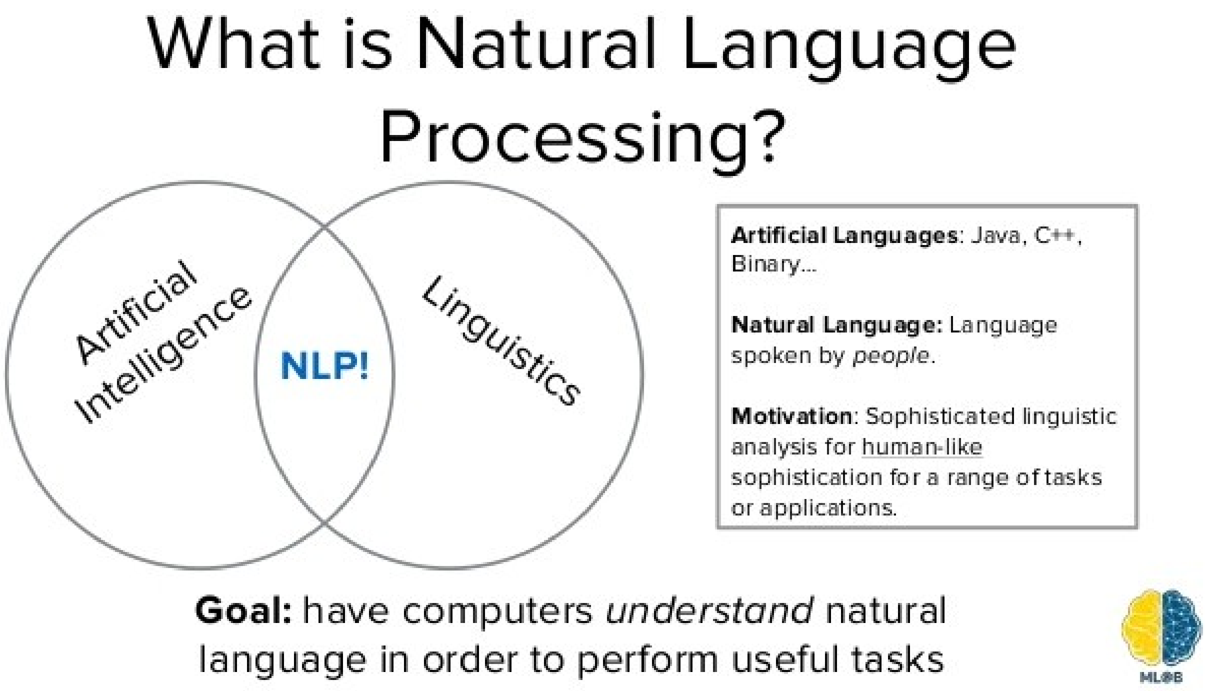
Machine Learning Using Tensorflow

Week 8:
Natural Language Processing

Shu-Ting Pi, PhD
UC Davis

TensorFlow™

What is Natural Language Processing



What is Natural Language Processing?

Artificial Languages: Java, C++, Binary...

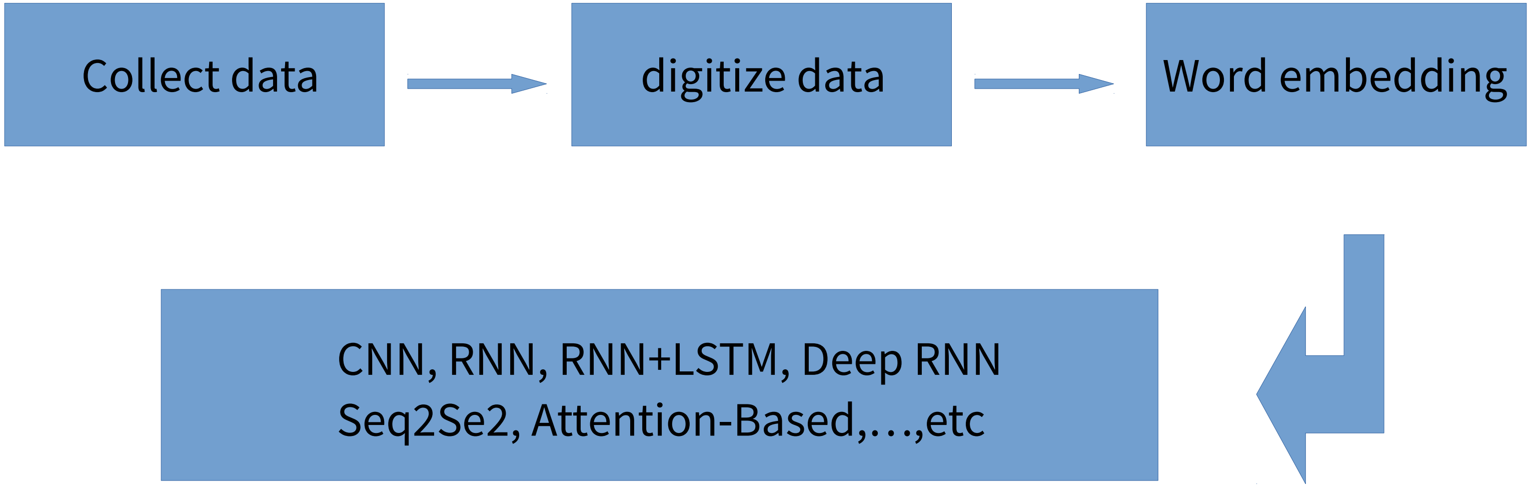
Natural Language: Language spoken by *people*.

Motivation: Sophisticated linguistic analysis for human-like sophistication for a range of tasks or applications.

Goal: have computers *understand* natural language in order to perform useful tasks

Check : https://en.wikipedia.org/wiki/Natural_language_processing

How to perform simple NLP in deep learning?



Collect data

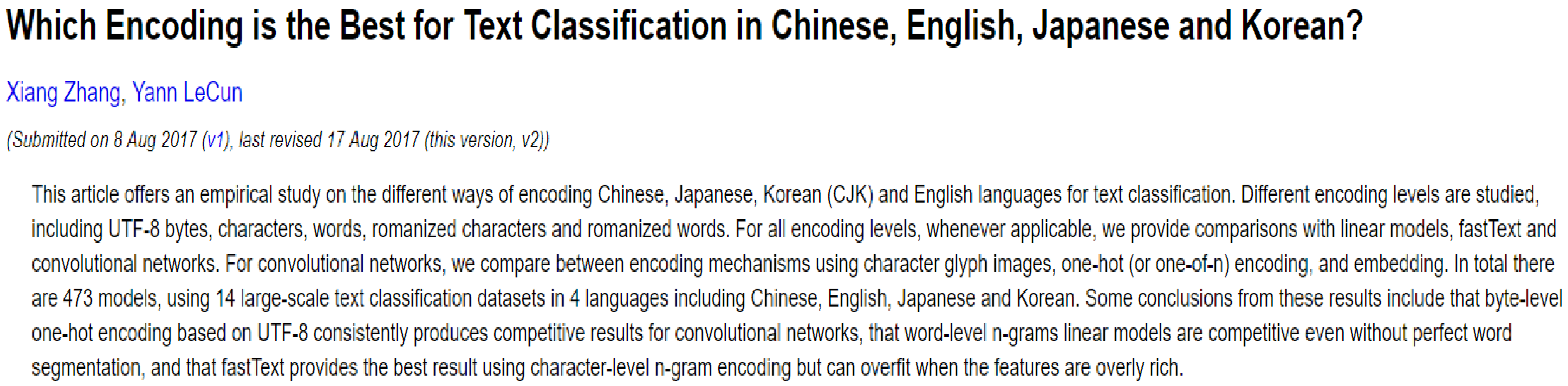
digitize data

Word embedding

CNN, RNN, RNN+LSTM, Deep RNN
Seq2Se2, Attention-Based,…,etc

You could publish a paper by using different combinations !

