Tutorial 1

Learning multi-grained aspect target sequence for Chinese sentiment analysis

H Peng, Y Ma, Y Li, E Cambria

Knowledge-Based Systems (2018)

Ideas

- Task: Aspect term sentiment classification
- Problems
  - Eg.: The red apple released in California was not that interesting.
  - Eg.: The room size is small, but the view is excellent.
- Opportunities in Chinese
  - Compositionality

Train (火车)
Wood (木)

=

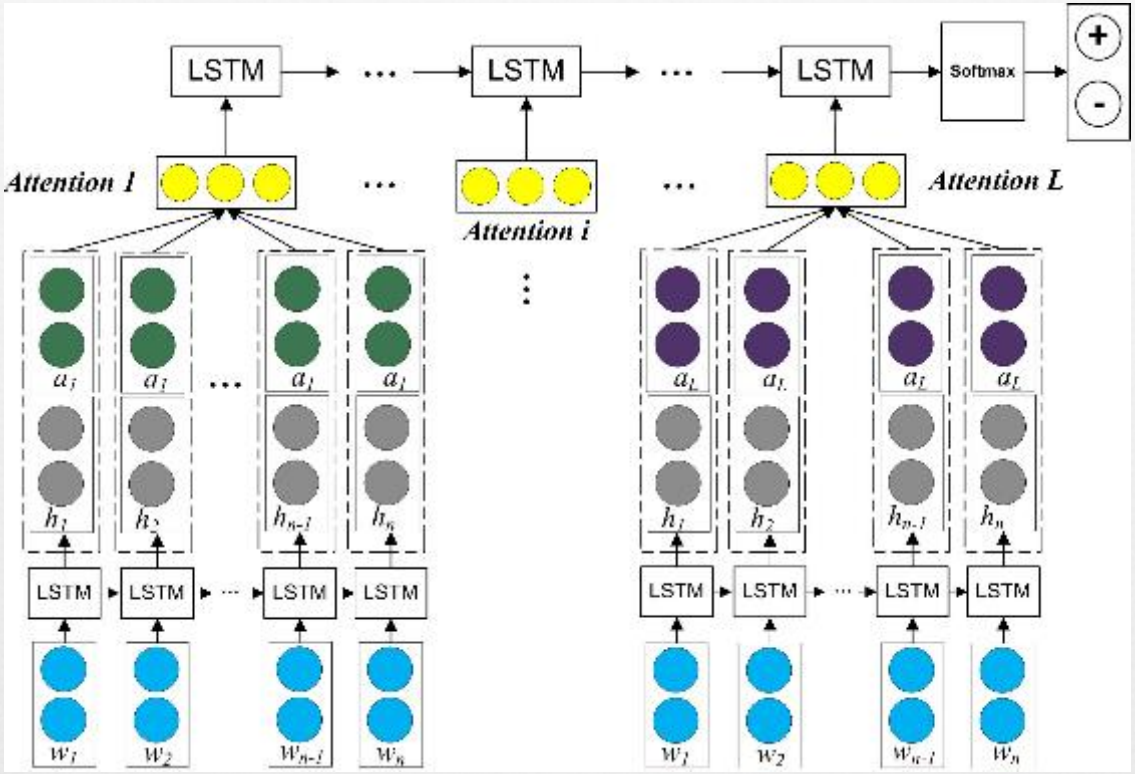
Fire (火)
Jungle (林)

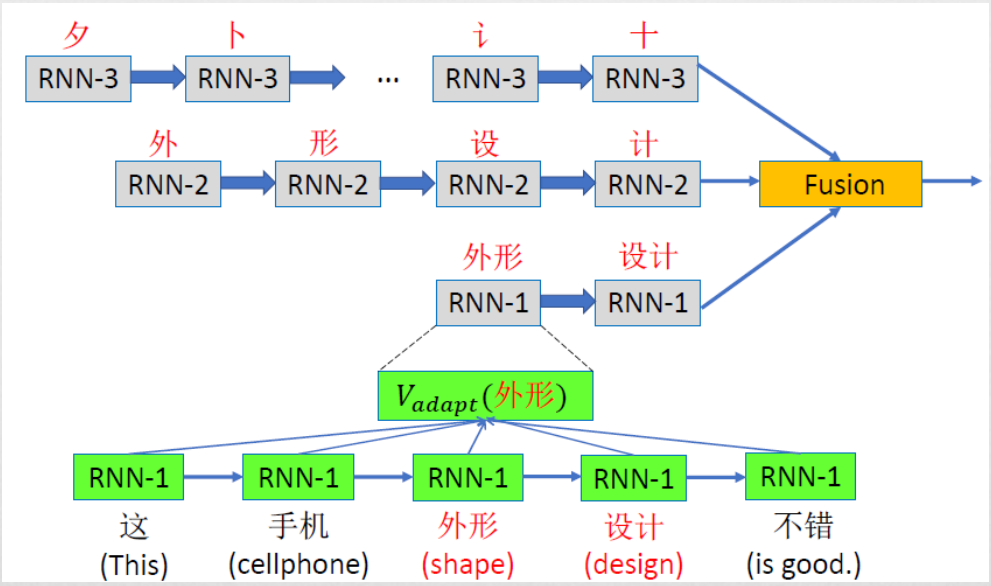
+

Vehicle (车)
Forest (森)

Solutions

- Adaptive word embeddings
  - Attention mechanism
- Aspect target sequence modelling
  - Sequence modelling-LSTM
- Multi-grained learning
  - Fusion of granularities


