

Affect-rich Dialogue Generation using OpenSubtitles 2018

Yan Fu^a

^a*Institute of Electrical Engineering, EPFL Lausanne, Switzerland*
{yan.fu}@epfl.ch

June 26, 2019

Abstract—Mainly this report contains three parts: first we introduce three data sets used in this project; then we extract multi-turn dialogues based on sentence similarity and reach $p_k=0.262$; finally, this report also tries to improve basic Seq2Seq dialogue generation by creating an emotional response, and compare the results of affect-rich method with basic ones.

1 Introduction

To learn an open domain dialogue generation model, corpora of huge size are needed. However, for current public data sets, they are not perfect: either they are well labelled but are of small size (e.g. Cornell Movie–Dialogs Corpus, Daily Dialogue), or they have huge dialogues but are specific on certain topics (e.g. Ubuntu Dialogue Corpus).

OpenSubtitles 2018 is a large data set of movie and TV subtitles and includes a total of 446,612 files. And it contains topics on different domains. But the big problem of OpenSubtitles 2018 is that, it is an unlabelled data set, not characters, no scenes and no dialogue boundaries. This causes big trouble when we want to extract multi-turn dialogues to train our model. So the first step is to perform dialogue extraction on OpenSubtitles 2018.

Many researchers use OpenSubtitles as their training set. They extract the sentence with a question mark and the next sentence and consider the pair of sentences is one dialogue^[5]. However, this method ignores the continuity between sentences, and it fails to extract multi-turn dialogue. In this report, we adapt embedding-enhanced texttiling^[2] to perform better segmentation based on the similarity between consecutive sentences. We validate our method on Cornell Movie–Dialogs Corpus, and test it on Scripts data set (both will be introduced in next session) with p_k 0.262 and 0.295 respectively. Then we apply this method on OpenSubtitles 2018, and we will get a corpus of 7 million dialogues at a rough estimation, which is much larger than Cornell Movie–Dialogs Corpus.

Then we improve the basic Seq2Seq model with affect rich approach^[5] and train the model with a small subset of the dialogue extracted from former step. And we find out that with maximum mutual information objective function and VAD embedding, the model can generate a less generic and more affective response. For example, given the input 'are you gonna miss me?', the basic Seq2Seq will respond with 'no.' while our model will say 'i do n't know, sweetheart.'

⁰ Codes and results are in the Github repo: <https://github.com/yf0726/OpenSubtitlesAffectiveDiag>

2 Data set

In this part we introduce the OpenSubtitles 2018 data set, Cornell Movie–Dialogs corpus and scripts data set used in this project

1. OpenSubtitles 2018

We downloaded English subtitles from <http://opus.nlpl.eu/OpenSubtitles2018.php>. The original subtitles on OpenSubtitles is .src file, and the data set downloaded has already converted .src file into .xml file.

Only time stamp and lines are known in the OpenSubtitles data set. No characters, no pre-known dialogue session, even no names of the movies. We have 446,612 .xml files in total. And some of them are trailers, lyrics or lectures (whose duration is either too short (few minutes) or too long (more than 3 hours)), and these files generally do not contain meaningful dialogue. Given that most movies and episodes last between half an hour and three hours, we only use files whose durations are in 0.5 - 3 hours. And finally, we got 349,313 processed subtitles, accounting for 78% of the total.

Mainly there are three main tags in each xml file: <s> tag, <time> tag and <mega> tag. In each <s> tag is a whole sentence, in each <time> tag are words shown in one screen.

- Generally, <s> tag and <time> tag are paired one to one: it means one whole sentence in one screen (as shown in Example 1, the sentence is: What do you know about Wakanda?). But there are some exceptions:
- Multiple <s> tags in one <time> tag: it means short sentences by different character are shown in one screen (as shown in Example 2).
- Multiple <time> tags in one <s> tag: it means the sentence is too long and is split to be shown in multi-screens (as shown in Example 3).

```
<s id="2">
  <time id="T1S" value="00:00:06,000" />
  <w id="2.1">What</w>
  <w id="2.2">do</w>
  <w id="2.3">you</w>
  <w id="2.4">know</w>
  <w id="2.5">about</w>
  <w id="2.6">Wakanda</w>
  <w id="2.7">?</w>
  <time id="T1E" value="00:00:10,000" />
</s>
```

Figure 1: Example1 of OpenSubtitles 2018

In <meta> tag there contains some meta information for each subtitle: number of sentences, number of tokens, language, duration, etc. Note that the information in <meta> tag is not very reliable, on the one hand not all subtitles have full information, on the other hand, the information itself is confusing and even wrong.