Models, Models, …, and Models !



Which Encoding is the Best for Text Classification in Chinese, English, Japanese and Korean?

Xiang Zhang, Yann LeCun

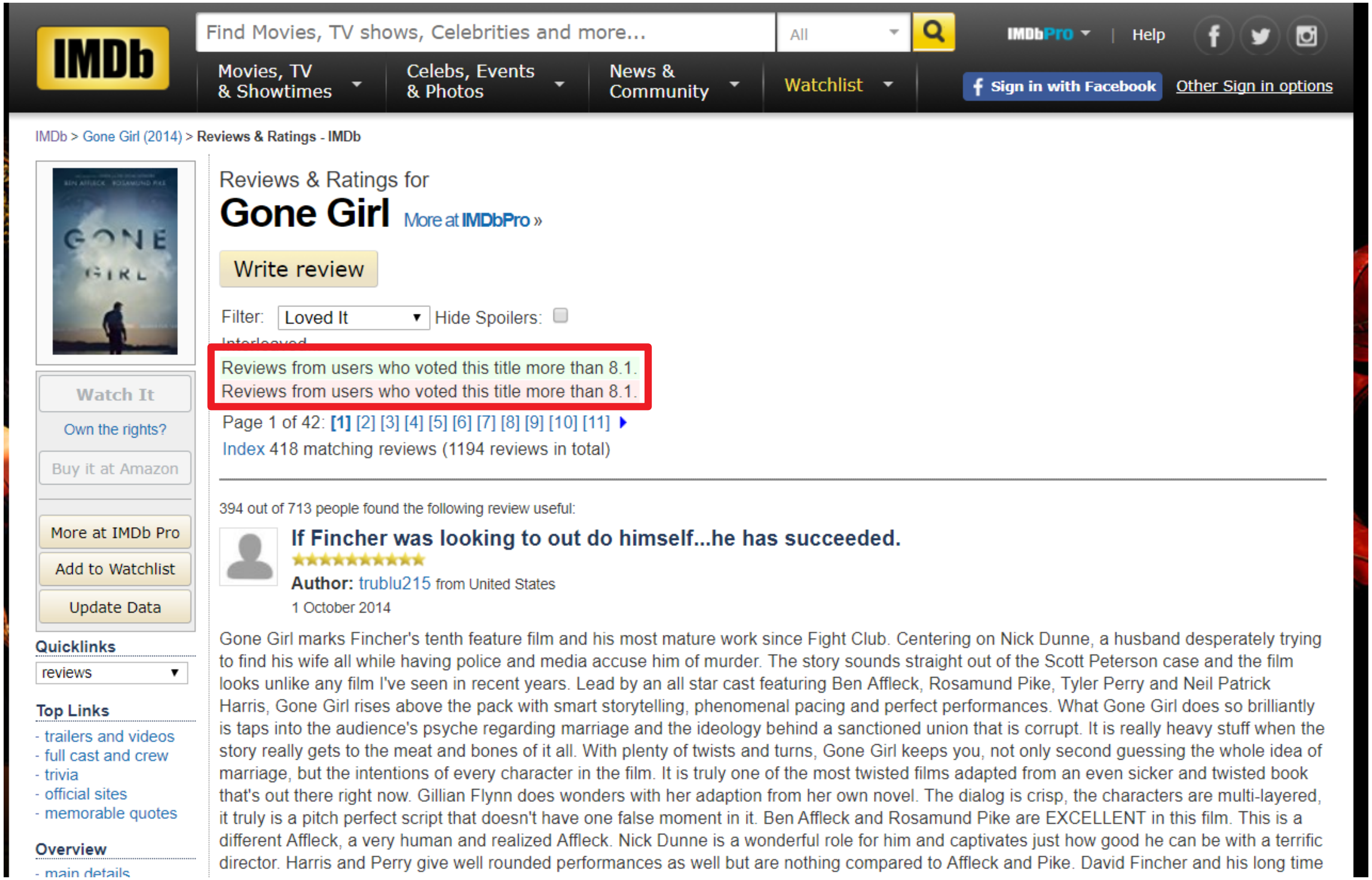
(Submitted on 8 Aug 2017 (v1), last revised 17 Aug 2017 (this version, v2))

This article offers an empirical study on the different ways of encoding Chinese, Japanese, Korean (CJK) and English languages for text classification. Different encoding levels are studied, including UTF-8 bytes, characters, words, romanized characters and romanized words. For all encoding levels, whenever applicable, we provide comparisons with linear models, fastText and convolutional networks. For convolutional networks, we compare between encoding mechanisms using character glyph images, one-hot (or one-of-n) encoding, and embedding. In total there are 473 models, using 14 large-scale text classification datasets in 4 languages including Chinese, English, Japanese and Korean. Some conclusions from these results include that byte-level one-hot encoding based on UTF-8 consistently produces competitive results for convolutional networks, that word-level n-grams linear models are competitive even without perfect word segmentation, and that fastText provides the best result using character-level n-gram encoding but can overfit when the features are overly rich.

4 languages, 473 encoding models on CNN !

Digitalize your data

IMDB database



IMDb > Gone Girl (2014) > Reviews & Ratings - IMDb

Reviews & Ratings for

Gone Girl More at IMDbPro »

Filter: Loved It Hide Spoilers: □

Interleaved

Reviews from users who voted this title more than 8.1.

Reviews from users who voted this title more than 8.1.

Page 1 of 42: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11]

Index 418 matching reviews (1194 reviews in total)

394 out of 713 people found the following review useful:

If Fincher was looking to out do himself...he has succeeded.

