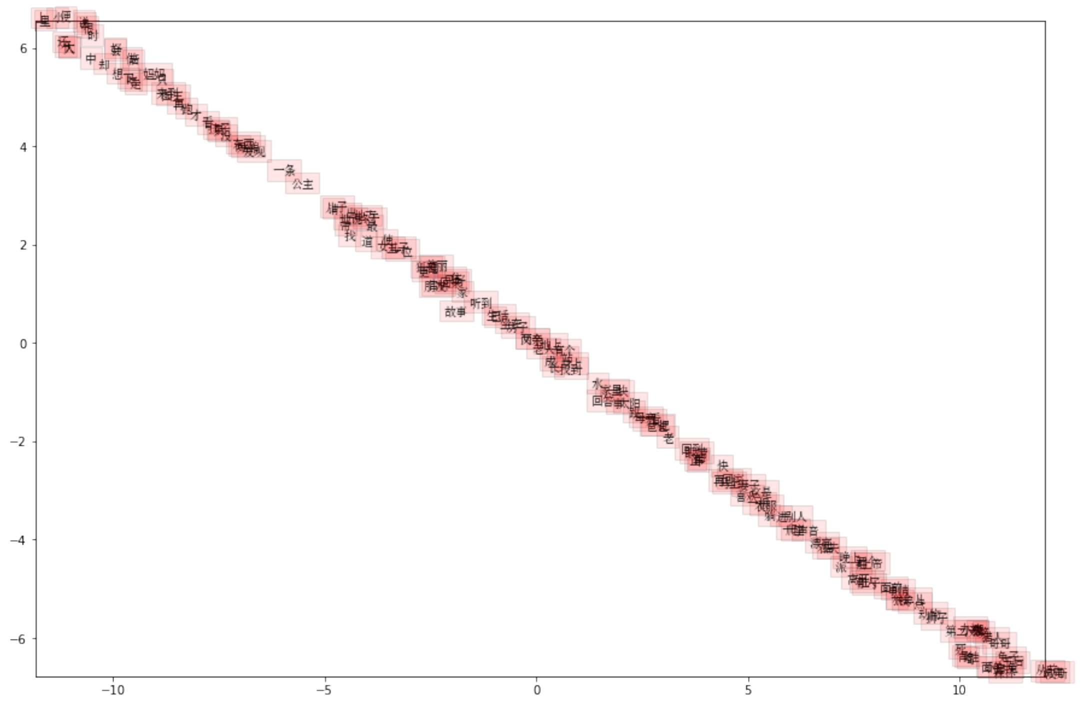


Figure 7: Word2Vec Model Distribution with Small Corpus $min\_count = 1, size = 100$

