

A Descriptive Text Auto-Generation and Evaluation

Honglin Bao, Jiaoping Chen

Content

1. Introduction and Motivation
2. Data Preprocessing
3. Problem definition
4. Machine learning: multi-class classification and generation
5. Evaluation: bleu and coherence
6. Future work

1. Introduction and Motivation

• Problem of Modeling Language

- Natural language emerge, hard to specify
 - Natural language involves many terms that can be used in multiple ambiguities scenarios but can still be understood by other people
 - Word usages change across time, which indicates new words are induced or same word might have additional meaning.
 - Formal grammars and structures are not powerful enough to specify the language because they might fail to capture other information like semantic syntactics.

→ learn the language model from examples

• (Statistical) Language Modeling (LM)

Use probabilistic models to predict the next word in the sequence given the words that precede it

- learn the relative frequency of word co-occurrence based on sample texts, such as N-grams
 - $P(w_n|w_1, w_2, \dots, w_{n-1})$: a parameter of an N-gram model. Estimate those parameters we use word occurrence frequencies. $C(w_i)$ is the count of occurrence of w_i .
 - Unigram: $P(w_i) = \frac{C(w_i)}{N}$; Bigram: $P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$; Trigram: $P(w_i|w_{i-2}w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})}$
- LM can be used speech recognition, handwriting recognition...
- LM can be used independently, such as generate new sequences of text that come from the corpus → our task
- However, LM is difficult to obtain true generalization in a discrete word-indices space due to the lack of obvious relations between the word indices.

1. Introduction and Motivation

- **Natural Language Modeling (NLM)**

- **Word embeddings**

- represents each word as a real-valued vector to solve the n-gram data sparsity issue (that is, most possible word sequences are not observed in training even though we assume the probability of a word only depends on the previous n words).
 - (Kim et al. 2015) shows that word embeddings obtained in NLMs own the property that semantically close words are likewise close in the induced vector space.

- **Recurrent neural networks**

- allow to learn how to represent memory because neurons with input from recurrent connections represent short term memory, which leads to better performance.
 - (Tomas Mikolov et al. 2010) provides empirical evidence to suggest that feed-forward RNN-LMs are superior to standard n-gram approach, except for their high computational training complexity
 - (Tomáš Mikolov et al. 2011) shows the importance of backpropagation through time algorithm to reduce the number of parameters, and obtain a smaller, faster both during training and testing, and more accurate than feed-forward RNN-LM.
 - (Jozefowicz et al. 2016) evaluates the language models over large datasets, such as the corpus of one million words, and conclude that LSTM-based NLM outperform the classical method such as carefully tuned n-grams.
 - GPT-2 (a successor to GPT), was trained simply to predict the next word in 40GB of Internet text. It indicates the tasks can benefit from unsupervised techniques, given sufficient (unlabeled) data and compute.

→ we will utilize the *LSTM-based NLM approach* to implement one of language modeling tasks, the text generation task.

2. Data Preprocessing

- **1. Book** *The Republic* by Plato
- Contains 10 chapters, and each chapter is structured as a dialog on the topic of order and justice.
 - *“I agree, he said, because I do not wish to quarrel with you.*
 - *How good of you, I said; but I should like to know also whether injustice, having this tendency to arouse hatred, wherever existing, among slaves or among freemen, will not make them hate one another and set them at variance and render them incapable of common action?...”*
- ASCII text version from [Project Gutenberg website](http://www.gutenberg.org/files/13/13-h/13-h.htm)
- 15,802 lines of text
- **2. Wikipedia extractor**
- `res = requests.get('https://es.wikipedia.org/wiki/Andorra/' + ' '.join(sys.argv[1:]))`
- Data preprocessing:
 - split words based on white space, remove all punctuations, remove words that are not alphabetic and normalizing all words to lowercase....
 - a smaller vocabulary set with many many tokens

3. Problem Definition and Solution

- **Problem definition**

sequence of n words, for example, 50, is the input and the output is a word which is the $n+1$ item of the sequence. It, in other words, must learn which is the next most likely word given a sequence of words.

use the same model recursively with output passed as input (50 words to 50 words), the language model will be statistical and will predict the probability of each word given an input sequence of text. The predicted word will be fed in as input to in turn generate the next word.