★★★★★★★★★★

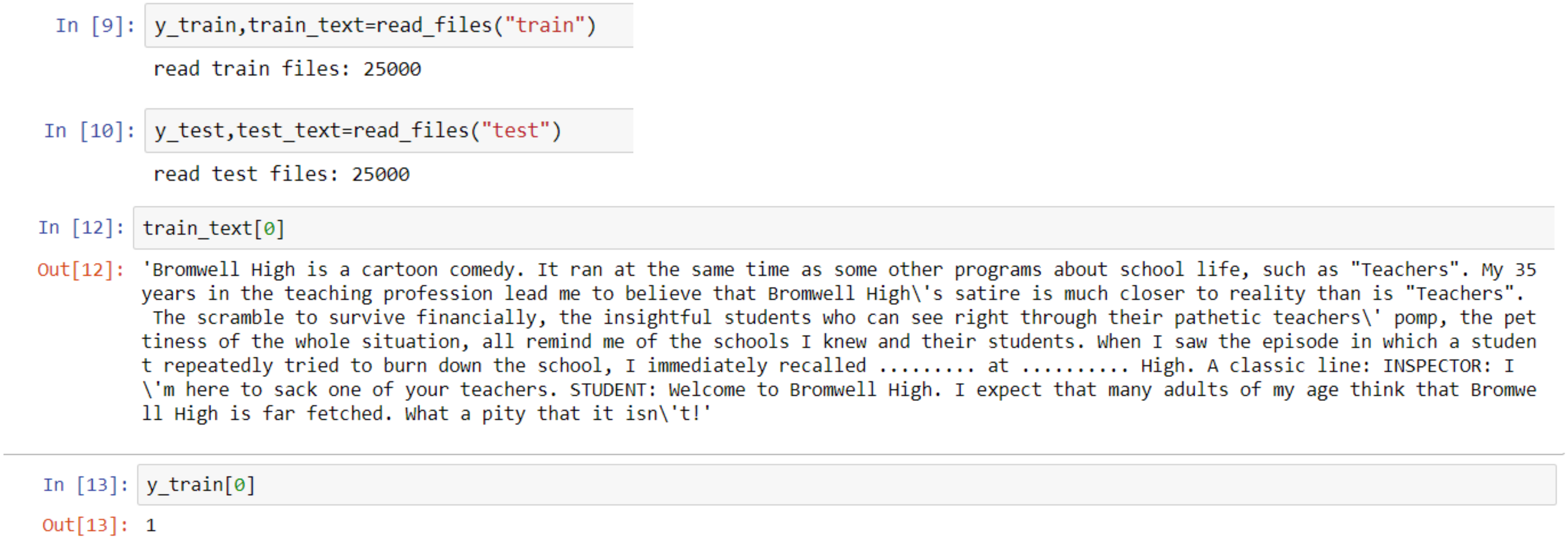
Author: trublu215 from United States

1 October 2014

Gone Girl marks Fincher's tenth feature film and his most mature work since Fight Club. Centering on Nick Dunne, a husband desperately trying to find his wife all while having police and media accuse him of murder. The story sounds straight out of the Scott Peterson case and the film looks unlike any film I've seen in recent years. Lead by an all star cast featuring Ben Affleck, Rosamund Pike, Tyler Perry and Neil Patrick Harris, Gone Girl rises above the pack with smart storytelling, phenomenal pacing and perfect performances. What Gone Girl does so brilliantly is taps into the audience's psyche regarding marriage and the ideology behind a sanctioned union that is corrupt. It is really heavy stuff when the story really gets to the meat and bones of it all. With plenty of twists and turns, Gone Girl keeps you, not only second guessing the whole idea of marriage, but the intentions of every character in the film. It is truly one of the most twisted films adapted from an even sicker and twisted book that's out there right now. Gillian Flynn does wonders with her adaption from her own novel. The dialog is crisp, the characters are multi-layered, it truly is a pitch perfect script that doesn't have one false moment in it. Ben Affleck and Rosamund Pike are EXCELLENT in this film. This is a different Affleck, a very human and realized Affleck. Nick Dunne is a wonderful role for him and captivates just how good he can be with a terrific director. Harris and Perry give well rounded performances as well but are nothing compared to Affleck and Pike. David Fincher and his long time

http://ai.stanford.edu/~amaas/data/sentiment/ train: 2500, test: 25000

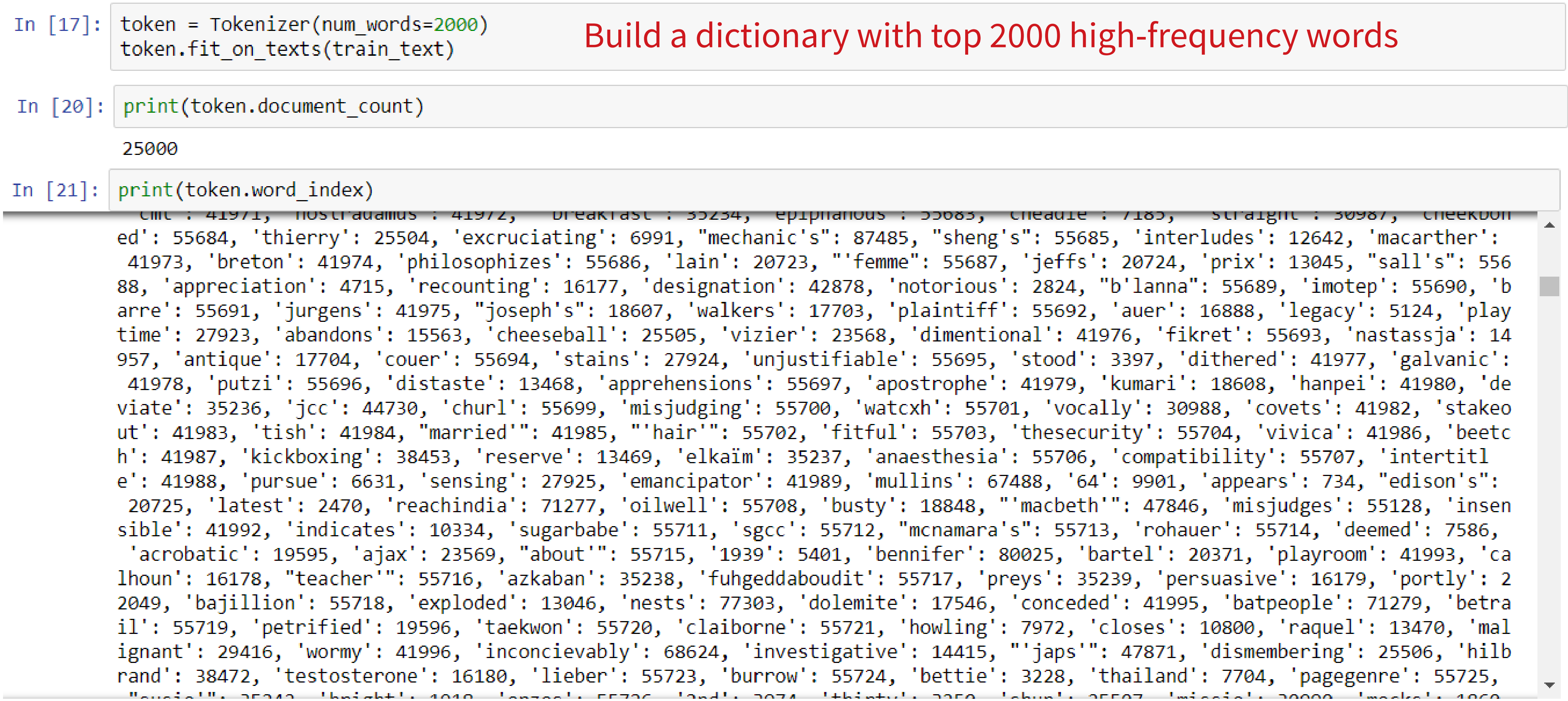
Take a look on the dataset



```
In [9]:  y_train,train_text=read_files("train")

         read train files: 25000

In [10]: y_test,test_text=read_files("test")

         read test files: 25000

In [12]: train_text[0]

Out[12]: 'Bromwell High is a cartoon comedy. It ran at the same time as some other programs about school life, such as "Teachers". My 35
         years in the teaching profession lead me to believe that Bromwell High\'s satire is much closer to reality than is "Teachers".
          The scramble to survive financially, the insightful students who can see right through their pathetic teachers\' pomp, the pet
         tiness of the whole situation, all remind me of the schools I knew and their students. When I saw the episode in which a studen
         t repeatedly tried to burn down the school, I immediately recalled ......... at .......... High. A classic line: INSPECTOR: I
         \'m here to sack one of your teachers. STUDENT: Welcome to Bromwell High. I expect that many adults of my age think that Bromwe
         ll High is far fetched. What a pity that it isn\'t!'

In [13]: y_train[0]

Out[13]: 1
```