LSTM
LSTM
LSTM
Softmax
Attention 1
Attention i
Attention L
LSTM
LSTM
LSTM
LSTM


夕
卜
讠
十
RNN-3
RNN-3
RNN-3
RNN-3
外
形
设
计
RNN-2
RNN-2
RNN-2
RNN-2
Fusion
外形
设计
RNN-1
RNN-1
V_{adapt}(外形)
RNN-1
RNN-1
RNN-1
RNN-1
RNN-1
这
(This)
手机
(cellphone)
外形
(shape)
设计
(design)
不错
(is good.)

```
#Sentence embedding vectors
inputs = tf.placeholder(tf.float32,shape=(None, 35,300))
#Sentence sequential info learning
lstm1 = LSTM(output_dim=75, return_sequences=True,W_regularizer=l2(0.01))(inputs)
#Aspect mask
aspect_id = Input(dtype=np.float32,batch_shape=[inputs_shape[0],inputs_shape[1],3])
#Aspect words embedding
x_batch_dot = K.batch_dot(inputs,aspect_id, axes=[1, 1])
#Adaptive embedding learning
def aspect_layer(seq):
    vec=x_batch_dot[:,:,seq]
    vec2= K.repeat(vec, 35)
    atten_inputs=tf.concat(2,[lstm1,vec2])
    atten_inputs=tf.cast(atten_inputs,tf.float32)
    gi= Dense(1, bias=True, activation='tanh')(atten_inputs)
    ai= Dense(1,bias=False,activation='softmax')(gi)
    atten_output = K.batch_dot(atten_inputs,ai, axes=[1, 1])
    atten_output=K.squeeze(atten_output,2)
    return atten_output

# aspect from 1 to 3
atten_output1=aspect_layer(0)
atten_output2=aspect_layer(1)
atten_output3=aspect_layer(2)
# construct input sequence for target sequence learning
lstm_input= tf.concat(1,[tf.expand_dims(atten_output1, 1),tf.expand_dims(atten_output2, 1)])
lstm_input1=tf.concat(1,[lstm_input,tf.expand_dims(atten_output3, 1)])
#Aspect target sequence learning
var0= LSTM(output_dim=100,  activation='tanh',return_sequences=False,W_regularizer=l2(0.01), inner_activation='sigmoid')(lstm_input1)
#Dropout
var1= Dropout(0.5)(var0)
#Softmax layer
predictions= Dense(3,bias=False,activation='softmax')(var1)
#Sentiment labels
labels = tf.placeholder(tf.float32, shape=(None,3))
#Error
loss = tf.reduce_mean(categorical_crossentropy(labels, predictions))
#Optimizer
train_step = tf.train.AdagradOptimizer(learning_rate=0.01).minimize(loss)
#Evaluation metric
acc_value = accuracy(labels, predictions)
```

https://github.com/SenticNet

Q1

Term	Docs1	Docs2	Docs3
Angels Fools	1	0	0
Angels rush	1	0	1
Angels fear	0	0	1
Fools rush	1	0	0
Fear fools	0	1	0
Fear to	0	1	1
Where angels	1	0	1
To tread	1	0	0
Fear in	0	1	1
Rush in	1	0	1

a) Which phrase query?
   1. fools rush in
   2. where angels rush in
   3. angels fear to tread
b) Which are , if any, the document retrieved?

Q2

term	doc1	doc2	doc3
angels	#36, 174, 252, 651$		#15, 123, 412$
fools	#1, 17, 74, 222$	#8, 78, 108, 458$	
fear		#13, 43, 113, 433$	#18, 328, 528$
in	#3, 37, 76, 444, 851$	#10, 20, 110, 470, 500$	#5, 17, 25, 195$
rush	#2, 66, 194, 321, 702$		#4, 16, 404$
to	#47, 86, 234, 999$	#14, 24, 774, 944$	#19, 319, 599, 709$
tread	#57, 94, 333$		
where	#67, 124, 393, 1001$	#11, 41, 101, 421, 431$;	#14, 36, 736$

- Wh…rase qu…
  (a) …
  (b) “where angels rush in”
  (c) “angels fear to tread”

Biword Index

- Index every consecutive pair of terms in the text as a phrase
- Es. *Friends, Romans, Countrymen* would generate the biwords:
  1. *friends romans*
  2. *romans countrymen*
- Longer phrase queries can be broken into the Boolean query on biwords:
- Es*. stanford university palo alto*
- *stanford university* AND *university palo* AND *palo alto*

Positional index

- Extract inverted index entries for each distinct term: ***to, be, or, not.***
- Merge their *doc:position* lists to enumerate all positions with "***to be or not to be***".
  - ***to****:*
    - *2*:1,17,74,222,551; *4*:8,16,190,429,433; *7*:13,23,191; ...
  - ***be****:*
    - *1*:17,19; *4*:17,191,291,430,434; *5*:14,19,101; ...
- Same general method for proximity searches

Group discussion

A1.a

- “fools rush in” => fools rush AND rush in
- “where angels rush in” => where angels AND angels rush AND rush in
- “angels fear to tread” => angels fear AND fear to AND to tread

A1.b

- “fools rush in” = doc1
- “where angels rush in” = doc1, doc3
- “angels fear to tread” = null

A2

- “fools rush in” => doc1

Fools #1, 17, 74, 222$ **rush** #2, 66, 194, 321, 702$ **in** #3, 37, 76, 444, 851$

- “where angels rush in” => doc3

Where #14, 36, 736$ **angels** #15, 123, 412$ **rush** #4, 16, 404$ **in** #5, 17, 25, 195$

~~Doc1~~;No positional merge available

Where #67, 124, 393, 1001$ **angels** #36, 174, 252, 651$ **rush** #2, 66, 194, 321, 702$ **in** #3, 37, 76, 444, 851$

Q3

- Consider the table of term frequencies for 3 documents denoted Doc1, Doc2, Doc3 below. Compute the tf-idf weights for the terms *car, auto, insurance, best*, for each document, using the idf values from the table below.