For pre-processing, first, we need to extract sentences in tags. And there are some non-speech parts in OpenSubtitles data set (e.g. special symbols (namely nondigit, non-English), brackets and contents in between like [BOOM]). These are not relevant to the dialogue itself so we remove them. And also, we replaced numbers with <NUM>.

2. Cornell Movie–Dialogs Corpus

```

<s id="1">
  <time id="T1S" value="00:00:37,237" />
  <w id="1.1">Man</w>
  <w id="1.2">:</w>
</s>
<s id="2">
  <w emphasis="true" id="2.1">Life</w>
  <w emphasis="true" id="2.2">is</w>
  <w id="2.3">a</w>
  <w emphasis="true" id="2.4">story</w>
  <w emphasis="true" id="2.5">.</w>
  <time id="T1E" value="00:00:39,808" />
</s>

```

Figure 2: Example2 of OpenSubtitles 2018

We validate our method on the Cornell Movie–Dialogs corpus. In Cornell Movie–Dialogs corpus, each sentence is given a unique serial number (LineID), and these numbers are sequenced in time: smaller number means this sentence occurs earlier in the movie. In each row, there are IDs of two speaker, movie ID and then sentence IDs in one dialogue. In the original data set, the dialogues are sorted according to speaker ID (eg. all dialogues of speaker u0 and speaker u2 in movie 0 will are placed at the top.)

Because we want to find dialogue boundaries in consecutive texts, first we re-order the dialogues in Cornell Movie–Dialogs corpus by the serial number of starting sentence of each dialogue. Then we have dialogues in time order.

3. Scripts data set

The authors^[1] collected scripts of corresponding movie or TV in OpenSubtitles 2018 data set. We asked them for the scripts and corresponding subtitles; then we found dialogues boundaries for each subtitle based on the scripts. Finally, we got 946 labeled subtitles, in total of 718524 sentences and 162832 dialogues. And we will also test our segmentation methods on this labeled data set.

3 Dialogue Segmentation

3.1 Problems

As mentioned before, OpenSubtitles data set does not include the information like characters and boundaries of dialogues. So we need to extract dialogues by ourselves to create our dialogue training data set. Here we first introduce concepts used in later part:

- Sentence: Group of words separated by punctuation like . ? ! etc. In OpenSubtitle, we define one sentence as words in same <s> tag.
- Turn: Sentences by one speaker in one dialogue, not interrupted by another speaker.
- Dialogue: Consecutive turns focused on related topics.
- Session: Consecutive turns, we try to extract one dialogue in one session. And in a later experiment, we consider there are ten sentences in one session.

OpenSubtitles does not have turns information neither does it have dialogue boundaries, so to get an ideal data set we will need to:

```

<s emphasis="true" id="17">
  <time id="T21S" value="00:01:48,808" />
  <w id="17.1">Even</w>
  <w id="17.2">the</w>
  <w id="17.3">barkers</w>
  <w id="17.4">pitching</w>
  <w id="17.5">unwinnable</w>
  <w id="17.6">games</w>
  <time id="T21E" value="00:01:51,288" />
  <time id="T22S" value="00:01:51,344" />
  <w id="17.7">for</w>
  <w id="17.8">kewpie</w>
  <w id="17.9">doll</w>
  <w id="17.10">prizes</w>
  <w id="17.11">do</w>
  <w id="17.12">so</w>
  <w id="17.13">with</w>
  <w id="17.14">all</w>
  <w id="17.15">the</w>
  <w id="17.16">joie</w>
  <w id="17.17">de</w>
  <w id="17.18">vivre</w>
  <w id="17.19">of</w>
  <w id="17.20">a</w>
  <w id="17.21">cancer</w>
  <w id="17.22">patient</w>
  <w id="17.23">.</w>
  <time id="T22E" value="00:01:56,794" />
</s>

```

Figure 3: Example3 of OpenSubtitles 2018

SpeakerID1	SpeakerID2	MovielD	LineIDs
0	u0	u2	m0
1	u0	u2	m0
2	u0	u2	m0
3	u0	u2	m0
4	u0	u2	m0
5	u0	u2	m0
6	u0	u2	m0
7	u0	u2	m0
8	u0	u2	m0

SpeakerID1	SpeakerID2	MovielD	LineIDs
0	u0	u3	m0
1	u8	u9	m0
2	u2	u7	m0
3	u2	u7	m0
4	u2	u7	m0
5	u2	u7	m0
6	u5	u8	m0
7	u5	u8	m0
8	u5	u6	m0

Figure 4: Original Cornell Movie–Dialogs corpus

- First perform the turn segmentation, find out characters for each sentence;
- Second perform dialogue segmentation to extract multi-turn dialogues.