- **Solution**

- **The general idea is to use an Embedding Layer to learn the representation of words, and a Long Short-Term Memory (LSTM) to learn to predict words based on their context.**

4. Neural Network Model

- **Word Embedding (tokenization)**

- Before implementing word embedding for each input sentence, we would first map input words to integers. So that we could convert the prediction to numbers and then check its corresponding words using the same mapping. Therefore, we firstly identify all unique words in data and assign each word a unique integer, and then convert all word-based sequences into a list of integers.
- Word embeddings is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. They are vector representations of particular words. As a high-dimensional sparse vector in which all elements are integers. Each cell in the vector represents a separate English word; the value in a cell represents the number of times that word appears in a sentence.

input (selected line of sentence, split the data into separate training sequences by splitting based on new lines and white space) -> embedding layer (integer vector representation of tokens)

Multiclass classification

- One-of-many classification. Each sample can belong to ONE of C classes. The deep network will have C output neurons that can be gathered in a vector s (Scores). The target (ground truth) vector t will be a one-hot vector with a positive class and $C-1$ negative classes. This task is treated as a single classification problem of samples in one of C classes.

- `y = to_categorical(y, num_classes=vocab_size)`
- total number of classes = vocab size.
- we need to one hot encode the output word.
- This means converting it from an integer to a vector of 0 values, one for each word in the vocabulary, with a 1 to indicate the specific word at the index of the words integer value.
- why? model learns to predict the probability distribution for the next word and the ground truth from which to learn from is 0 for all words except the actual word that comes next.

LSTM

embedding layer (integer vector representation of tokens) -> split input and output (array slicing)

Keras provides the `to_categorical()` that can be used to one hot encode the output words for each input-output sequence pair.

-> fit model and train (LSTM, the goal of training is categorical cross entropy loss)

```
model = Sequential()
model.add(Embedding(vocab_size, 50, input_length=seq_length))
model.add(LSTM(100, return_sequences=True))
model.add(LSTM(100))
model.add(Dense(100, activation='relu'))
model.add(Dense(vocab_size, activation='softmax'))
print(model.summary())
# compile model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# fit model
model.fit(X, y, batch_size=128, epochs=100, validation_split=0.1, show_accuracy=True, verbose=1)

# save the model to file
model.save('model.h5')
```

Layer (type)	Output Shape	Param #
=====		
embedding_1 (Embedding)	(None, 50, 50)	370500
<hr/>		
lstm_1 (LSTM)	(None, 50, 100)	60400
<hr/>		
lstm_2 (LSTM)	(None, 100)	80400
<hr/>		
dense_1 (Dense)	(None, 100)	10100
<hr/>		
dense_2 (Dense)	(None, 7410)	748410
=====		
Total params: 1,269,810		
Trainable params: 1,269,810		
Non-trainable params: 0		
<hr/>		

5 Results & Evaluation

Reference	Candidate
when he said that a man when he grows old may learn many things for he can no more learn much than he can run much youth is the time for any extraordinary toil of course and therefore calculation and geometry and all the other elements of instruction which are	preparation for dialectic should be presented to the name of idle spendthrifts of whom the other is the manifold and the unjust and is the best and the other which delighted to be the opening of the soul of the soul and the embroiderer will have to be said at
the two philosophers had more in common than they were conscious of and probably some elements of plato remain still undetected in aristotle in english philosophy too many affinities may be traced not only in the works of the cambridge platonists but in great original writers like berkeley or coleridge	other has knowledge are greatest as come on with be conversation to and perhaps any time to platonic keep but by as aristocracies as ideal acknowledge either every sympathy probably become find no so as other words to other century platonic by as well same numbers seem irregular if state

Evaluation

We are applying two kinds of evaluation for text generation.
We call this 'cross text evaluation' and 'intrinsic text evaluation'

Why cross-text?

we generate a description through some text inputs, so we want to know the relationship between input and output.