Positive comment: +1, Negative comment: -1

Build Dictionary



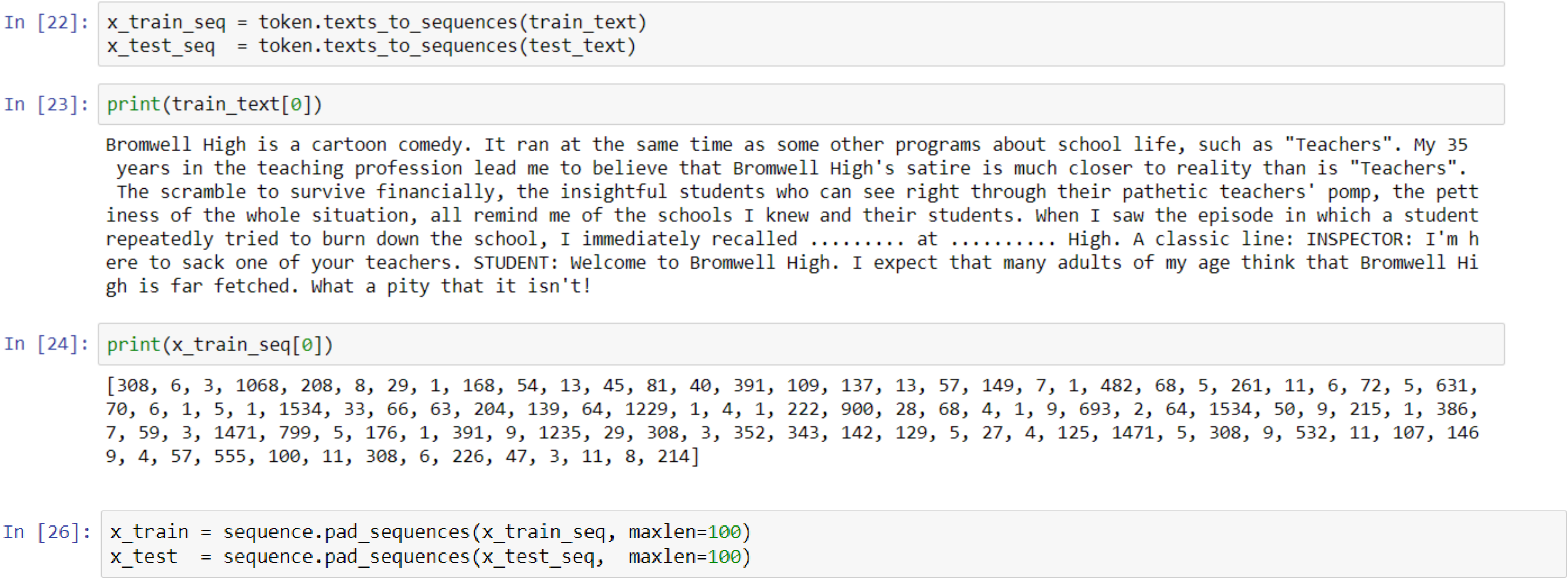
```
In [17]: token = Tokenizer(num_words=2000)
         token.fit_on_texts(train_text)
```

Build a dictionary with top 2000 high-frequency words

```
In [20]: print(token.document_count)

         25000

In [21]: print(token.word_index)
```

```
 'cmt': 41971, 'nostradamus': 41972, 'breakfast': 35234, 'epiphanous': 55683, 'cheadle': 7185, 'straight': 30987, 'cheekbon
ed': 55684, 'thierry': 25504, 'excruciating': 6991, "mechanic's": 87485, "sheng's": 55685, 'interludes': 12642, 'macarther':
 41973, 'breton': 41974, 'philosophizes': 55686, 'lain': 20723, "'femme": 55687, 'jeffs': 20724, 'prix': 13045, "sall's": 556
88, 'appreciation': 4715, 'recounting': 16177, 'designation': 42878, 'notorious': 2824, "b'lanna": 55689, 'imotep': 55690, 'b
arre': 55691, 'jurgens': 41975, "joseph's": 18607, 'walkers': 17703, 'plaintiff': 55692, 'auer': 16888, 'legacy': 5124, 'play
time': 27923, 'abandons': 15563, 'cheeseball': 25505, 'vizier': 23568, 'dimentional': 41976, 'fikret': 55693, 'nastassja': 14
957, 'antique': 17704, 'couer': 55694, 'stains': 27924, 'unjustifiable': 55695, 'stood': 3397, 'dithered': 41977, 'galvanic':
 41978, 'putzi': 55696, 'distaste': 13468, 'apprehensions': 55697, 'apostrophe': 41979, 'kumari': 18608, 'hanpei': 41980, 'de
viate': 35236, 'jcc': 44730, 'churl': 55699, 'misjudging': 55700, 'watcxh': 55701, 'vocally': 30988, 'covets': 41982, 'stakeo
ut': 41983, 'tish': 41984, "married'": 41985, "'hair'": 55702, 'fitful': 55703, 'thesecurity': 55704, 'vivica': 41986, 'beetc
h': 41987, 'kickboxing': 38453, 'reserve': 13469, 'elkaïm': 35237, 'anaesthesia': 55706, 'compatibility': 55707, 'intertitl
e': 41988, 'pursue': 6631, 'sensing': 27925, 'emancipator': 41989, 'mullins': 67488, '64': 9901, 'appears': 734, "edison's":
 20725, 'latest': 2470, 'reachindia': 71277, 'oilwell': 55708, 'busty': 18848, "'macbeth'": 47846, 'misjudges': 55128, 'insen
sible': 41992, 'indicates': 10334, 'sugarbabe': 55711, 'sgcc': 55712, "mcnamara's": 55713, 'rohauer': 55714, 'deemed': 7586,
 'acrobatic': 19595, 'ajax': 23569, "about'": 55715, '1939': 5401, 'bennifer': 80025, 'bartel': 20371, 'playroom': 41993, 'ca
lhoun': 16178, "teacher'": 55716, 'azkaban': 35238, 'fuhgeddaboudit': 55717, 'preys': 35239, 'persuasive': 16179, 'portly': 2
2049, 'bajillion': 55718, 'exploded': 13046, 'nests': 77303, 'dolemite': 17546, 'conceded': 41995, 'batpeople': 71279, 'betra
il': 55719, 'petrified': 19596, 'taekwon': 55720, 'claiborne': 55721, 'howling': 7972, 'closes': 10800, 'raquel': 13470, 'mal
ignant': 29416, 'wormy': 41996, 'inconcievably': 68624, 'investigative': 14415, "'japs'": 47871, 'dismembering': 25506, 'hilb
rand': 38472, 'testosterone': 16180, 'lieber': 55723, 'burrow': 55724, 'bettie': 3228, 'thailand': 7704, 'pagegenre': 55725,
 "susie'": 35242, 'bright': 1018, 'enzos': 55726, '2nd': 2074, 'thirty': 3250, 'shun': 25507, 'missie': 30000, 'mocks': 1860
```

Convert word to index



```
In [22]: x_train_seq = token.texts_to_sequences(train_text)
         x_test_seq  = token.texts_to_sequences(test_text)
```

```
In [23]: print(train_text[0])
```

```
Bromwell High is a cartoon comedy. It ran at the same time as some other programs about school life, such as "Teachers". My 35
 years in the teaching profession lead me to believe that Bromwell High's satire is much closer to reality than is "Teachers".
 The scramble to survive financially, the insightful students who can see right through their pathetic teachers' pomp, the pett
iness of the whole situation, all remind me of the schools I knew and their students. When I saw the episode in which a student
repeatedly tried to burn down the school, I immediately recalled ......... at .......... High. A classic line: INSPECTOR: I'm h
ere to sack one of your teachers. STUDENT: Welcome to Bromwell High. I expect that many adults of my age think that Bromwell Hi
gh is far fetched. What a pity that it isn't!
```