Most work using OpenSubtitles notice these two problems but do not solve it. Generally, they simply consider each sentence as one turn, and they take sentence ending with a question mark as input, the sentence next to input as output. Some constraints may be added, varying from different researchers, like the time interval between input and output should be less than 20 seconds, either sentence should be less than 20 words, etc. Few works were found on the first problem. *Automatic turn segmentation for Movie TV subtitles*^[1] introduced a supervised method on turn segmentation. Aside OpenSubtitles 2018, they scraped scripts of corresponding subtitles and used the character information in scripts as the label. They scraped 7000+ scripts, and then performed sentence alignment to align sentence in OpenSubtitles and those in scripts, only 34% movies and 60% episodes were successfully aligned. Finally, they annotated 5,413 subtitles and got 1,521,382 sentences labeled with the speaker. Then they extracted features like timing, punctuation, lexical

features, visual features, length features, adjacency features, edit distance features etc and used the linear classifier. They got an accuracy of 78.1% on the test set. However, this method is too time-consuming: first, we need to find out the name of subtitles, which is not included in OpenSubtitles data set, and we will have to search for movie names from the original OpenSubtitles website, download the .src file and then convert it to the .xml file. Then, web scraping and parsing are needed for scripts. Thirdly, sentence alignment is needed, and the inconsistency of scripts and subtitles add even more difficulties to this task (scripts online are usually an early version of drafts, and are usually quite different from the final subtitle). Due to the limited time, we hold the simple assumption that each sentence is a turn.

For the second problem, here we adapted methods in *Dialogue Session Segmentation by Embedding-Enhanced TextTiling*^[2]. Details are shown in the next subsection.

3.2 Solutions

We redefine the dialogue segmentation as following: given the beginning sentence of one dialogue and consecutive sentences of the beginning sentence (name as 'a session'), we want to find one dialogue boundary in that session so that context from starting sentences to the sentence just before the boundary can form one meaningful dialogue. And here we regard sentence ending with a question mark as the beginning sentence.

3.2.1 Time-based segmentation

As mentioned above, one simple idea is that sentences in a short time interval should be relevant and thus has a higher probability to belong to the same dialogue. And in the OpenSubtitles data set, we have a timestamp for each sentence. So in one session, if the time gap (which means the gap between two sentences' time stamp) of the beginning sentence and sentence A is larger than the threshold, then sentence A will be considered as the dialogue boundary.

However, there is no strict guideline in how to choose the time threshold; and this method is also hard to be evaluated as I did not find labeled open-source data set with time stamp.

3.2.2 Similarity-based segmentation

Here we adapted the embedding-enhanced texttiling^[2]. Simply speaking, the basic idea is to calculate the similarity between consecutive sentences, cut off where the similarity is low so that consecutive sentences with high similarity will be considered as topic relevant dialogue. And still, first we assume that sentence ending with a question mark is the beginning of each dialogue, then the problem is converted to find the boundary of dialogue in consecutive lines after the beginning sentence. First, we need to convert words to vectors. Then we need to define the sentence level similarity. For two words we can compute their cosine distance, the closer to 1 the more similar they are; and the opposite for -1. One intuition is to average all word vectors in one sentence and use the averaged vector as the representative vector for this sentence, and then calculate the cosine distance between two sentence vectors as the similarity between two sentences.

There is another definition of sentence similarity. For similarity we followed the best measure in that paper, the heuristic max similarity between sentence 1 (S_1) and sentence

2 (S_2):

$$sim(S_1, S_2) = \frac{1}{n_1} \sum_{i=1}^{n_1} \max_{j=0}^{n_2} \{cos(w_i, v_j)\}$$

More detailedly, n_1 is the number of words in sentence 1, w_i is the i -th word in sentence 1; n_2 and v_j means the same for sentence 2. So for heuristic max similarity, we find the most related word in s_2 for each word in s_1 , and then the sentence similarity is the average of words in s_1 . The authors^[2] said this is better than sum pooling method (first get the vectors for s_1 and s_2 by averaging words in them, and then calculate the cosine similarity of that two vectors) which has a blurring side effect.

Then we calculated the depth score. For example, now we have one session beginning with question mark sentence. And we have calculated the similarity between neighboring sentences in that session, the depth score of the i -th sentence in that session will be the difference of the i -th similarity and maximal similarity up to the i -th. Then the boundary of dialogue should be where the depth score is low.

There are two ways to find the boundary. First is using a threshold. The threshold is defined as:

$$th = \mu - \alpha * \sigma$$

where μ and σ are the mean and standard deviation of session depth score. α is a hyperparameter. We tuned α on Cornell dialogue data set, where dialogue boundaries are given.