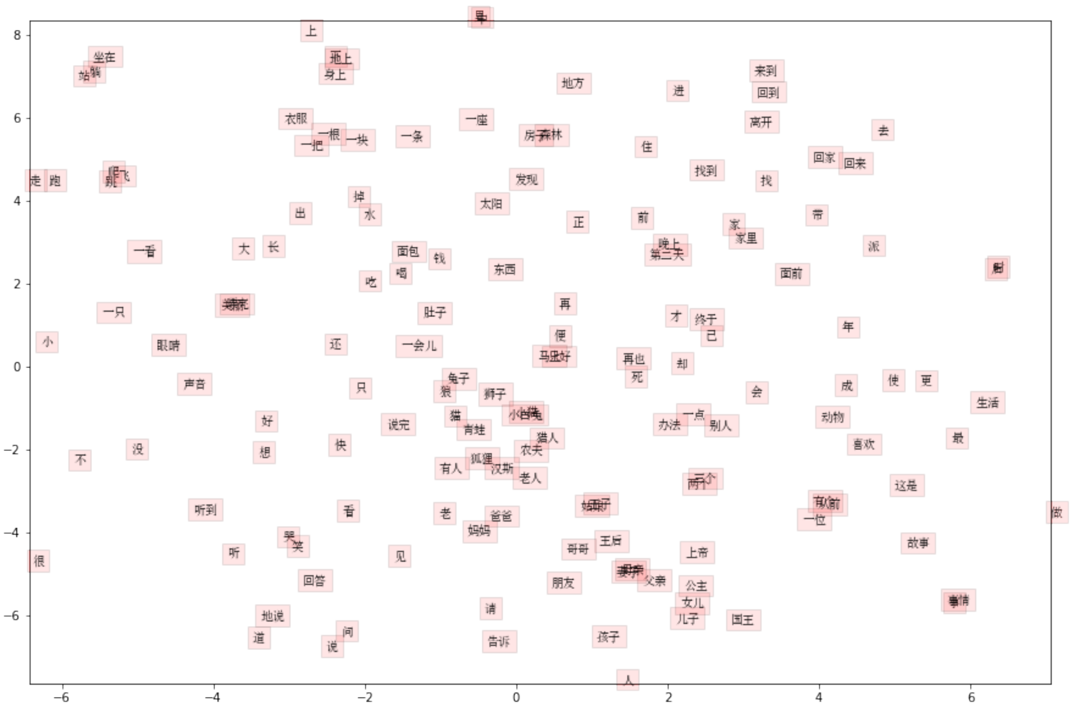


Figure 8: Word2Vec Model Distribution with Small Corpus $min\_count = 1, size = 50$

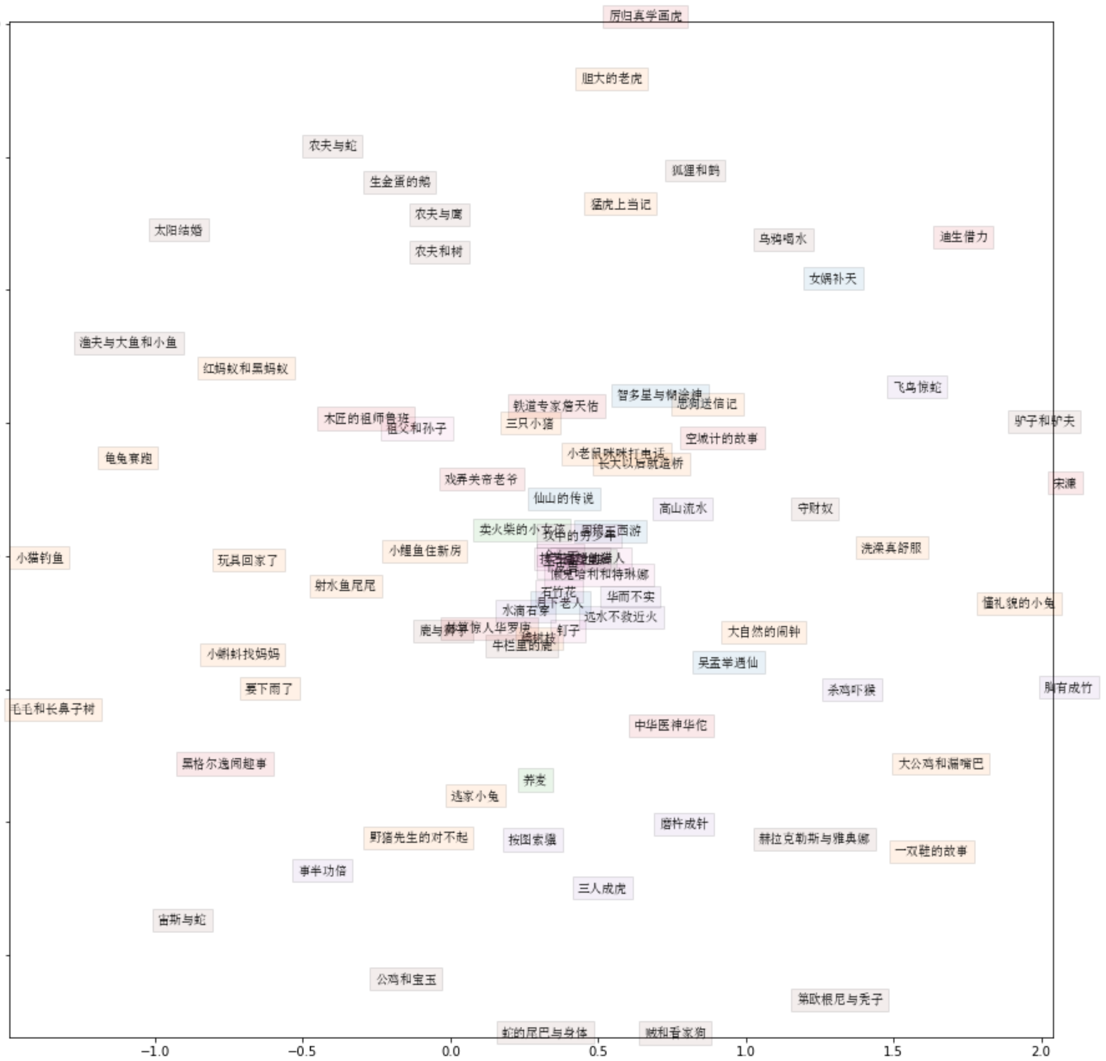


Figure 9: BOW Story Vector Distribution