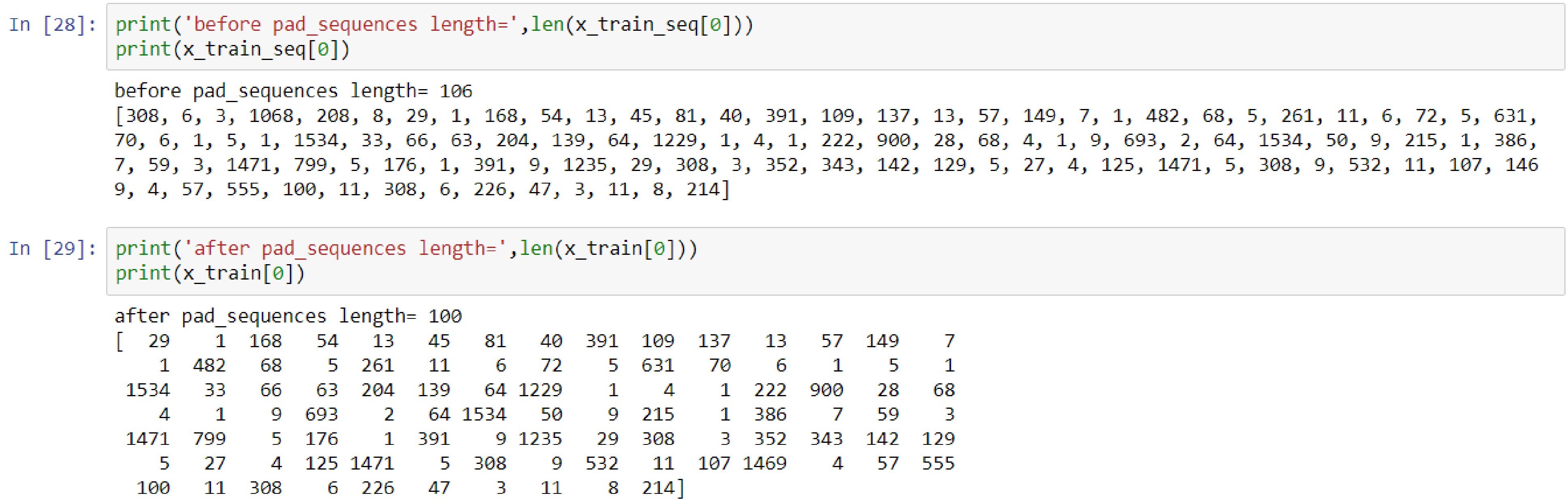
```
In [24]: print(x_train_seq[0])
```

```
[308, 6, 3, 1068, 208, 8, 29, 1, 168, 54, 13, 45, 81, 40, 391, 109, 137, 13, 57, 149, 7, 1, 482, 68, 5, 261, 11, 6, 72, 5, 631,
70, 6, 1, 5, 1, 1534, 33, 66, 63, 204, 139, 64, 1229, 1, 4, 1, 222, 900, 28, 68, 4, 1, 9, 693, 2, 64, 1534, 50, 9, 215, 1, 386,
7, 59, 3, 1471, 799, 5, 176, 1, 391, 9, 1235, 29, 308, 3, 352, 343, 142, 129, 5, 27, 4, 125, 1471, 5, 308, 9, 532, 11, 107, 146
9, 4, 57, 555, 100, 11, 308, 6, 226, 47, 3, 11, 8, 214]
```

```
In [26]: x_train = sequence.pad_sequences(x_train_seq, maxlen=100)
         x_test  = sequence.pad_sequences(x_test_seq,  maxlen=100)
```

Use dictionary indexes to replace words. If a word not in the list, ignore it.

Word padding



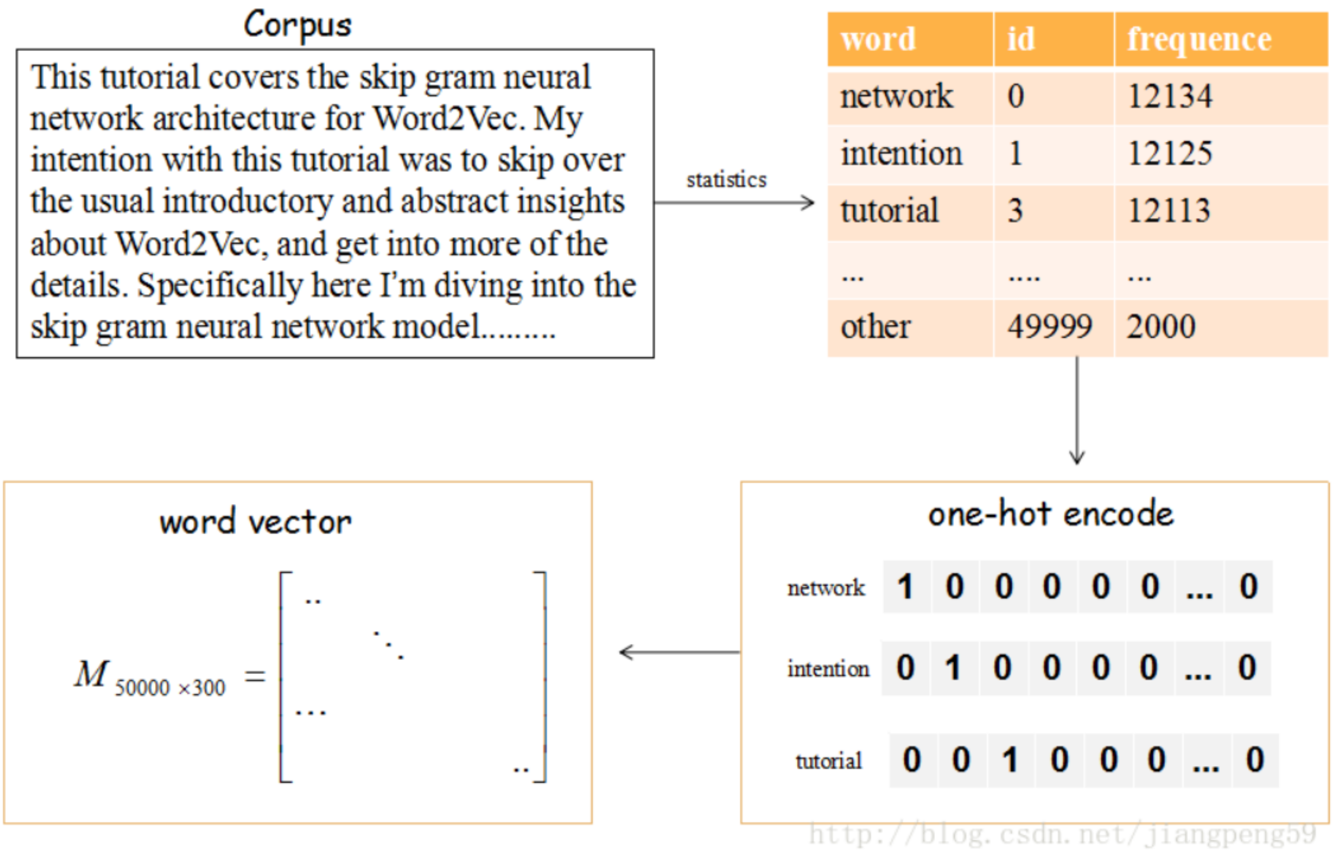
```
In [28]: print('before pad_sequences length=',len(x_train_seq[0]))
         print(x_train_seq[0])

before pad_sequences length= 106
[308, 6, 3, 1068, 208, 8, 29, 1, 168, 54, 13, 45, 81, 40, 391, 109, 137, 13, 57, 149, 7, 1, 482, 68, 5, 261, 11, 6, 72, 5, 631,
70, 6, 1, 5, 1, 1534, 33, 66, 63, 204, 139, 64, 1229, 1, 4, 1, 222, 900, 28, 68, 4, 1, 9, 693, 2, 64, 1534, 50, 9, 215, 1, 386,
7, 59, 3, 1471, 799, 5, 176, 1, 391, 9, 1235, 29, 308, 3, 352, 343, 142, 129, 5, 27, 4, 125, 1471, 5, 308, 9, 532, 11, 107, 146
9, 4, 57, 555, 100, 11, 308, 6, 226, 47, 3, 11, 8, 214]
```

```
In [29]: print('after pad_sequences length=',len(x_train[0]))
         print(x_train[0])

after pad_sequences length= 100
[  29    1  168   54   13   45   81   40  391  109  137   13   57  149    7
    1  482   68    5  261   11    6   72    5  631   70    6    1    5    1
 1534   33   66   63  204  139   64 1229    1    4    1  222  900   28   68
    4    1    9  693    2   64 1534   50    9  215    1  386    7   59    3
 1471  799    5  176    1  391    9 1235   29  308    3  352  343  142  129
    5   27    4  125 1471    5  308    9  532   11  107 1469    4   57  555
  100   11  308    6  226   47    3   11    8  214]
```

Ok, now all data become vectors with the same size !
Too long => truncate the front part
Too short => put 0 in the front part

Word Embedding

What is word embeding?