Another idea is to cut off the session at the lowest point of depth score. Since there will be multi-lowest points in one session, we return the index of the first lowest point.

We combine the result of threshold cut-off and lowest point cut-off. Because we want to have related sentences as long as possible, so the final prediction is the maximum of threshold cut-off and lowest point cut-off. For example, we have ten sentences in a session; threshold cut-off says the boundary is at sentence 7, and at sentence 3, 6 and 8 there are the local lowest points. So the final prediction will be $\max(\min(3,6,8),7) = 7$.

3.3 Metrics

Another problem is to define the metric to evaluate the quality of segmentation. Even though the dialogue segmentation can be somewhat considered as classification task (label 0 inside one dialogue as there is no boundary; label 1 between two dialogues as there is a boundary), standard classification metrics(e.g. accuracy) are not very suitable in our cases. In this section, we will show why and discuss few commonly used metrics in text segmentation task.

3.3.1 Classification-Based Metrics

As mentioned in the former section, the segmentation task can be seen as a classification task: we consider each sentence as one sample, and only predict 1 where the sentence is considered as the boundary of two dialogues (sentence itself is the beginning of next dialogue, and the sentence before it is the ending of current dialogue) and 0 elsewhere.

However, such a metric is not appropriate in our cases. In the following table, we show a simple example. We have 10 sentences and 3 ground truth boundaries, which mean that

S_1 to S_4 , S_5 to S_8 are two whole dialogues and S_9 is ground truth boundary and beginning of the third dialogue. And row pred1 and pred2 are two predictions. We can easily see that pred1 and pred2 have same accuracy, but intuitionally, pred1 should be better than pred2: each boundary of pred1 is only 1 sentence away from ground truth, while in pred2 the prediction is far away from the ground truth.

Besides, unlike common text segmentation task, where the aim is to segment all possible sub-topics as many as possible, our task aims to find the end of dialogue given its beginning sentence. That is to say, we now know S_1 is the beginning of one dialogue, and we have 8 sentences following S_1 (S_2 to S_{10}), we want to find the boundary S_i , so that S_1 to S_i can make up one meaningful dialogue. Under the assumption that different dialogue have different topics, and topic in one dialogue does not change a lot, the word 'meaningful' here refers that the topics in S_1 to S_i should be relevant. And still, pred1 is better than pred2. In pred 1, though S_4 is missed, S_1 to S_3 is still focusing on same topic; in pred 2, S_1 to S_6 is predicted to be one dialogue, but S_5, S_6 actually belong to another dialogue. Thus it is worse. So when predicting dialogue boundary, a boundary earlier is better than a later one, and accuracy cannot reflect such information.

Sentences	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
GroundTruth	1	0	0	0	1	0	0	0	1	0
Pred1	1	0	0	1	0	0	0	1	0	0
Pred2	1	0	0	0	0	0	1	0	0	1

Table 1: An example binary classification evaluation

So accuracy is not suitable in our situation: though ideally, we want to predict exact boundary as ground truth, predictions near to ground truth are better than those far away; and predictions earlier than ground truth is tolerable. So in the next section we introduce different metrics.

3.3.2 Segmentation-Based Metrics

1. Hinge-like loss

Here we use a self-defined loss (hinge-like loss, which means it works similar to hinge loss). If the prediction is earlier than ground truth, then the loss is the distance between ground truth label and predicted label; if the prediction is later than ground truth then the loss will be the length of the session. It works as a punishment for late prediction.

For example, ground truth label is at sentence 6 and prediction is at sentence 4, then the loss will be 2, and if the prediction is at sentence 8 and session length is 10, then the loss will be 10. The smaller the loss is, the closer the prediction is to ground truth. Though hinge-like loss is easily understood, it heavily depends on the session size: hinge-like loss will be smaller on the small session but larger on the big session, thus causing difficulties in comparing results.

2. P_k

To combat this problem, we use P_k (proposed by Beeferman et al. (1999)), which expresses a probability of segmentation error: the average probability, given two points in the data set, that the predictor is incorrect as to whether they are separated by a boundary or not.

(Note that as P_k scores are probabilities, they range between 0 and 1, but a higher score means less accurate: a higher probability of error). To calculate P_k , we take a window of fixed width k and move it across the session, at each step examining whether the prediction is correct about the separation (or not) of the two ends of the window. For a single window position with start i and end j , we can express this separation via the indicator function $\delta_S(i, j)$:

$$\delta_S(i, j) = \begin{cases} 1 & \text{if predict that } S_i \text{ and } S_j \text{ are in same dialogue} \\ 0 & \text{else} \end{cases}$$

For a single window (i, j) , the error of a predicted segmentation P relative to a reference segmentation R can then be calculated as: And the error is:

$$\delta_P(i, j) \oplus \delta_R(i, j)$$

where \oplus is logic operator XOR, $A \oplus B$ equals to 1 if and only if A and B are different, else $A \oplus B$ equals to 0.

