Lecture 2   IE Basis

Fang Li
Dept. of Computer Science

Contents

- Basic Text Processing
- Language Model
- Vector Space Model
- Word Vector (option)

Basic Text Processing

- Aim: let computer to understand human language.

Basic Text Processing

- Steps:

1. Segmenting/tokenization

脑壳/疼/啊/，真/不/晓得/是/感冒/搞/的.

What're → what are   I'm → I am

Free software:

中科院分词 http://ictclas.org/index.html

Stanford 分词：http://nlp.stanford.edu/

Jieba 分词，海量分词，…

Basic Text Processing (cont.)

2. **Normalizing word form** (such as: English, German)

He studies English very hard.

Lemmatization（词干提取）: Studies→ study

Upper case and lower case: Fed. vs. fed

Morphemes（词最小语义单位）: cat vs. cats

Same lemma, different word forms.

e.g, uninterested =un(prefix)+interest(stem)+ed(suffix)

Porter's algorithm for English stemmer

- Change different word forms into its stem （词干The core meaning-bearing units）

Step 1a

```
sses → ss   caresses → caress
ies  → i    ponies   → poni
ss   → ss   caress   → caress
s    → ø    cats     → cat
```

Step 1b

```
(*v*)ing → ø
walking  → walk
sing     → sing
(*v*)ed  → ø
plastered → plaster
```

Step 2 (for long stems)

```
ational→ ate relational→ relate
izer→ ize    digitizer → digitize
ator→ ate    operator  → operate
…
```

Step 3 (for longer stems)

```
al   → ø   revival    → reviv
able → ø   adjustable → adjust
ate  → ø   activate   → activ
…
```

Basic Text Processing (cont.)

3. **Part of Speech Tagging** (noun, verb)

--With segmentation together

Example:

为加强对案件的督办和指导,省有关部门迅速成立工作组,赴阜新督办、指导案件调查工作,并将情况上报有关部门。

→

为/p 加强/v 对/p 案件/n 的/u 督办/v 和/c 指导/n ,/wp 省/n 有关/v 部门/n 迅速/a 成立/v 工作组/n ,/wd 赴/v 阜新/ns 督办/v 、/wp 指导/v 案件/n 调查/v 工作/v ,/wp 并/c 将/p 情况/n 上报/v 有关/v 部门/n 。/wp

Basic Text Processing (cont.)

4. **Sentence Parsing** (syntactic analysis)

Two views of linguistic structure:

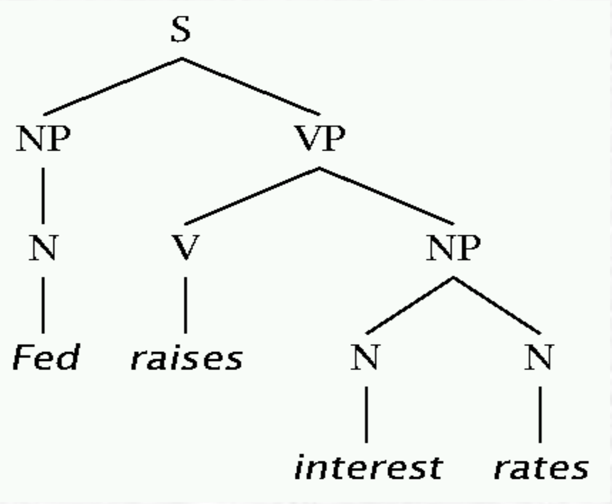
① Constituency (phrase structure) 成分树

② Dependency Structure 依赖树

Constituency (phrase structure)

Phrase structure organizes words into nested constituents.

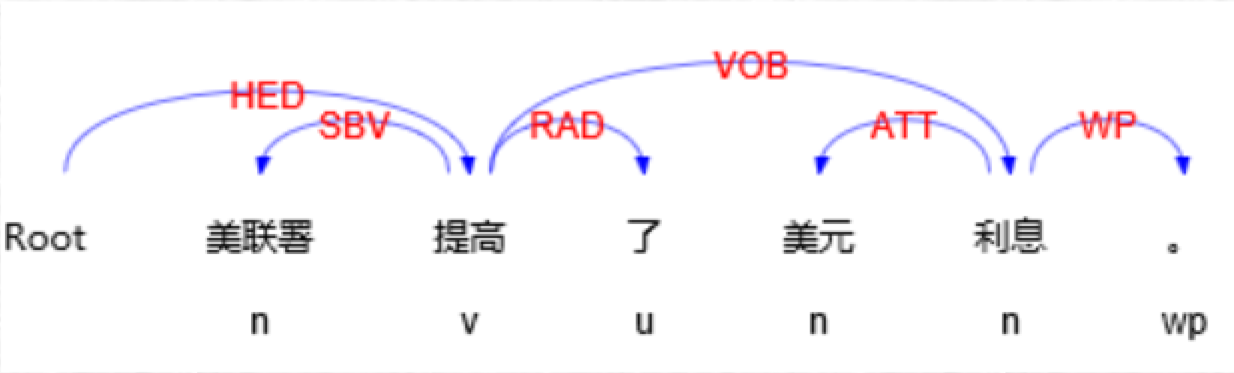
S → NP VP
NP→N | ADJ N | N N
VP→V NP | V