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

term	Doc 1	Doc 2	Doc3	idf
car	27	10	24	1.65
auto	3	33	0	2.08
insurance	0	33	29	1.62
best	14	0	17	1.5

Recall

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Best known weighting scheme in information retrieval
  - Note: the “-” in tf-idf is a hyphen, not a minus sign!
  - Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document

Group discussion

A3

$$w_{t,d} = (1 + \log_{10} \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

tf	Doc 1	Doc 2	Doc3	idf
car	27	10	24	1.65
auto	3	33	0	2.08
insurance	0	33	29	1.62
				1.5

1+log tf	Doc 1	Doc 2	Doc3
car	2.43	2	2.38
auto	1.48	2.52	0
insurance	0	2.52	2.46
best	2.15	0	2.23

w	Doc 1	Doc 2	Doc3
car	4.01	3.3	3.93
auto	3.08	5.24	0
insurance	0	4.08	3.99
best	3.23	0	3.35

Q4

- Refer to the tf and idf values for four terms and three documents from Q3. Compute the two top scoring documents on the query *best car insurance* for each of the following weighing schemes: (i) nnn.atc; (ii) ntc.atc.

ddd.qqq

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

tf-idf example: lnc.ltc

Recall

Document: *car insurance auto insurance*
Query: *best car insurance*

Term	Document				Query						Prod
	tf-raw	tf-wt	wt	normalize	tf-raw	tf-wt	df	idf	wt	normalize	
auto	1	1	1	0.52	0	0	5000	2.3	0	0	0
best	0	0	0	0	1	1	50000	1.3	1.3	0.34	0
car	1	1	1	0.52	1	1	10000	2.0	2.0	0.52	0.27
insurance	2	1.3	1.3	0.68	1	1	1000	3.0	3.0	0.78	0.53

Doc length $= \sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$

Score = 0+0+0.27+0.53 = 0.8

Group discussion

A4

- Find document vectors: (i) nnn (ii) ntc

nnn	Doc1	Doc2	Doc3
car	27*1*1=27	10*1*1=10	24*1*1=24
auto	3*1*1=3	33*1*1=33	0*1*1=0
insuran	0*1*1=0	33*1*1=	29*1*1=

ntc	Doc1	Doc2	Doc3
car	(27*1.65=44.55)/49.6= 0.90	(10*1.65=16.5)/88.55= 0.19	(24*1.65=39.6)/66.5= 0.60
auto	(3*2.08=6.24)/49.6= 0.13	(33*2.08=68.64)/88.55= 0.78	0*2.08=0
insurance	0*1.62=0	(33*1.62=53.46)/88.55= 0.60	(29*1.62=46.98)/66.5= 0.71
best	(14*1.5=21)/49.6= 0.42	0*1.5=0	(17*1.5=25.5)/66.5= 0.38

A4

- Find the vector for query *best car insurance*: (i,ii) atc

	tf	a	t	at	atc
car	1	0.5+0.5*1/1=1	1.65	1.65	0.60
auto	0	0	0	0	0
insurance	1	0.5+0.5*1/1=1	1.62	1.62	0.59
best	1	0.5+0.5*1/1=	1.5	1.5	0.54

max(tf)=1
length=2.76

nnn.atc	Doc1	Doc2	Doc3
car	27*0.6=16.2	10*0.6=6	24*0.6=14.4
auto	0	0	0
insurance	0*0.59=0	33*0.59=19.47	29*0.59=17.11
best	14*0.54=7.56	0	17*0.54=9.18
SUM	23.75 (3rd)	25.47 (2nd)	40.69 (1st)

A4

- (ii) ntc.atc

ntc	Doc1	Doc2	Doc3	atc
car	0.90	0.19	0.60	0.60
auto	0.13	0.79	0	0
insurance	0	0.61	0.71	0.59
best	0.42	0	0.38	0.54

ntc.atc	Doc1	Doc2	Doc3
car	0.90*0.6=0.54	0.19*0.6=0.11	0.60*0.6=0.36
auto	0	0	0
insurance	0	0.61*0.59=0.36	0.71*0.59=0.42
best	0.42*0.54=0.23	0	0.38*0.54=0.21
SUM	0.77 (2nd)	0.47(3rd)	0.99 (1st)

Q5

	Antony and Cleopatra	Julius Caesar	The Tempest
Antony	157	73	0
Brutus	28	157	0
Caesar	232	227	0
Calpurnia	0	10	0
Cleopatra	23	0	37
Mercy	0	10	15
Worser	2	0	1

a) Compute the cosine similarity and the Euclidian distance between the documents and the query: **"caesar mercy brutus"** based on the term-document count matrix above.

b) How does the Euclidian distance change if we normalize the vectors?