And we obtain P_k by moving the window and summing it up:

$$P_k = \frac{\sum_{i=1}^{N-k} \delta_P(i, i+k) \oplus \delta_R(i, i+k)}{N-k}$$

The choice of k is arbitrary, but it is generally set to be half the average segment length in the reference segmentation R . This value ensures (under some assumptions) that the four obvious baseline algorithms (hypothesizing no boundaries, boundaries everywhere, evenly-spaced boundaries or randomly-spaced boundaries) all have $P_k = 0.5$. A perfect segmenter will score 0, of course; a score of 1.0 will only be achieved by a truly terrible segmenter which manages to hypothesize boundaries in all and only the wrong places.

3.4 Training

3.4.1 Virtual sentence

The authors^[2] proposed the concept of virtual sentences to train word embeddings for conversation data.

Sheriff: Hannah, was that Holtz?
Hannah: Yes, Sheriff, it was.
Sheriff: Thank you.
Hannah: You 're welcome.

'Virtual sentences' means concatenating two consecutive sentences as one, and train word embedding on these virtual sentences. For example, 'Hannah, was that Holtz? Yes, Sheriff, it was.' is one virtual sentence concatenating 'Hannah, was that Holtz?' and 'Yes, Sheriff, it was.' The motivation for training on virtual sentences is that, in dialogues, consecutive sentences have stronger interaction. One common example is that 'You're welcome' is a very common response to 'Thank you' but rare is the situation that the word 'welcome' will come up with 'thank you' in one sentence. But constructing virtual sentences can solve this

problem (e.g. we have the virtual sentences 'Thank you. You're welcome.'). Considering that our method is to cut off texts where successive similarity is low, the concept of a virtual sentence is believed to be useful to learn common question-response pair.

3.5 Results

First, we compare the results of different word embedding (with or without virtual sentence; a lower case for all words or not; removing punctuation or not).

In the below table, we show the results of predictions using different word embeddings, and all are made with $\alpha = 0.5$, session length = 10. We can see all predictions are much better than random prediction. Using virtual sentence improve the performance slightly. And taking high accuracy, low hinge-like loss and low P_k into account, we finally choose to embed method 6 as the best method.

Embedding	ACC	Hinge	P_k	Random ACC	Random Hinge	Random P_k
1	0.307	4.229	0.268	0.100	6.204	0.501
2	0.311	4.209	0.265	0.101	6.198	0.499
3	0.305	4.144	0.264	0.102	6.167	0.498
4	0.316	4.188	0.265	0.099	6.19	0.498
5	0.314	4.2	0.266	0.098	6.202	0.498
6	0.314	4.105	0.262	0.100	6.152	0.501

Table 2: All predictions are made under $\alpha = 0.5$, session length = 10

Embedding Method	Details of embedding
1	Basic embedding
2	Use virtual sentence
3	Use virtual sentence and remove punctuation
4	Lower all words
5	Lower all words and use virtual sentence
6	Lower all words and use virtual sentence and remove punctuation

Table 3: Details for each embedding method

Then we look into how different α affects prediction performance.

alpha	ACC	Hinge	P_k	Random Hinge	Random Hinge	Random P_k
0.1	0.305	3.723	0.271	0.101	6.349	0.503
0.2	0.309	3.804	0.267	0.098	6.344	0.503
0.3	0.310	3.918	0.266	0.099	6.293	0.501
0.4	0.311	4.013	0.264	0.096	6.253	0.501
0.5	0.314	4.105	0.262	0.102	6.143	0.499

Table 4: Results of different alphas

Higher alpha means lower threshold in depth score, and if the depth score is too low and all scores in the session are higher than the threshold ($\mu - \alpha * \sigma$), then we set the prediction at the tail of this session. So higher alpha will cause later prediction and thus higher hinge-like loss. However, higher alpha also causes higher accuracy and lower P_k . We need to balance between them. And here we choose alpha = 0.5 (the same value picked by text tiling method in nltk package).