Dependency Structure

Dependency structure shows **which words depend on which other words.**

- The arrow connects a head with a dependent.
- Dependencies form a tree (connected, acyclic, single head)



Basic Text Processing (cont.)

- **5. Semantic Analysis**

Know the meaning of words and sentences.

A word may have different meaning:

A bank: financial institution / sloping land

The boy (person) *put the tortoise* (a kind of animal) *on the rug* (a material).

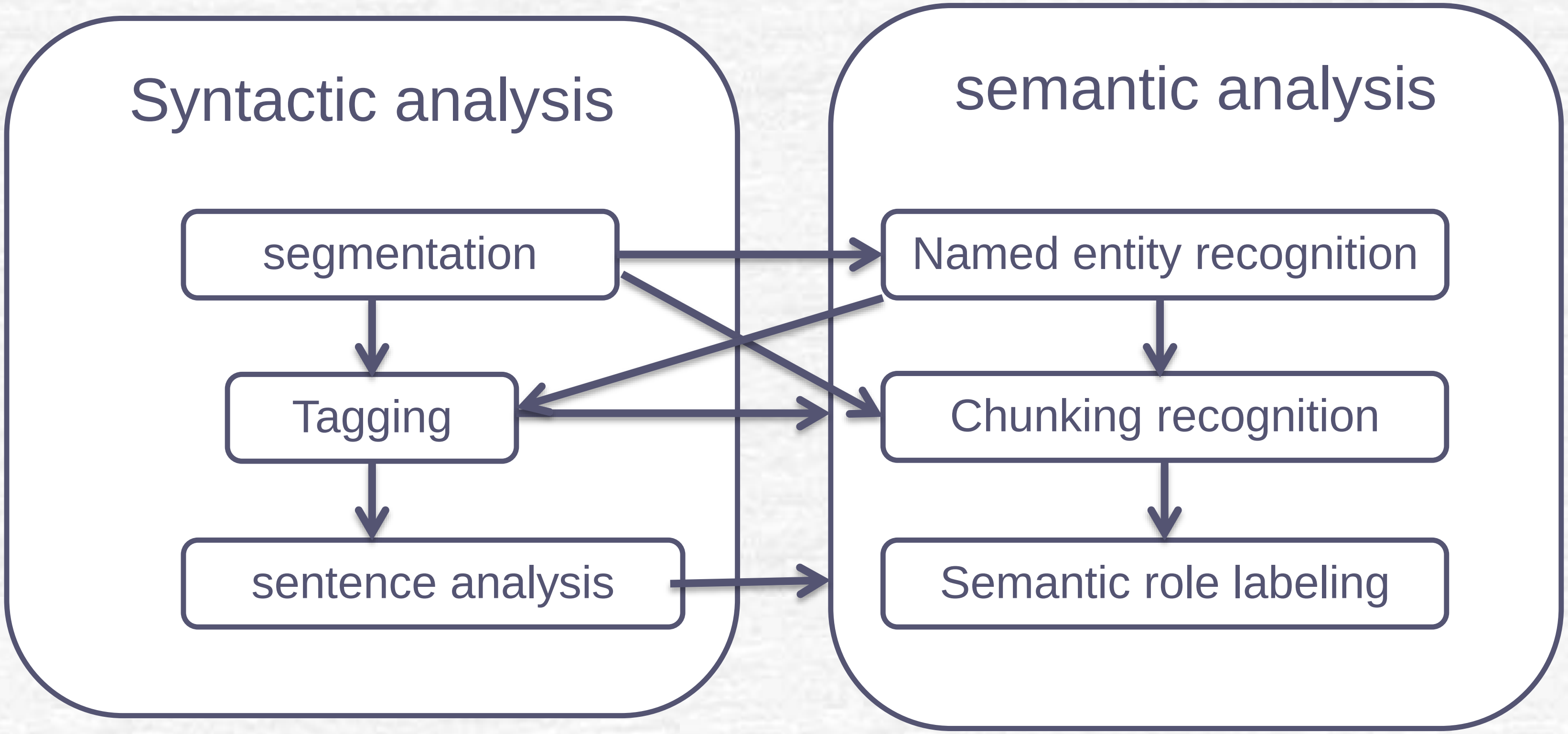
Basic Text Processing (cont.)