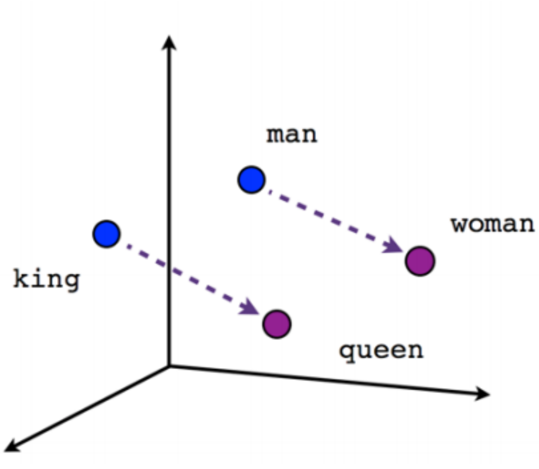
This tutorial covers the skip gram neural network architecture for Word2Vec. My intention with this tutorial was to skip over the usual introductory and abstract insights about Word2Vec, and get into more of the details. Specifically here I'm diving into the skip gram neural network model.........

word	id	frequence
network	0	12134
intention	1	12125
tutorial	3	12113
...	....	...
other	49999	2000

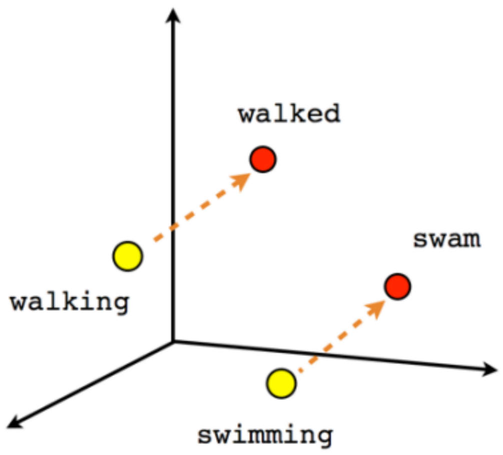
network	1	0	0	0	0	0	...	0
intention	0	1	0	0	0	0	...	0
tutorial	0	0	1	0	0	0	...	0

One-hot encoding is bad:
1). sparse 2). high dimension
3). words are not relevant 4).word vectors are “orthogonal”

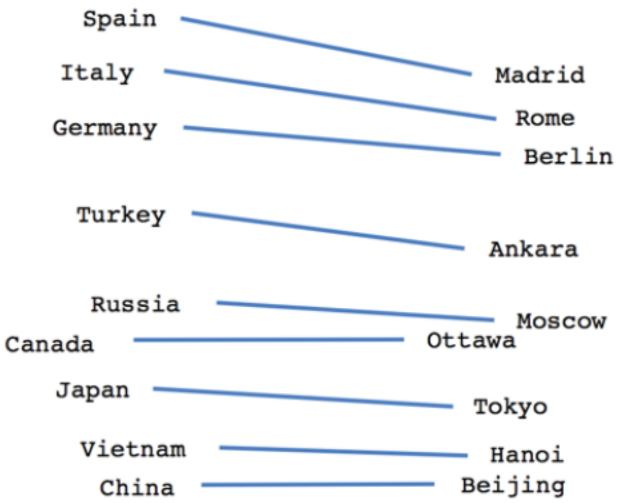
What can word embedding tell us?



Male-Female



Verb tense



Country-Capital

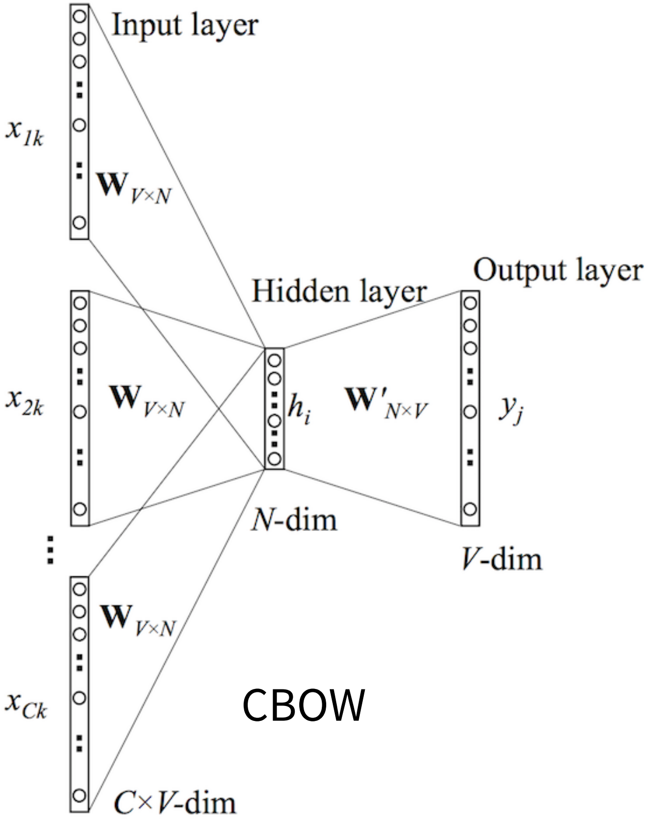
(King) – (man) = (queen)
(queen) – (woman) = (king)

(walked) – (walking) =
(swam)-(swimming)

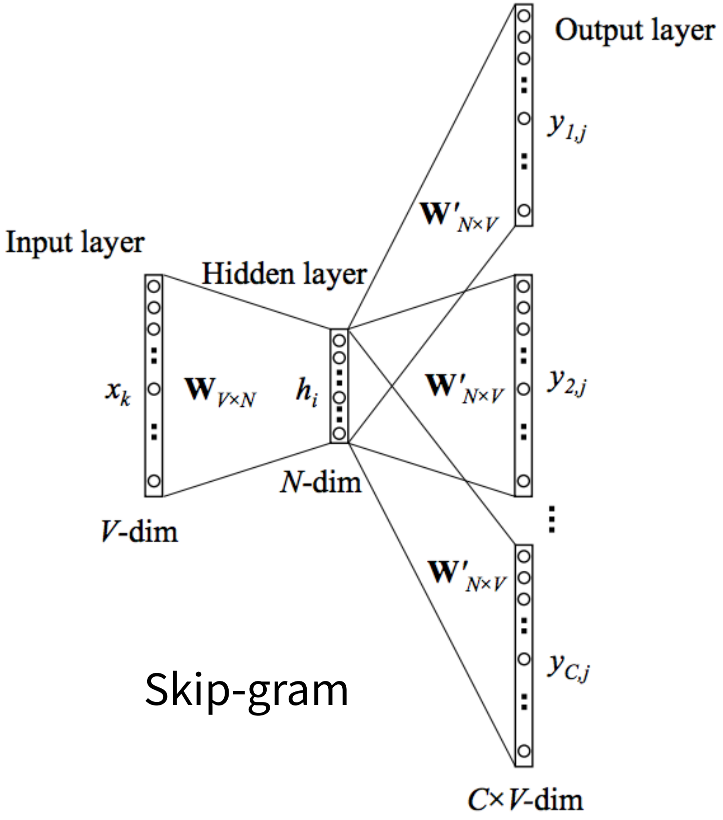
(Rome) + (Madrid) =
(Italy)+(Rome)