On scripts data set, using alpha = 0.5 and embedding 6, we got the result of accuracy 0.290, hinge loss 4.342 and p_k 0.295 on average, which are very close to that of Cornell data set.

data set	alpha	embedding	ACC	Hinge	P_k
Cornell Movie–Dialogs Corpus	0.5	method 6	0.314	4.105	0.262
Scripts data set	0.5	method 6	0.290	4.342	0.295

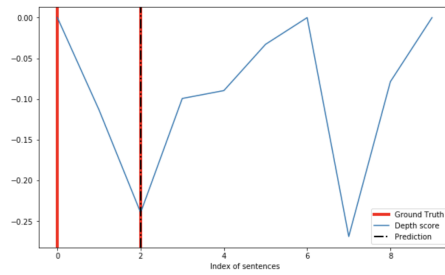
Table 5: Comparison of Segmentation Results on two data sets

Then we show some results on the Cornell Movie–Dialogs corpus. In each image, the first red line is the beginning of the dialogue, the second red line is the dialogue boundary (beginning of next dialogue), and the black dash line is the predicted boundary. The blue plot is the depth score of the session. And in each image, we also show the index and content of sentence of that session. For example, in the next figure we show a perfect prediction, which predicts sentence 2-nd as dialogue boundary. So the extracted dialogue will be:

Turn 0: That ca n’t be right ; it ’s only twenty-six light years away .
Turn 1: I scanned it at Arecibo ; negative results , always .

Then we show examples of early prediction as well as late prediction. Though it is not perfect, an early prediction is tolerable.

And in original text tiling, smoothing for depth score is needed. However the smoothing is not doing good in our case. And this time we add the orange plot showing the result of smoothing. Using smoothing we can get the general trend of the depth score, but it will cause dislocation when we are looking for the lowest point. For example, the lowest point should be at 2-nd sentence, but after smoothing, the position moves to 4-th sentence. Also, metrics can show how badly smoothing is doing (with accuracy 0.272, hinge loss 4.314 and P_k 0.274, all worse than without smoothing).



0 That ca n't be right ; it 's only twenty-six light years away .

1 I scanned it at Arecibo ; negative results , always .

2 How 's the spying tonight , guys ?

3 NORAD 's not tracking any spacecraft in our vector including snoops ; shuttle Endeavor 's in sleep mode .

4 Okay , let 's just slow down . Pull up the starfield signal origin .

5 It ca n't be coming from Vega , the system 's too young .

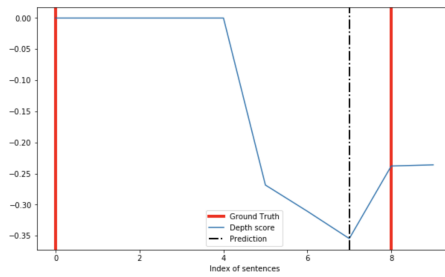
6 Ca n't we get rid of them ?

7 It 's a civilian facility .

8 Colonel Jarrod , I 'd like a twenty mile radio-silent perimeter put around this installation immediately .

9 And a hundred mile airspace .

Figure 6: An example of accurate prediction.



0 Halloran .

1 You betrayed me ! Now every psychopath in the city knows I 'm back in business ... You lied to me !

2 I did not ; the Mouth -- that 's what we call Susan Schiffer -- got it on her own .

3 Why should I trust you ?

4 Because I 'm all you 've got .

5 It 's a woman shot in a car ?

6 Yes . I have to go ...

7 She on the passenger side ?

8 Helen , hang up , let Ruben get on with his work ...

9 What 's that music . It 's Abba . I can hear it . It 's Abba .

Figure 7: An example of early prediction.

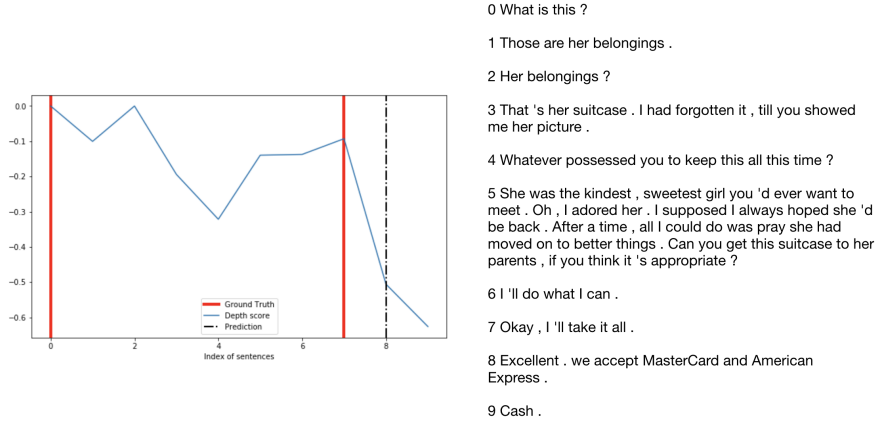


Figure 8: An example of late prediction.