- 6. **Discourse Analysis**

Know the meaning of a paragraph, a text.

Tom (person) is a little boy. He (Tom) puts the tortoise(a kind of animal) on the rug (a material).

Summarization for Chinese Language Processing



Open Sources for NLP

Name	Down load address	Language
哈工大LTP框架	https://github.com/HIT-SCIR/ltp/releases	C++
Stanford NLP框架	http://nlp.stanford.edu/software/index.shtml http://corenlp.run/	Java
ICTCLAS分词系统	http://www.threedweb.cn/forum-2-1.html	C++
结巴分词系统	https://github.com/fxsjy/jieba	Python
Ansj 分词系统	https://github.com/NLPchina/ansj_seg	Java

哈工大语言分析系统的标注说明

- 句法分析

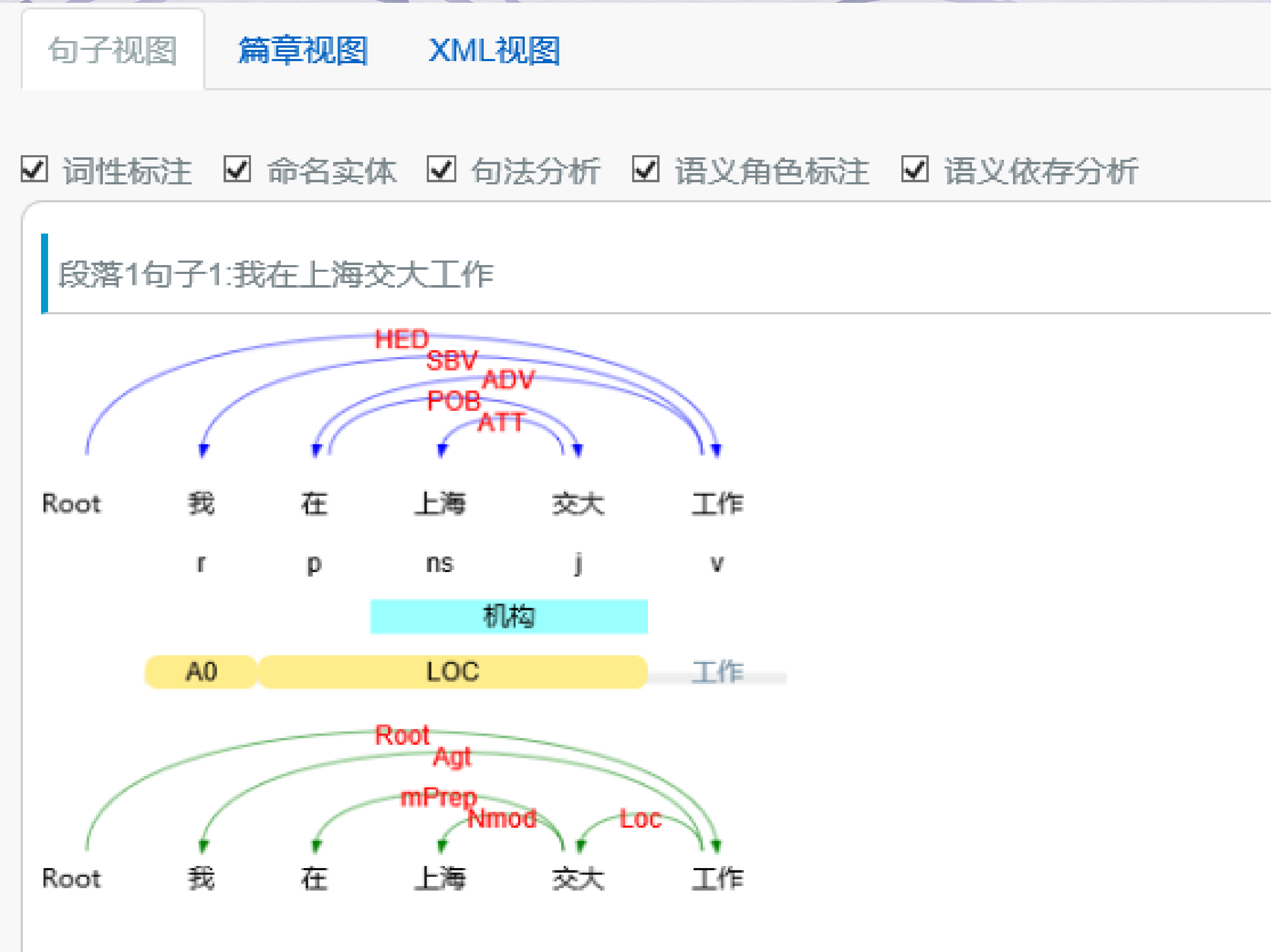
标记	解释	标记	解释
SBV	主谓关系	FOB	前置宾语
VOB	动宾关系	ADV	状中结构
IOB	间宾关系	CMP	动补结构
POB	介宾关系	IS	独立结构
ATT	定中关系	DBL	兼语
COO	并列关系	LAD	左附加关系
HED	核心关系	RAD	右附加关系