At least: school, teacher, students are close

Word2Vec



CBOW

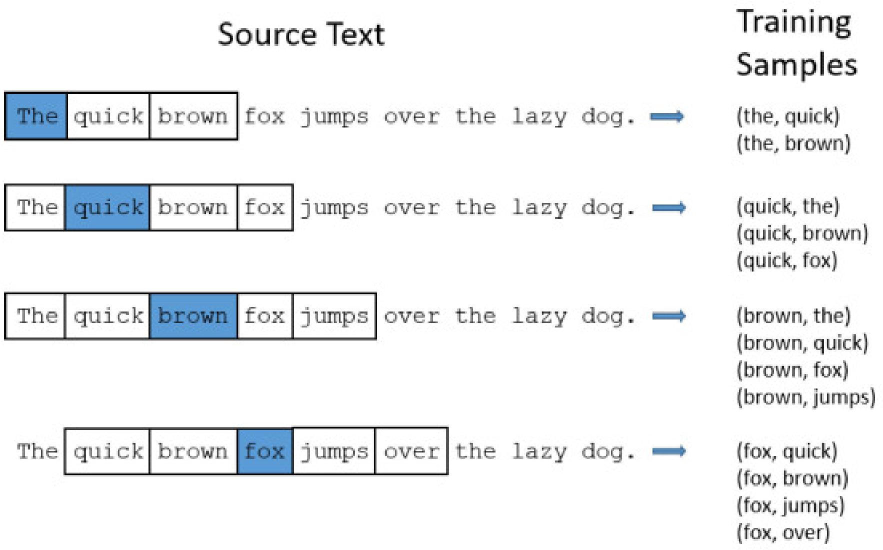


Skip-gram

CBOW is faster but Skip-gram is better for infrequent words

Glove is another popular word embedding algorithm based on co-occurrence (much faster!

Skip-gram



Source Text

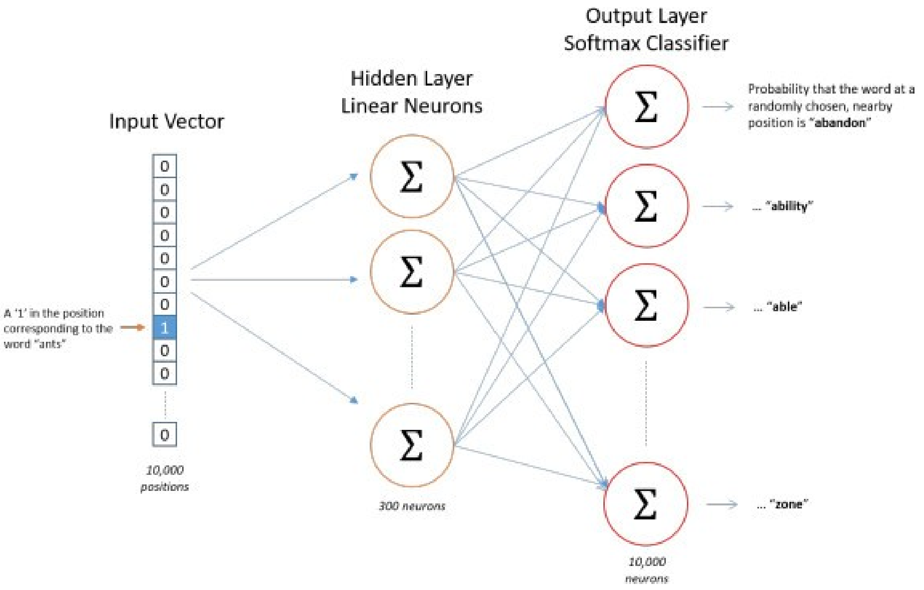
Training Samples

The quick brown fox jumps over the lazy dog. → (the, quick)
(the, brown)

The quick brown fox jumps over the lazy dog. → (quick, the)
(quick, brown)
(quick, fox)

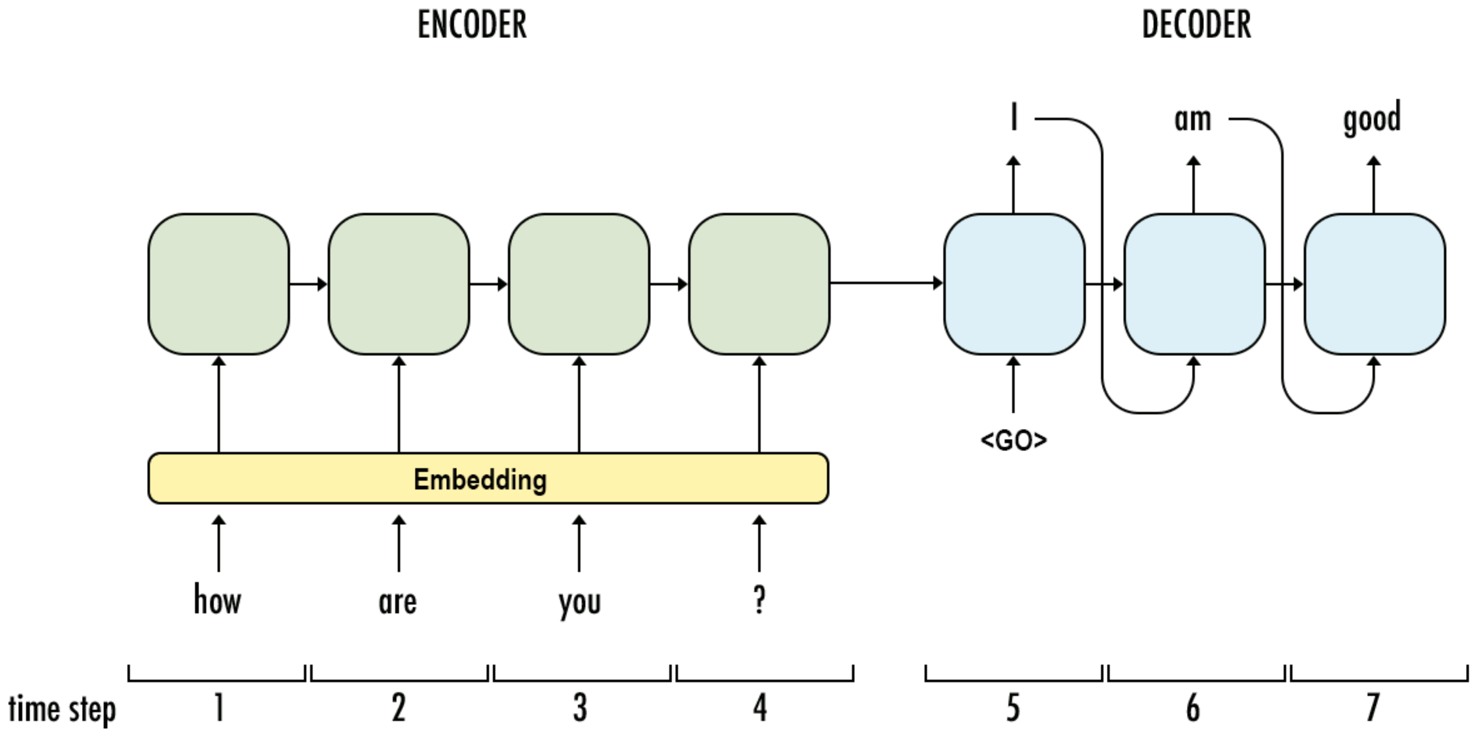
The quick brown fox jumps over the lazy dog. → (brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)

The quick brown fox jumps over the lazy dog. → (fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)