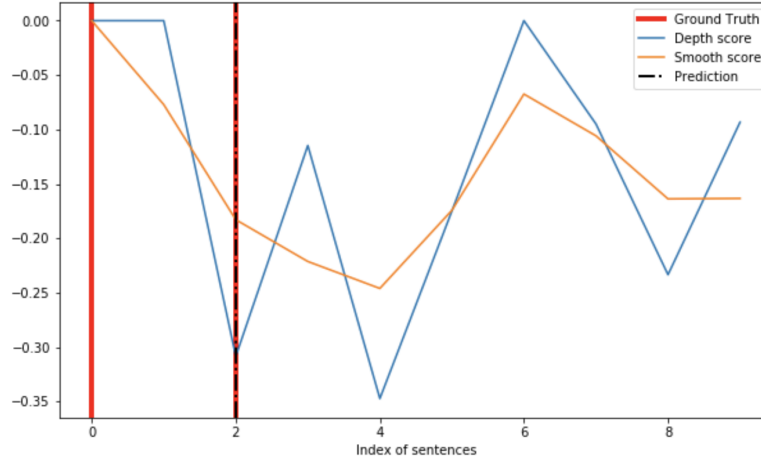


Figure 9: An example of dislocation caused by smoothing

4 Affect-rich dialogue generation

4.1 Affect-rich Dialogue generation model

Seq2Seq based conversational model (abbreviation for sequence to sequence) is widely used to generate open-domain dialogues. It can learn semantic and syntactic relations. However, basic Seq2Seq conversational model tends to generate safe and highly-generic replies (e.g. 'I do not know.') regardless of the input, which is not wanted in a dialogue system, because it closes a conversation.

Also, it is hard for basic Seq2Seq model to capture and generate emotional sentences, and affect-rich approach is needed. And to avoid general replies, maximum mutual information is used in the beam search strategy.

4.1.1 Affect-rich approach

The idea of affect-rich dialogue generation^[5] is to extend the Seq2Seq model adopting VAD affective notations to embed each word with effects. Besides, an affect-incorporated objective function is used during training to encourage affect-rich response.

VAD (Valence, Arousal and Dominance) notation is widely used as a representation of human emotions in psychology. We adopted the annotated lemma-VAD pairs corpus^[6] to encode word affects. This corpus comprises 13,915 lemmas with VAD values annotated in the [1, 9] scale. And we extended the basic VAD corpus to 36,267 lemmas by assigning the average VAD values of their synonyms to absent lemmas. Then VAD values of all words are clipped to the [3, 7] interval to prevent extreme values. And for words not in the extended VAD corpus, neural VAD values [5, 3, 5] are assigned to them.

1. Affective Embedding

We incorporate VAD embedding by concatenating VAD value after word embedding:

$$e(x_t) = [\mathbf{x}_t; \lambda \overline{VAD}(x_t)]$$

where \mathbf{x}_t is the word embedding of the word x_t , $\overline{VAD}(x_t) = VAD(x_t) - [5, 3, 5]$, λ is a scalar denoting the strength of VAD embeddings.

2. Affective Attention

The authors^[5] assume that human pay more attention on affect-rich word and incorporate affect into attention as follows:

$$\begin{aligned} e_{t't} &= \mathbf{h}_t^T \mathbf{s}_{t'} + \eta_t \\ \eta_t &= \gamma |\mu(x_t)(1 + \beta) \otimes \overline{VAD}(x_t)|_2^2 \\ \beta &= \tanh(\mathbf{W}^b \mathbf{x}_{t-1}) \end{aligned}$$

where \otimes denotes element-wise multiplication, \mathbf{W}^b is a model parameter, β is a scaling factor in V, A, D dimension, γ controls the magnitude of affect bias, and $\mu(x_t)$ measures the term importance of x_t .

3. Term Importance

Term importance is used to compute the importance of a word. And in this report, we adopt 'local importance', which is calculated as following:

$$\mu(x_t) = \frac{\log(1/p(x_t) + \epsilon)}{\sum_{t=1}^{t=T} \log(1/p(x_t) + \epsilon)}$$

where $p(x_t)$ denotes the term frequency of x_t in training corpus, ϵ is a small constant with value 10^{-8} .

4. Affective Objective Function

To encourage the generation of affect-rich words, the authors incorporate VAD embedding of words into loss:

$$\Psi_t = -|V| \frac{1 + \delta |\overline{VAD}(y_t)|_2}{\sum_{\hat{y}_t \in V} (1 + \delta |\overline{VAD}(\hat{y}_t)|_2)}$$

where y_t denotes the target token at decoding time step t , V denotes the data set vocabulary, and δ regulates the contribution of VAD embedding in loss function.