- 语义分析：

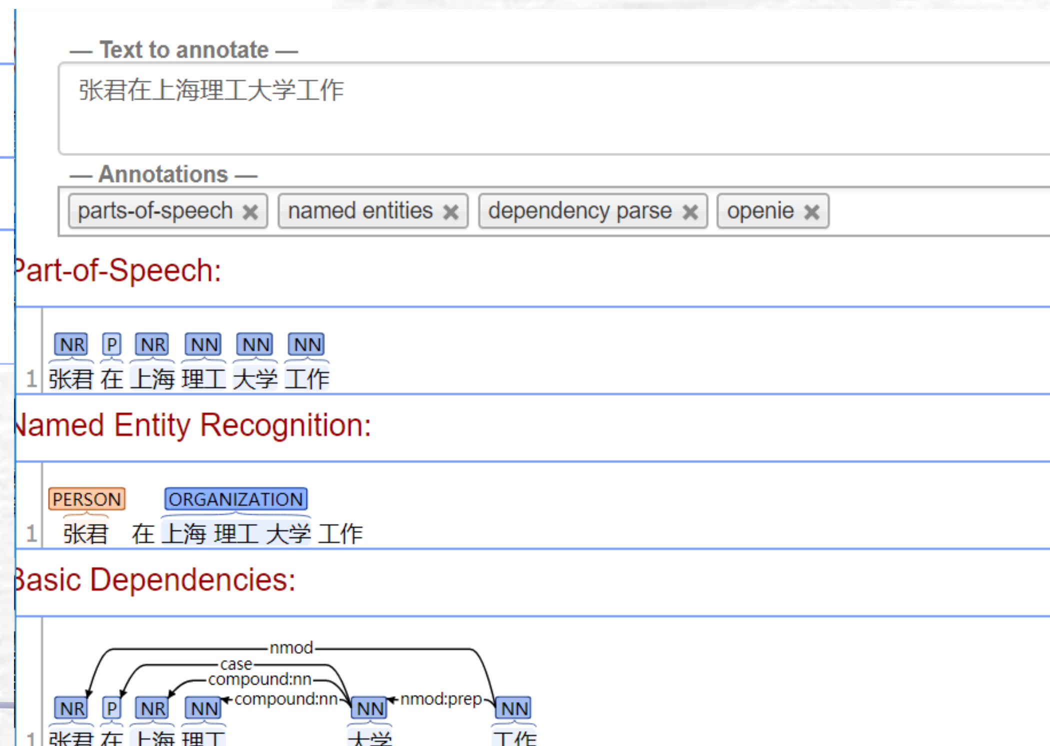
标记	解释	标记	解释
AGT	施事关系	LOC	空间角色
DATV	源事关系	mPrep	介词标记
ePURP	目的关系	Nmod	情态标记
eSucc	顺承关系	eCau	原因关系
mPunc	标点标记	eResu	结果关系
Pat	受事关系	。。。	
Root	根		