Popular Models

sequence-to-sequence model



$$X = \langle x_1, x_2 \dots x_m \rangle$$

$$Y = \langle y_1, y_2 \dots y_n \rangle$$

$$C = \mathcal{F}(x_1, x_2 \dots x_m)$$

$$y_i = \mathcal{G}(C, y_1, y_2 \dots y_{i-1})$$

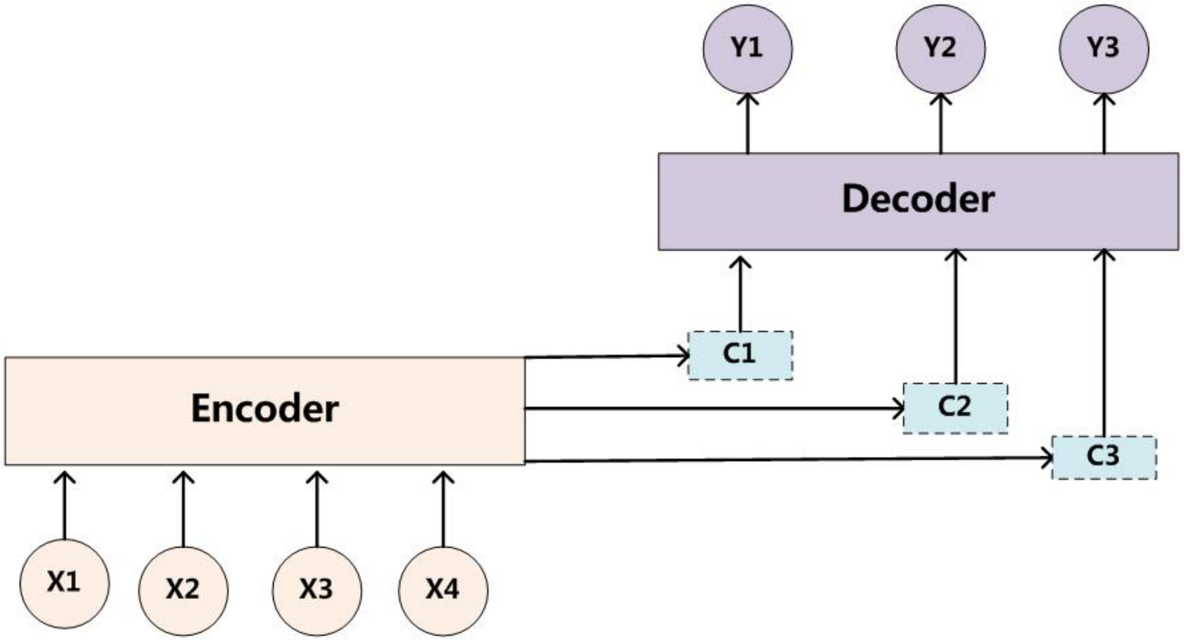
Attention-Based Model

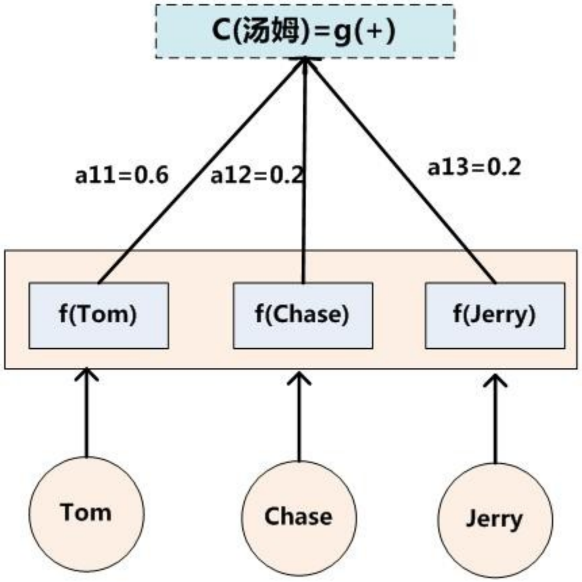
Focus on Mickey

Can you tell me some details of Goofy

https://arxiv.org/abs/1508.04025

Attention-based model





$$y_1 = f1(C_1)$$
$$y_2 = f1(C_2, y_1)$$
$$y_3 = f1(C_3, y_1, y_2)$$

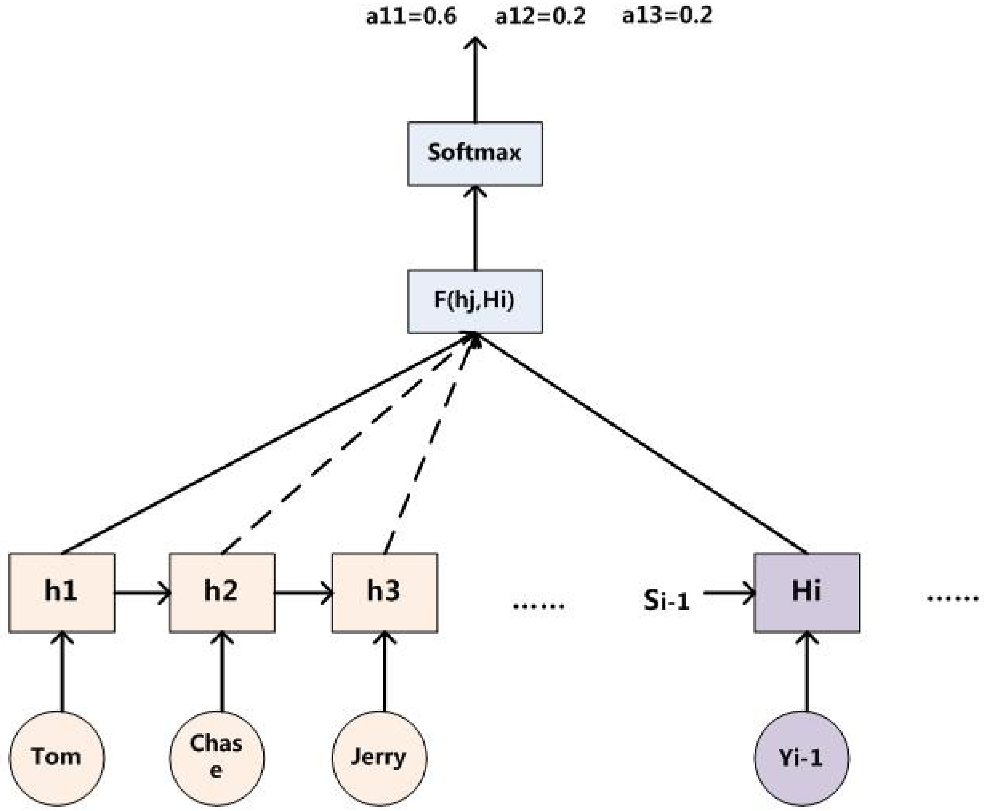
$$C_{汤姆} = g(0.6 * f2(\text{"Tom"}), 0.2 * f2(\text{Chase}), 0.2 * f2(\text{"Jerry"}))$$

$$C_{追逐} = g(0.2 * f2(\text{"Tom"}), 0.7 * f2(\text{Chase}), 0.1 * f2(\text{"Jerry"}))$$

$$C_{杰瑞} = g(0.3 * f2(\text{"Tom"}), 0.2 * f2(\text{Chase}), 0.5 * f2(\text{"Jerry"}))$$

$$c_i = \overset{g}{\boxed{\sum_{j=1}^{T_x}}} \alpha_{ij} h_j$$

How to get attention weight?



$$\begin{aligned} a_t(s) &= \operatorname{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \\ &= \frac{\exp\left(\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \end{aligned}$$

$$\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s] & \textit{concat} \end{cases}$$

Wa are parameter matrix to be learned