NB: Compute the vector space using tf-idf formula of Q3

$$w_{t,d} = (1 + \log_{10} tf_{t,d}) \times \log_{10}(N / df_t)$$

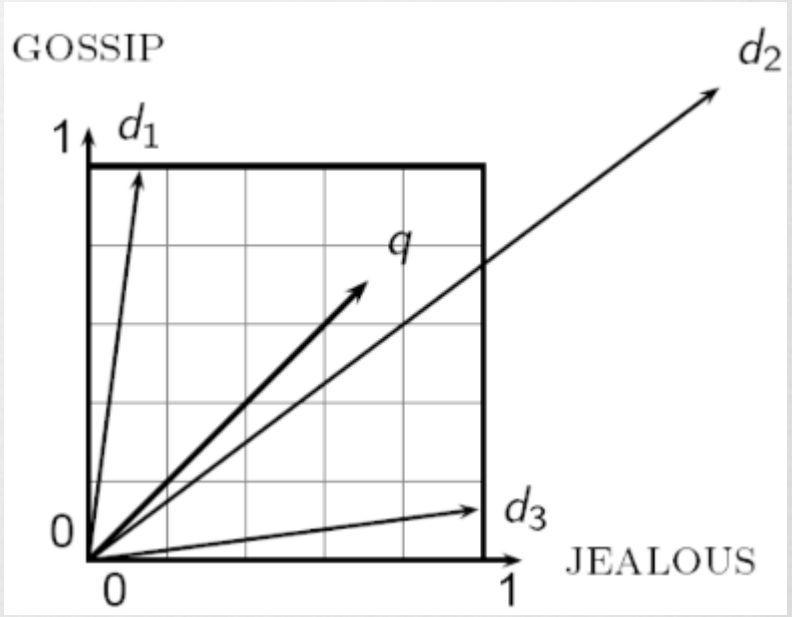
Euclidean distance

Recall

- Euclidean distance: the distance between points (x_1,y_1) and (x_2,y_2) is given by:

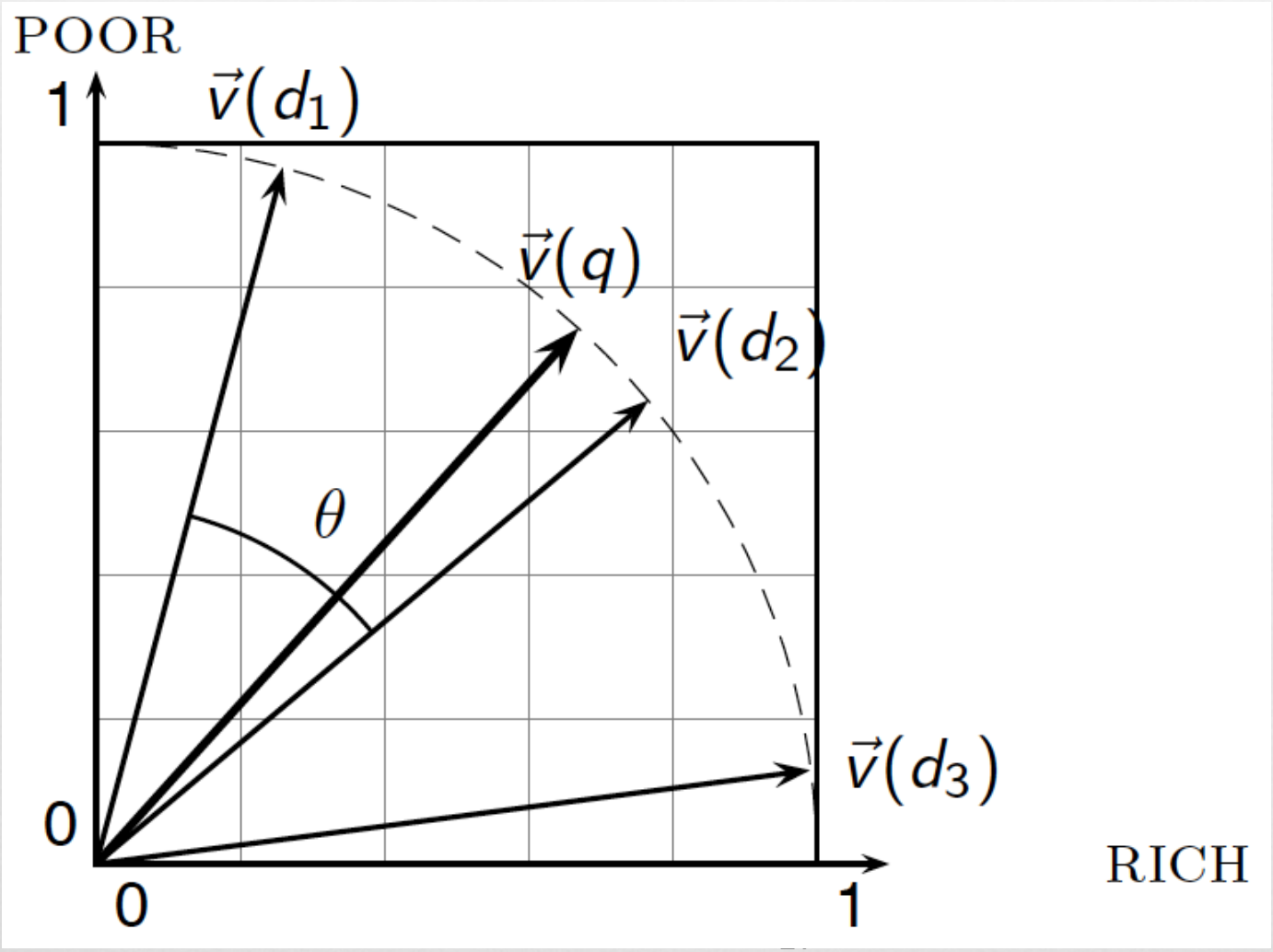
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- Unfortunately, this distance is biased by the length of the vectors. So is not able to detect the correct terms distribution



Cosine similarity illustrated

Recall



Group discussion

A5

- Compute the vector space

	Antony and Cleopatra	Julius Caesar	The Tempest	Query
Antony	0.56	0.50	0	0
Brutus	0.43	0.56	0	0.18
Caesar	0.59	0.59	0	0.18
Calpurnia	0	0.95	0	0
Cleopatra	0.42	0	0.45	0
Mercy	0	0.35	0.38	0.18
Worser	0.23	0	0.18	0

A5

$$ED = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$cos\_sim(q, d) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \sqrt{\sum_{i=1}^{n} d_i^2}}$$