哈工大语言云演示

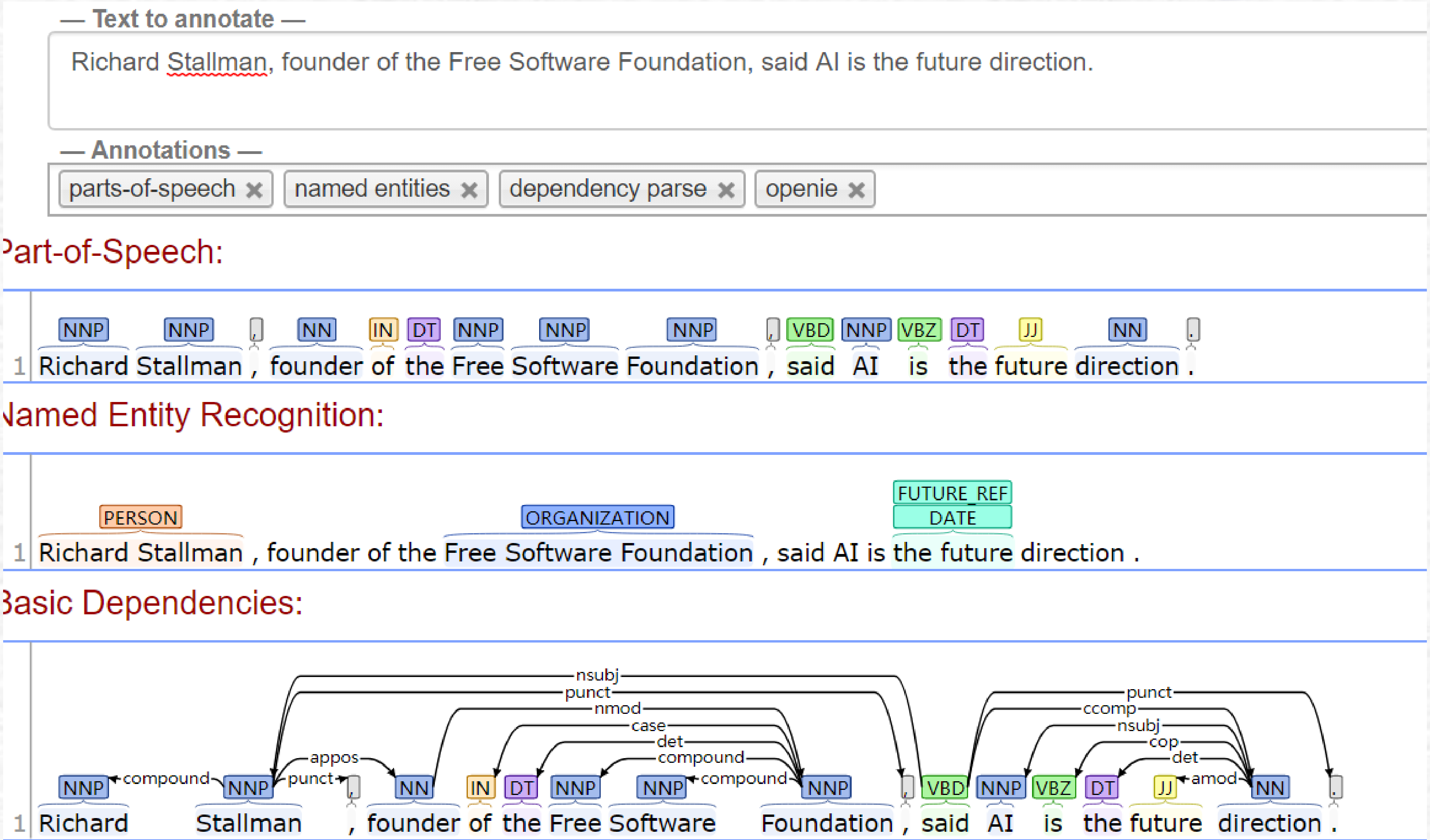
- http://www.ltp-cloud.com


句子视图
篇章视图
XML视图
词性标注
命名实体
句法分析
语义角色标注
语义依存分析
段落1句子1:我在上海交大工作
HED
SBV
ADV
POB
ATT
Root 我 在 上海 交大 工作
r p ns j v
机构
A0
LOC
工作
Root
Agt
mPrep
Nmod
Loc
Root 我 在 上海 交大 工作