Why intrinsic text?

the candidate text is artificially generated, we want to quantify the coherence score to evaluate the quality of output.

BLEU

Proceedings of the 40th Annual Meeting of the Association for
Computational Linguistics (ACL), Philadelphia, July 2002, pp. 311-318.

BLEU: a Method for Automatic Evaluation of Machine Translation

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu

IBM T. J. Watson Research Center

Yorktown Heights, NY 10598, USA

{papineni,roukos,toddward,weijing}@us.ibm.com

We consider an improved version

1. modified n-gram precision on blocks of text

$\text{Count}_{\text{clip}} = \min(\text{Count}, \text{Max_Ref_Count})$

2. add the clipped n-gram counts for all the candidate sentences and divide by the number of candidate n-grams in the test corpus to compute a modified precision score for the entire test corpus

$$p_n = \frac{\sum_{C \in \{\text{Candidates}\}} \sum_{n\text{-gram} \in C} \text{Count}_{\text{clip}}(n\text{-gram})}{\sum_{C' \in \{\text{Candidates}\}} \sum_{n\text{-gram}' \in C'} \text{Count}(n\text{-gram}')} .$$

final bleu is around 0.24

3. sentence brevity penalty

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} .$$

Evaluate Coherence

- **Sentence coherence VS Topic coherence**
- Sentence coherence [Perplexity]:
 - Specifically, the inverse probability of the test set, normalized by the number of words. (The smaller the better.)
 - It can be regarded as a branching factor (the weighted average number of choices a random variable has).
 - But it is also an exponential of cross entropy but assume all words have the same probability (1 / # of words) in p.

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}} = \sqrt[N]{\prod \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$

$$\begin{aligned} PP(W) &= \exp\left(-\sum P(w_i) * \ln q(w_i)\right) \\ &= \exp\left(-\sum \frac{1}{N} * \ln q(w_i)\right) \\ &= q(w_1)^{-\frac{1}{N}} q(w_2)^{-\frac{1}{N}} * \dots * q(w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{q(w_1) \dots q(w_N)}} \end{aligned}$$

where $q(\cdot)$ is a proposed probability model, our prediction.

Evaluate Coherence

- **Sentence coherence VS Topic coherence**
- Topic coherence
 - Coherence: words can support each other for specified topic. That is, words from the same topic should be well classified by the ideal model.
 - A good model will generate coherent topics, i.e., each topic with high topic coherence score.
 - The coherence scores are mainly comparative: if topic B has a higher coherence score than topic A, it is 'better' (more coherent).
 - Example of a coherent fact set:
"the game is a team sport", "the game is played with a ball", "the game demands great physical efforts"
- **Past research has shown that predictive likelihood (or equivalently, perplexity) and human judgment are often not correlated, and even sometimes slightly anti-correlated [1].**
- **In this project, we'll explore more about Topic Coherence, and how we can use it to quantitatively justify the model selection.**

[1] Hu, Yuening, et al. "Interactive topic modeling." *Machine Learning* 95.3 (2014): 423-469.