	Antony and Cleopatra	Julius Caesar	The Tempest
Cosine similarity	0.57	0.62	0.35
Euclidian distance	0.90	1.22	0.58

A5

Normalized values				
	Antony and Cleopatra	Julius Caesar	The Tempest	Query
Antony	0.251	0.169	0	0
Brutus	0.193	0.19	0	0.33
Caesar	0.265	0.2	0	0.33
Calpurnia	0	0.322	0	0
Cleopatra	0.188	0	0.446	0
Mercy	0	0.119	0.376	0.33
Worser	0.103	0	0.178	0
Euclidian distance normaliz	0.49	0.47	0.67	

Tutorial 2

Context-Dependent Sentiment Analysis in User-Generated Videos

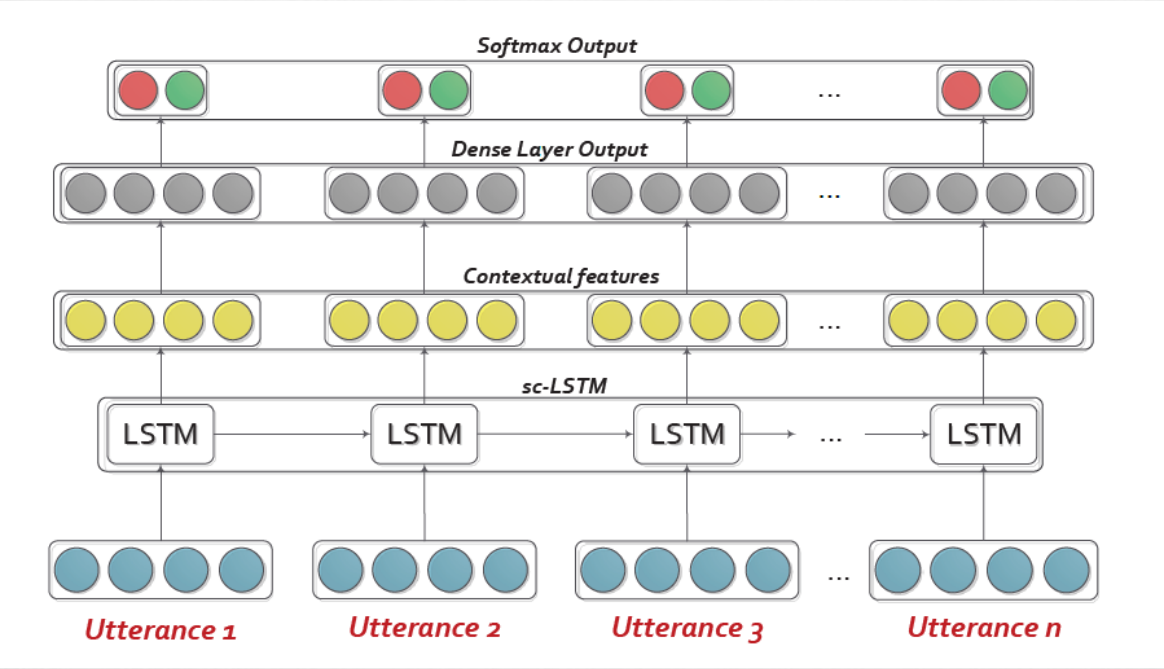
Poria, S., Cambria, E., Hazarika, D., Majumder, N., Zadeh, A., & Morency, L. P. (2017). In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 873-883).