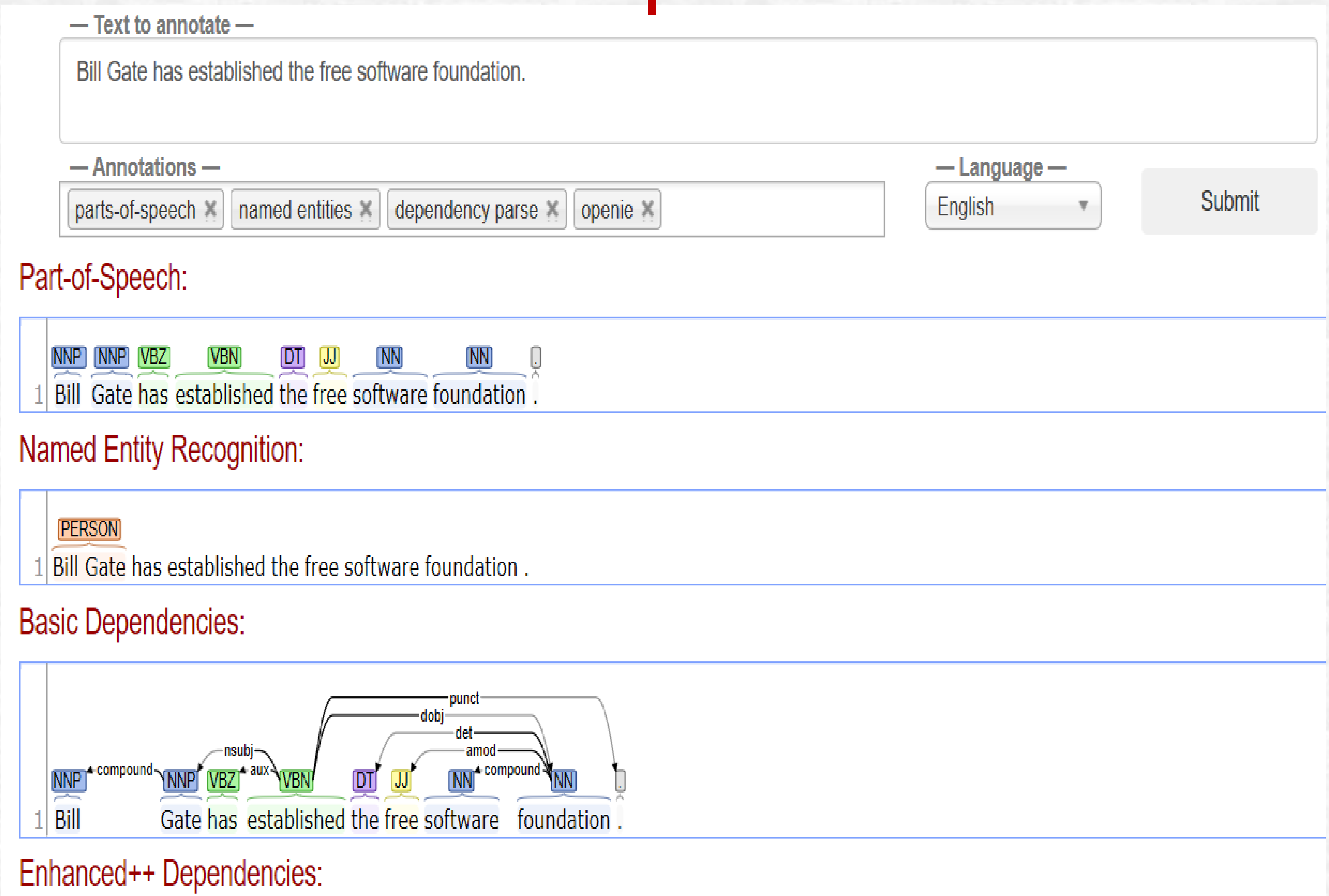
http://corenlp.run/



The probability of using 张俊 as a person name is less than 张君 . 张君is more offen than 张俊 as a person name.


— Text to annotate —
Richard Stallman, founder of the Free Software Foundation, said AI is the future direction.
— Annotations —
parts-of-speech
named entities
dependency parse
openie
Part-of-Speech:
NNP NNP , NN IN DT NNP NNP NNP , VBD NNP VBZ DT JJ NN .
1 Richard Stallman , founder of the Free Software Foundation , said AI is the future direction .
Named Entity Recognition:
PERSON
ORGANIZATION
FUTURE_REF
DATE
1 Richard Stallman , founder of the Free Software Foundation , said AI is the future direction .
Basic Dependencies:
nsubj
punct
nmod
case
det
compound
appos
punct
compound
ccomp
nsubj
cop
det
amod
NNP NNP , NN IN DT NNP NNP NNP , VBD NNP VBZ DT JJ NN .
1 Richard Stallman , founder of the Free Software Foundation , said AI is the future direction .

Why ？ Capitalization



How to <u>model</u> natural language ?

- using Grammar 需要人写文法
- using probability 需要大量语料

Language Model

- To assign a probability of a sentence.

For example,

"I am Smith" is a sentence.

P(I am Smith)=

$p(I)\ p(am \mid I)\ p(smith \mid I\ am) =$

$0.68 * 0.90 * 0.5 = 0.31$

$P(w_1,w_2,w_3,w_4,..w_n)=$

$p(w1)p(w_2 \mid w_1)...p(w_n \mid w_1...w_{n-1})$

Markov Assumption

Markov Assumption: only the prior local context—the last few words – affects the next word.