4.1.2 Maximum Mutual Information

To avoid generic response, maximum mutual information (MMI)^[7] objective function is adopted in this project.

The standard objective of Seq2Seq model is to maximize the log-likelihood of target T given source S :

$$\hat{T} = \operatorname{argmax}\{\log p(T|S)\}$$

However, generic responses have high probability in this objective function. And to avoid this, we try to balance between the probability of sources given targets and that of targets given sources:

$$\hat{T} = \operatorname{argmax}\{(1 - \lambda)\log p(T|S) + \lambda\log p(S|T)\}$$

And the new objective is intractable since $\log p(S|T)$ can only be calculated after target prediction. So here we split the objective into two steps (named as 'MMI-bidi'): first we use beam search to find N-best responses given $\hat{T} = \operatorname{argmax}\{\log p(T|S)\}$; then for N responses we calculate the probability $p(S|T)$ and re-rank them.

4.2 Training

In this part we introduce training set and parameters used during the training. Ideally, we will use the result of OpenSubtitles dialogue segmentation in previous part as our training set. However, due to the limit of computation ability of my laptop, here we do not train with multi-turn dialogues but only two-turn dialogues (only the first two turns in each dialogue are used). And we choose dialogues with turns that are neither too long nor too short (word counts between 3-20). Finally we have 46400 dialogues in training set, and 9984 dialogues in test set. The hyper-parameters are chosen as below:

- $\lambda = 0.1$ (for affective embedding)
- $\gamma = 0.5$ (for affective attention)
- $\delta = 0.15$ (for the affect objective function)
- beam search width = 32
- learning rate = 0.001

4.3 Results

In this section we will show the result of dialogue generation.

$P(T S)$	Responses	$P(S T)$	Responses
-6.457921	my name.	-6.353932	what 's wrong with you .
-6.605519	all right.	-9.697718	oh , my god . my name 's .
-6.686670	i do n't know .	-12.913490	oh , my god . that 's ...
-6.767957	oh , my god .	-13.057776	oh , my god . my name .
-6.977804	maybe .	-13.782521	my name is sherlock holmes .

Table 6: Input: 'what 's your name ?'

From table 6 we can see that re-order the beam search result with $P(S|T)$ can avoid generic answers like 'I do n't know'. For each question we have 32 answer candidates, and we re-order them by $P(S|T)$, take the top-10 answers, and then among those 10 answer candidates we select the one with highest VAD value as the final response to given question. In the following table we compare the results of responses maximizing $P(T|S)$ (basic generation) and responses balancing between $P(T|S)$, $P(S|T)$ and VAD value(MMI+ affect-rich generation).

input	Basic generation	MMI+ affect-rich generation
what 's your name ?	my name .	i do n't know . i was just looking for my friend .
why did i agree to this ?	i do n't know .	i do n't know , but i did n't know why i was talking about it .
are you gonna miss me ?	i am .	i do n't know , sweetheart .
does that mean his parents are still alive ?	no .	well , it does n't matter what he says .
what am i doing down here ?	what are you doing ?	well , i thought you were n't supposed to be right here .

Table 7: Basic generation vs. MMI+ affect-rich generation

Though the predictions are not perfect (because we only run on small training set with very few epochs), we can still see that¹ using MMI+ affect-rich we can generate less generic, longer and more affective responses.

References

- [1] Lison P, Meena R. Automatic turn segmentation for movie and tv subtitles[C]//2016 IEEE Spoken Language Technology Workshop (SLT). IEEE, 2016: 245-252.
- [2] Song Y, Mou L, Yan R, et al. Dialogue session segmentation by embedding-enhanced texttiling[J]. arXiv preprint arXiv:1610.03955, 2016.
- [3] Hearst M A. TextTiling: Segmenting text into multi-paragraph subtopic passages[J]. Computational linguistics, 1997, 23(1): 33-64.
- [4] Purver M. Topic segmentation[J]. Spoken language understanding: systems for extracting semantic information from speech, 2011: 291-317.
- [5] Zhong P, Wang D, Miao C. An Affect-Rich Neural Conversational Model with Biased Attention and Weighted Cross-Entropy Loss[J]. arXiv preprint arXiv:1811.07078, 2018.
- [6] Warriner, A. B.; Kuperman, V.; and Brysbaert, M. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. Behavior Research Methods 45(4):11911207.
- [7] Li J, Galley M, Brockett C, et al. A diversity-promoting objective function for neural conversation models[J]. arXiv preprint arXiv:1510.03055, 2015.

¹ More examples are available in the 'results/predictions.csv' file.