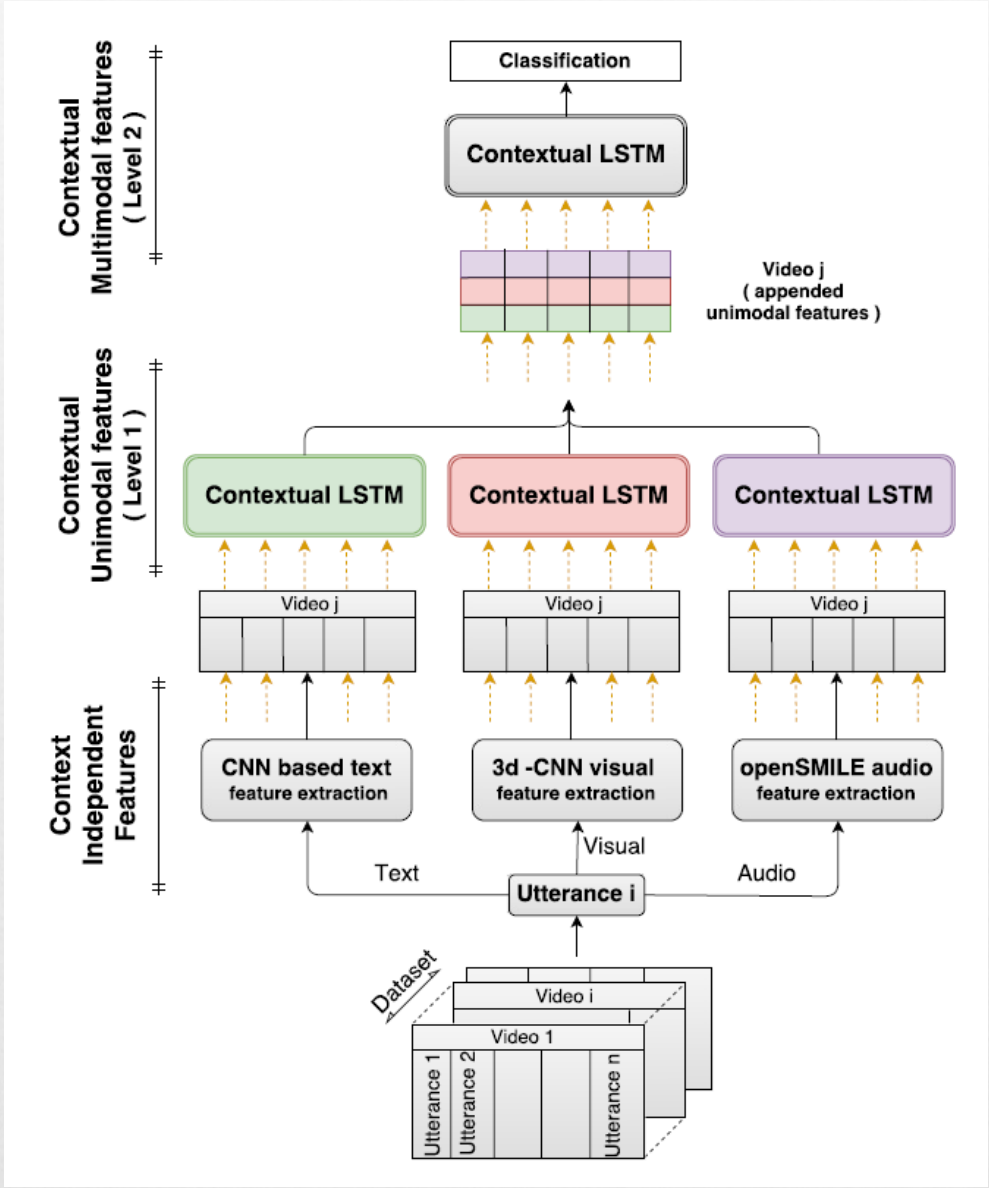
Idea

- Utterance context influences sentiment
  - eg.: Movie review of ‘Green Hornet’:
    - “The Green Hornet did something similar”
    - “It engages the audience more”,
    - “they took a new spin on it”,
    - “and I just loved it”

Solutions

- Model the order of utterance appearance
  - Contextual LSTM
- Fusion of modalities
  - Hierarchical Framework


Softmax Output
Dense Layer Output
Contextual features
sc-LSTM
LSTM
LSTM
LSTM
LSTM
Utterance 1
Utterance 2
Utterance 3
Utterance n


Contextual Multimodal features (Level 2)
Classification
Contextual LSTM
Video j (appended unimodal features)
Contextual Unimodal features (Level 1)
Contextual LSTM
Contextual LSTM
Contextual LSTM
Video j
Video j
Video j
Context Independent Features
CNN based text feature extraction
3d -CNN visual feature extraction
openSMILE audio feature extraction
Visual
Text
Audio
Utterance i
Dataset
Video i
Video 1
Utterance 1
Utterance 2
Utterance n

```
def multimodal(unimodal_activations):

    print "starting multimodal"
    #Fusion (appending) of features

    train_data = np.concatenate((unimodal_activations['text_train'], unimodal_activations['audio_train'], unimodal_activations['video_train']), axis=2)
    test_data = np.concatenate((unimodal_activations['text_test'], unimodal_activations['audio_test'], unimodal_activations['video_test']), axis=2)
    train_mask=unimodal_activations['train_mask']
    test_mask=unimodal_activations['test_mask']
    train_label=unimodal_activations['train_label']
    test_label=unimodal_activations['test_label']

    #Multimodal model

    input_data = Input(shape=(train_data.shape[1],train_data.shape[2]))
    masked = Masking(mask_value =0)(input_data)
    lstm = Bidirectional(LSTM(300, activation='tanh', return_sequences = True, dropout=0.4))(masked)
    inter = Dropout(0.9)(lstm)
    inter1 = TimeDistributed(Dense(500,activation='relu'))(inter)
    inter = Dropout(0.9)(inter1)
    output = TimeDistributed(Dense(2,activation='softmax'))(inter)

    model = Model(input_data, output)
    aux = Model(input_data, inter1)
    model.compile(optimizer='adadelta', loss='categorical_crossentropy', sample_weight_mode='temporal')
    early_stopping = EarlyStopping(monitor='val_loss', patience=10)
    model.fit(train_data, train_label,
                    epochs=200,
                    batch_size=10,
                    sample_weight=train_mask,
                    shuffle=True,
                    callbacks=[early_stopping],
                    validation_split=0.2)
```

https://github.com/SenticNet

Q1

- Consider the following 10 class conditioned word probabilities (c_0=non-spam, c_1=spam):

Word w_i	brand	huge	hottest	incredible	million	new	offers	pay	save	family
$p(w_i\|c_0)$	0.10	0.20	0.30	0.05	0.05	0.10	0.20	0.02	0.03	0.40
$p(w_i\|c_1)$	0.98	0.92	0.91	0.99	0.98	0.99	0.93	0.99	0.99	0.02

- For each of the 3 email snippets below, ignoring case, punctuations, and words beyond the 10 known vocabulary words, compute the class conditioned document probabilities for each of the 3 documents (6 in total: $P(d_1|c_0)$, $P(d_2|c_0)$, $P(d_3|c_0)$, $P(d_1|c_1)$, $P(d_2|c_1)$, $P(d_3|c_1)$) using the Naïve Bayes model.