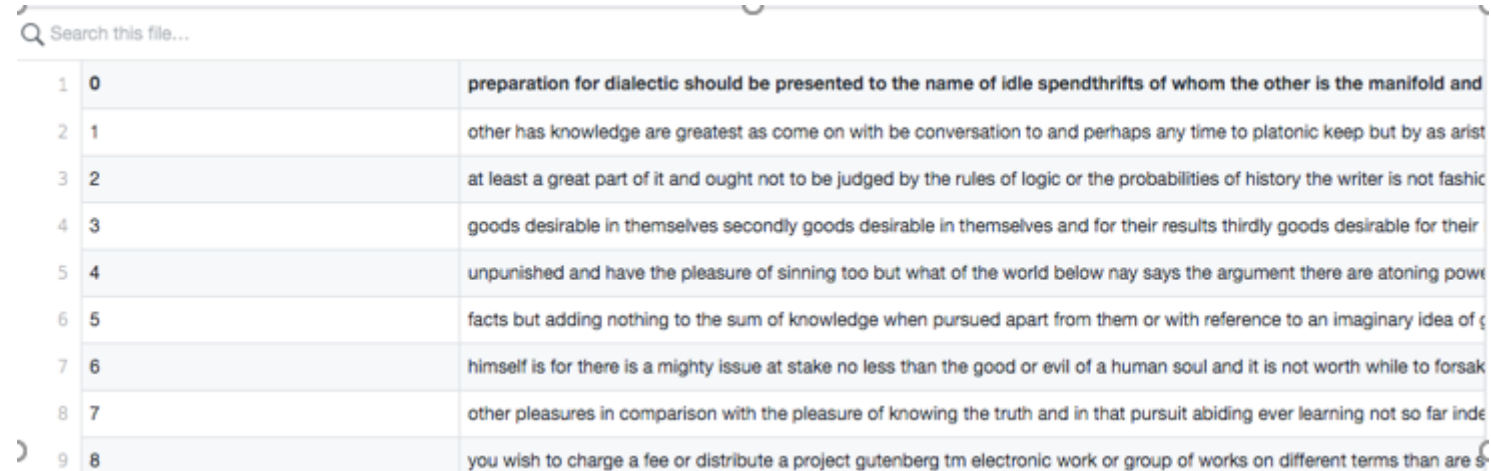
Evaluate Coherence

- Topic coherence Measurements: a topic with four words {game, sport, ball, team}

UCI measure (Newman et al., 2010)	<ul style="list-style-type: none"> - pointwise mutual information (PMI) - sliding window - external corpus - one-set split 	$PMI(w_i, w_j) = \log \frac{P(w_i, w_j) + \epsilon}{P(w_i) * P(w_j)}$ $C_{UCI} = \frac{2}{N \cdot (N - 1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^N PMI(w_i, w_j)$	$C_{UCI} = \frac{1}{6} \cdot (PMI(game, sport) + PMI(game, ball) + PMI(game, team) + PMI(sport, ball) + PMI(sport, team) + PMI(ball, team))$
UMass measure (Mimno et al., 2011)	<ul style="list-style-type: none"> - document co-occurrence - original corpus - one-preceding split 	$C_{UMass} = \frac{2}{N \cdot (N - 1)} \sum_{i=2}^N \sum_{j=1}^{i-1} \log \frac{P(w_i, w_j) + \epsilon}{P(w_j)}$	$C_{UMass} = \frac{1}{6} \left(\log(P(sport game)) + \log(P(ball game)) + \log(P(ball sport)) + \log(P(team game)) + \log(P(team sport)) + \log(P(team ball)) \right)$
NPMI measure Aletras and Stevenson (2013a)	<ul style="list-style-type: none"> - NPMI (an increase of r gives higher NPMI values more weight) - sliding window - external corpus - one-set split 	Replace PMI to NPMI in C_UCI $NPMI(w_i, w_j)^\gamma = \left(\frac{\log \frac{P(w_i, w_j) + \epsilon}{P(w_i) \cdot P(w_j)}}{-\log(P(w_i, w_j) + \epsilon)} \right)^\gamma$	Replace PMI to NPMI in C_UCI
Coefficient of Variance measure Roder et al. (2015)	<ul style="list-style-type: none"> - NPMI - Sliding window - internal corpus - one-set split - Cosine similarity 	Replace PMI to NPMI in C_UCI Combine cosine measure with the NPMI and the <u>boolean sliding window</u>	$C_{cos} = \frac{1}{6} \cdot (cos(\vec{v}_{game}, \vec{v}_{sport}) + cos(\vec{v}_{game}, \vec{v}_{ball}) + cos(\vec{v}_{game}, \vec{v}_{team}) + cos(\vec{v}_{sport}, \vec{v}_{ball}) + cos(\vec{v}_{sport}, \vec{v}_{team}) + cos(\vec{v}_{ball}, \vec{v}_{team}))$