$P(w_n \mid w_1, w_2, w_3, \ldots w_{n-1})$

n=1 unigram p(w)=p(w1)p(w2)p(w3)…

n=2 bigram p(w)=p(w1)p(w2|w1)p(w3|w2)…

n=3 trigram p(w)=p(w1) p(w2|w1)p(w3|w2,w1) p(w4|w3,w2)…

n=…

N-gram Language Model

N-gram language Model:

The task of predicting the next word can be stated as attempting to estimate the probability function p

Bigram: $p(w_i|w_{i-1})$

Parameters estimation :

$P(A|B)=P(A,B)/P(B)$

Bigram: $p(w_i|w_{i-1}) = \text{count}(w_{i-1},w_i)/\text{count}(w_{i-1})$

Example of Bigram model

<s>I am smith </s>

<s>smith I am </s>

<s>I do not like eggs and ham </s>

P(I | <s>)=2/3= 0.67

P(am | I)=2/3= 0.67

P(smith | am)=1/3=0.33

P(</ s >| smith>)=1/3=0.33

P(smith | <s>) =1/3

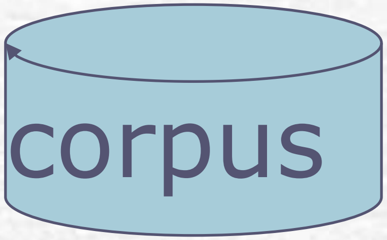
....

Language Model:
Example of Bigram model (cont.)

- Based on those probabilities, we can **predict** a word given a previous word.

I saw a car (0.3)

I saw a bird (0.6) √



Language Model: Estimation of n-gram model

- suppose V=20000 words.

Bigram model parameters be 20000*19999=400 million

Trigram model para. = 8 trillion

- Growth in number of parameters for n-gram model, <u>unigram</u> and <u>bigram</u> are often used.

Language Model: Smoothing

- <s>I am smith </s> V=9
- <s>smith I am </s>
- <s>I do not like eggs and ham </s>

P(like | I)= 0/3=0   not true

改进公式:

P(A|B) = (count(A,B) +1) / （count(B)+V）

P(like | I)= 1/12= 0.08

P(am| I)=3/12= 0.25 instead of 0.67

Unknown Words

- OOV words: out of vocabulary
- Create an unknown word token <UNK>
- Training of <UNK> probabilities

1. Create a fixed lexicon L of size V
2. At text normalization phase, any training word not in L changed to <UNK>
3. Now we train its probabilities like a normal word

- Use UNK probabilities for any word not in L

Unknown Words (example)

- <s>I am smith </s> V=7
- <s>smith I am </s>
- <s>I do not like eggs and ham </s>

P(A|B) = (count(A,B) +1) / （count(B)+V）

P(UNK)=2/11=0.18

P(UNK | I)=1/10=0.1

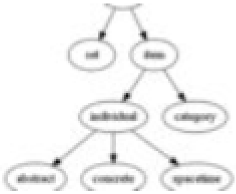
P(am | I) =3/10=0.3

Search Engines

Input keyword: artificial intelligence

How to find webpages to match the keyword?

Artificial intelligence - Wikipedia



Artificial intelligence (AI) is the intelligence exhibited by machines or software. It is an academic field of stu...

History   Research   Applications   Philosophy and ethics

en.wikipedia.org/wiki/... - 百度快照 - 翻译此页

Artificial Intelligence - Journal - Elsevier

Artificial Intelligence, which commenced publication in 1970, is now the generally accepted premier international forum for the publication of...