Recall

Naive Bayes Classifier

$$d = \langle x_1, x_2, \ldots, x_n \rangle$$

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}}\, P(c_j \mid x_1, x_2, \ldots, x_n)$$

The probability of a document *d* being in class *c*.

$$= \underset{c_j \in C}{\operatorname{argmax}}\, P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

Bayes' Rule

$$= \underset{c_j \in C}{\operatorname{argmax}}\, P(x_1 \mid c_j) P(x_2 \mid c_j) \ldots P(x_n \mid c_j) P(c_j)$$

Conditional Dependence Assumption

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N} \qquad \hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

Q1: documents

Word w_i	brand	huge	hottest	incredible	million	new	offers	pay	save	family
$p(w_i\|c_0)$	0.10	0.20	0.30	0.05	0.05	0.10	0.20	0.02	0.03	0.40
$p(w_i\|c_1)$	0.98	0.92	0.91	0.99	0.98	0.99	0.93	0.99	0.99	0.02

- d1: OEM software - throw packing case, leave CD, use electronic manuals. **Pay** for software only and **save** 75-90%! Find **incredible** discounts! See our special **offers**!
- d2: Our **Hottest** pick this year! **Brand new** issue Cana Petroleum! VERY tightly held, in a booming business sector, with a **huge** publicity campaign starting up, Cana Petroleum (CNPM) is set to bring all our readers **huge** gains. We advise you to get in on this one and ride it to the top!
- d3: Dear friend, How is your **family**? hope all of you are fine, if so splendid. Yaw Osafo-Maafo is my name and former Ghanaian minister of finance. Although I was sacked by President John Kufuor on 28 April 2006 for the fact I signed 29 **million** book publication contract with Macmillan Education without reference to the Public Procurement Board and without Parliamentary approval.

Q1: Naïve Bayes model

$$p(d_j \mid c_k) \propto \prod_{i=1}^{t} p(w_i \mid c_k)^{f(w_i, d_j)}$$

- where $f(w_i,d_j)$ = frequency of word w_i in document d_j

Word w_i	brand	huge	hottest	incredible	million	new	offers	pay	save	family
$p(w_i \mid c_0)$	0.10	0.20	0.30	0.05	0.05	0.10	0.20	0.02	0.03	0.40
$p(w_i \mid c_1)$	0.98	0.92	0.91	0.99	0.98	0.99	0.93	0.99	0.99	0.02

Hint

d2: Our **Hottest** pick this year! **Brand new** issue Cana Petroleum! VERY tightly held, in a booming business sector, with a **huge** publicity campaign starting up, Cana Petroleum (CNPM) is set to bring all our readers **huge** gains. We advise you to get in on this one and ride it to the top!

$$p(d_j \mid c_k) \propto \prod_{i=1}^{t} p(w_i \mid c_k)^{f(w_i, d_j)}$$

$p(d_2|c_0) = p(\text{hottest}|c_0)*p(\text{brand}|c_0)*p(\text{new}|c_0)*p(\text{huge}|c_0)^2$

Word w_i	brand	huge	hottest	incredible	million	new	offers	pay	save	family
$p(w_i\|c_0)$	0.10	0.20	0.30	0.05	0.05	0.10	0.20	0.02	0.03	0.40
$p(w_i\|c_1)$	0.98	0.92	0.91	0.99	0.98	0.99	0.93	0.99	0.99	0.02

Group discussion

A1

$$p(d_1 \mid c_0) \propto 0.05^1 \times 0.20^1 \times 0.02^1 \times 0.03^1 = 6 \times 10^{-6}$$

$$p(d_1 \mid c_1) \propto 0.99 \times 0.93 \times 0.99 \times 0.99 \approx 9.02 \times 10^{-1}$$

$$p(d_2 \mid c_0) \propto 0.10 \times 0.20^2 \times 0.30 \times 0.10 = 1.2 \times 10^{-4}$$

$$p(d_2 \mid c_1) \propto 0.98 \times 0.92^2 \times 0.91 \times 0.99 \approx 7.47 \times 10^{-1}$$

$$p(d_3 \mid c_0) \propto 0.05 \times 0.40 = 2 \times 10^{-2}$$

$$p(d_3 \mid c_1) \propto 0.98 \times 0.02 = 1.96 \times 10^{-2}$$

Q2

- Compute the posterior probabilities of each document in Question 1, given c_0 and c_1, (6 in total: $P(c_0|d_1)$, $P(c_1|d_1)$, $P(c_0|d_2)$, $P(c_1|d_2)$, $P(c_0|d_3)$, $P(c_1|d_3)$) assuming that 80% of all email received are spam, i.e., prior class probability $P(c_1)=0.8$ (from which you can derive $P(c_0)=1-P(c_1)$), and finally decide whether each document is spam.

$$p(c_k \mid d_j) \propto p(d_j \mid c_k)\, p(c_k)$$

Word w_i	brand	huge	hottest	incredible	million	new	offers	pay	save	family
$p(w_i\|c_0)$	0.10	0.20	0.30	0.05	0.05	0.10	0.20	0.02	0.03	0.40
$p(w_i\|c_1)$	0.98	0.92	0.91	0.99	0.98	0.99	0.93	0.99	0.99	0.02

Group discussion

Q2

- $P(c_0) = 1-P(c_1) = 0.2$

A2

- $P(c_0|d_1) \approx P(d_1|c_0)xP(c_0)=6x10^{-6}x0.2=1.2x10^{-6}$
- **$P(c_1|d_1) \approx P(d_1|c_1)xP(c_1)=0.902x0.8=0.72$**
- $P(c_0|d_2) \approx P(d_2|c_0)xP(c_0)=1.2x10^{-4}x0.2=2.4x10^{-5}$
- **$P(c_1|d_2) \approx P(d_2|c_1)xP(c_1)=0.747x0.8=0.6$**
- $P(c_0|d_3) \approx P(d_3|c_0)xP(c_0)=0.02x0.2=0.004$
- **$P(c_1|d_3) \approx P(d_3|c_1)xP(c_1)=0.0196x0.8=0.016$**

Q3

- Build a Naïve Bayes classifier using words as features for the training set in Table 2 and use the classifier to classify the test set in the table.

	docID	words in document	in $c = China$?
training set	1	Taipei Taiwan	yes
	2	Macao Taiwan Shanghai	yes
	3	Japan Sapporo	no
	4	Sapporo Osaka Taiwan	no
test set	5	Taiwan Taiwan Sapporo	?