Experiment

- Data Preparation



1	0	preparation for dialectic should be presented to the name of idle spendthrifts of whom the other is the manifold and
2	1	other has knowledge are greatest as come on with be conversation to and perhaps any time to platonic keep but by as arist
3	2	at least a great part of it and ought not to be judged by the rules of logic or the probabilities of history the writer is not fashic
4	3	goods desirable in themselves secondly goods desirable in themselves and for their results thirdly goods desirable for their
5	4	unpunished and have the pleasure of sinning too but what of the world below nay says the argument there are atoning powe
6	5	facts but adding nothing to the sum of knowledge when pursued apart from them or with reference to an imaginary idea of c
7	6	himself is for there is a mighty issue at stake no less than the good or evil of a human soul and it is not worth while to forsak
8	7	other pleasures in comparison with the pleasure of knowing the truth and in that pursuit abiding ever learning not so far inde
9	8	you wish to charge a fee or distribute a project gutenberg tm electronic work or group of works on different terms than are s

- Tokenization and cleaning

- Creating Bigram and Trigram Models

- Bigrams are two words frequently occurring together in the document. Trigrams are 3 words frequently occurring. e.g., Michigan_State_University

```
u'set', u'say', u'other', u'have', u'knowledge', u'dialectical', u'set', u'same', u'civilization', u'numbers', u'preparation  
u'dialectical', u'seem', u'irregular', u'greatest', u'other', u'come', u'time', u'idle', u'so', u'become', u'phraseology',  
u'conversation', u'issue', u'perplexity', u'issue', u'system', u'preparation', u'say', u'dialectical', u'perhaps',  
u'separation', u'any', u'time', u'platonic', u'so', u'preparation', u'keep', u'aristocracy', u'sum', u'ideal', u'acknowledge  
u'either', u'remarkable_issue', u'manifold', u'knowledge', u'set', u'spendthrift', u'sympathy', u'idle', u'spendthrift',  
u'embroider', u'probably', u'ridiculous', u'become', u'find', u'no', u'so', u'speculation', u'word', u'other', u'century',
```

Experiment

Now we have every basis ready

- We use Coefficient Variance Measure (C_v) with default sliding window (set as 110)
 - C_v is based on a sliding window, one-set segmentation of the top words and an indirect confirmation measure that uses NPMI and the cosine similarity
- We also construct two main inputs for the LDA topic model
 - Dictionary (id2word)
 - The internal corpus (denoted by the format of “word index - word frequency”)

Experiment

```
82 # Build LDA model
83 lda_model = gensim.models.LdaMulticore(corpus=corpus,
84                                         id2word=id2word,
85                                         num_topics=3,
86                                         random_state=100,
87                                         chunksize=100,
88                                         passes=10,
89                                         per_word_topics=True)
90
91 # Compute Coherence Score
92 coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized, dictionary=id2word, coherence='c_v')
93 #coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized, dictionary=id2word, coherence='c_p')
```

- num_topics is a hyper parameter which is hard to adjust. There are many works teaching you skills to set a proper prior value of topics.
- final C_v= 0.19

6. Conclusion&Future Works

Conclusion:

- Prepare text for developing a language model
- Design and fit a neural language model with a learned embedding and LSTM
- Generate new text from the source text using the model
- Apply two kinds of evaluation for text generation results.

Future works:

- Can we create a better sentence coherence measurement?
- Can we apply this novel coherence as the fitness of evolutionary search?
- Generate more text as population.
- Current supervision is to minimize cross entropy. We should think about combining loss and coherence to create a new optimization target.

Thanks

Q & A