Recall

Bayes probability

- Prior probability: $P(c_k)$
  - Probability of expecting class c_k before taking in account any evidence
- Likelihood: $P(doc_j|c_k) = \Pi_{i=1}^{n} P(x_i|c_k)$
  - True only because we make the **"naive" conditional independence assumptions**
- Posterior probability: $P(c_k|doc_j) \propto P(c_k)P(doc_j|c_k)$

Recall

Naive Bayes: Learning

$$P(c_K) = \frac{N(c_k)}{N}$$

Number of documents belonging to class c_k

Total number of documents

$$P(x_i|c_K) = \frac{N(x_i, c_k) + 1}{N(X, c_k) + |Vocabulary|}$$

Number of occurrence of term x_i in docs of class c_k

Number of terms appearing in docs of class c_k

Recall

MAP classifier

- MAP is maximum a posteriori
  - **Detect the class that maximize our posteriori probability**

$$\underset{c_k}{\operatorname{argmax}} P(c_k|doc_j) \propto \underset{c_k}{\operatorname{argmax}} P(c_k)\Pi_{i=1}^{n} P(x_i|c_k)$$

- We just try all the class c_k

Group discussion

A3

- Prior probability:
  - p(China)=2/4, p(~China)=2/4

	docID	words in document	in $c = China$?
training set	1	Taipei Taiwan	yes
	2	Macao Taiwan Shanghai	yes
	3	Japan Sapporo	no
	4	Sapporo Osaka Taiwan	no
test set	5	Taiwan Taiwan Sapporo	?

A3 (learning)

Doc Id	Terms		
1	Taipei	Taiwan	
2	Macao	Taiwan	Shanghai
3	Japan	Sapporo	
4	Sapporo	Osaka	Taiwan

Vocabulary = {Taipei, Taiwan, Macao, Shanghai, Japan, Sapporo, Osaka}
|Vocabulary| = 7

Doc class	#Terms
Yes	5
No	5

A3 (learning)

P(Taipei|yes)=(1+1)/(5+7)=2/12

P(Taipei|no)=(0+1)/(5+7)=1/12

P(Taiwan|yes)=(2+1)/(5+7)=3/12

P(Taiwan|no)=(1+1)/(5+7)=2/12

P(Sapporo|yes)=(0+1)/(5+7)=1/12

P(Sapporo|no)=(2+1)/(5+7)=3/12

$$P(x_i|c_k) = \frac{N(x_i, c_k)+1}{N(X, c_k)+|Vocabilary|}$$

A3 (classifying)

Doc Id	Terms		
5	Taiwan	Taiwan	Sapporo

$$P(c_k|d_5) \propto P(c_k)\Pi_{i=1}^{n} P(x_i|c_k)$$

P(yes|d_5)= $\frac{2}{4}\times\left(\frac{3}{12}\right)^2\times\frac{1}{12}\approx 2.60\times10^{-3}$

P(no|d_5)= $\frac{2}{4}\times\left(\frac{2}{12}\right)^2\times\frac{3}{12}\approx 3.47\times10^{-3}$

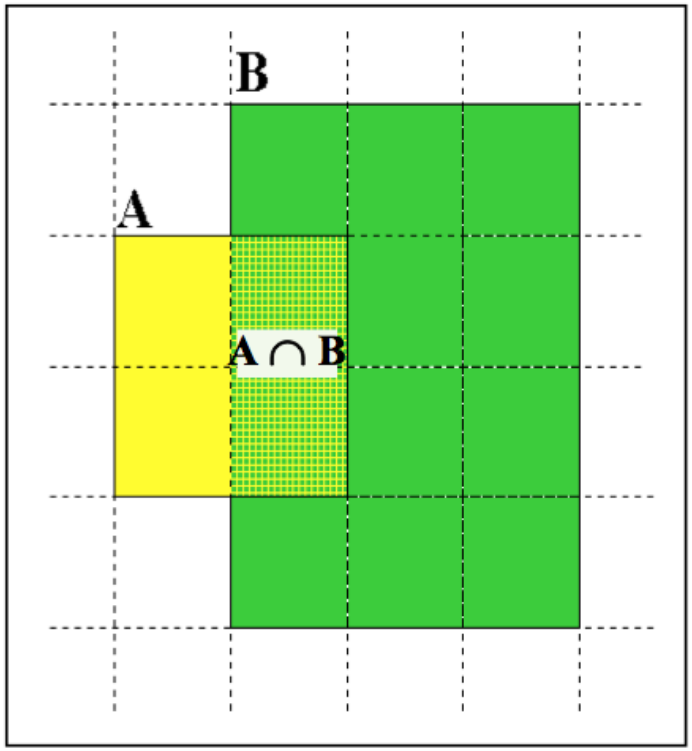
Answer: d_5 belongs to the class ‘no

Q4

- Each of two Web search engines A and B generates a large number of pages uniformly at random from their indexes. 30% of A’s pages are present in B’s index, while 50% of B’s pages are present in A’s index. What is the ratio between the number of pages in A’s index and the number of pages in B’s?

Relative Size from Overlap
Given two engines A and B

Recall



Sample URLs randomly from A

Check if contained in B and vice versa

$$A \cap B = (1/2) * \text{Size } A$$

$$A \cap B = (1/6) * \text{Size } B$$

$$(1/2)*\text{Size } A = (1/6)*\text{Size } B$$

$$\therefore \text{ Size } A / \text{ Size } B = (1/6)/(1/2) = 1/3$$

Each test involves: (i) Sampling (ii) Checking

Group discussion

A4

- 30% x A = 50% x B
- A/B = 5/3