www.journals.elsevier.... - 百度快照 - 翻译此页

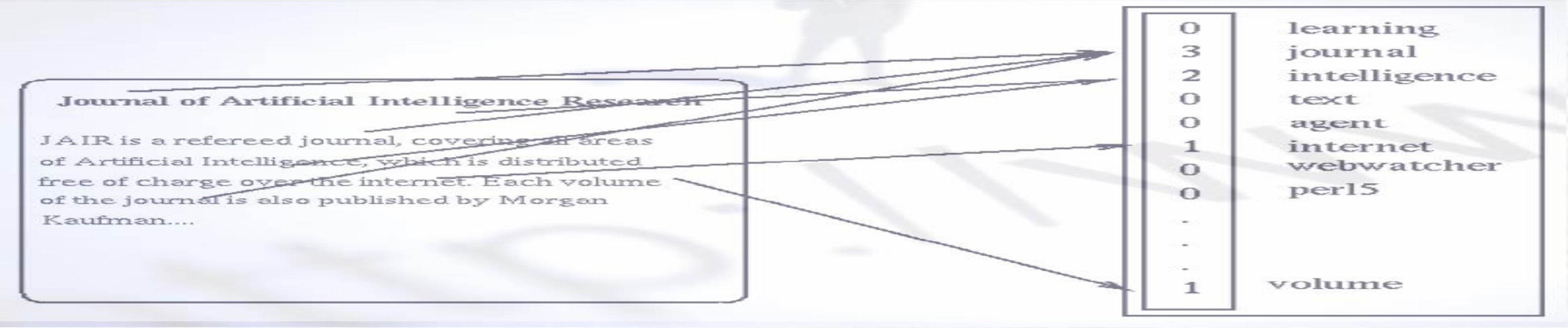
Artificial Intelligence: Friendly or Frightening?

The field of artificial intelligence is probably a long way from achieving "the singularity." But some experts say humanity isn't doing ...

www.livescience.com/49... - 百度快照 - 翻译此页

Vector Space Model

- **keyword** → a vector
- **Web Page** → a vector



- Search Engine calculates the similarity of two vectors to find the related web pages.

Vector Space Model

Word: one hot vector (only one 1, the other is 0)

How the web page → a vector ?

- Dimension: each word is as a dimension, there are V dimensions.
- Vocabulary (V): the size of vocabulary.

Term-Document Matrix. If the word appears in the webpage, the cell will be 1, otherwise is 0.

Dictionary	Web page 1	Web page 2
a	1	0
brown	1	0
dog	0	1
fox	1	0
jumped	0	1
lazy	0	1
over	0	1
quick	1	0
the	0	1

Term-Document Matrix (example)

Assume: 网页或文档相似，向量也相似.

- Three books in the following.
- 4 words as a vector space.

Each cell: count of term *t* in a document *d* : $tf_{t,d}$

	Java programming	Health Guide	Python Language
apricot	0	58	1
pinapple	0	60	2
digital	37	5	50
information	117	30	200

Term-Document Matrix (tf idf)

	Java programming	Health Guide	Python Language
apricot	0	12	1
pinapple	0	23	2
digital	15.34	3.6	16
information	18.36	4.8	19.7

The most popular weighting schema is normalized word frequency *tfidf*:

$$tfidf(w) = tf \cdot \log\left(\frac{N}{df(w)}\right)$$

***tf(w)* –term frequency (number of word occurrences in a document)**

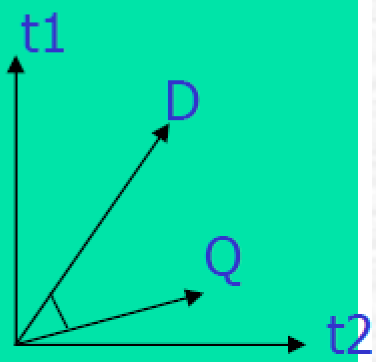
***df(w)* –document frequency (number of documents containing the word)**

***N* –number of all documents**

***tfidf(w)* –relative importance of the word in the document**

Measuring Similarity in Vector Space Model

Dot : $$Sim(D,Q) = D \bullet Q = \sum_i (a_i \times b_i)$$



Cosine : $$Sim(D,Q) = \frac{D \bullet Q}{\| D \| \times \| Q \|} = \frac{\sum_i (a_i \times b_i)}{\sqrt{\sum_i a_i^2 \times \sum_i b_i^2}}$$

Dice : $$Sim(D,Q) = \frac{2 \times D \bullet Q}{\| D \|^2 + \| Q \|^2} = \frac{2\sum_i (a_i \times b_i)}{\sum_i a_i^2 + \sum_i b_i^2}$$

Jaccard : $$Sim(D,Q) = \frac{D \bullet Q}{\| D \|^2 + \| Q \|^2 - D \bullet Q} = \frac{\sum_i (a_i * b_i)}{\sum_i a_i^2 + \sum_i b_i^2 - \sum_i (a_i * b_i)}$$

Example

	Java programming	Health Guide	Python Language
apricot	0	12	1
pinapple	0	23	2
digital	15.34	3.6	16
information	18.36	4.8	19.7

Is Java Programming similar (P) similar to Health guide (H) or Python Language(L)?

- Dot: Sim(P,H)=143.35 Sim(P,L)=245.76
- Cos: Sim(P,H)= 0.22 Sim(P,L)=0.99

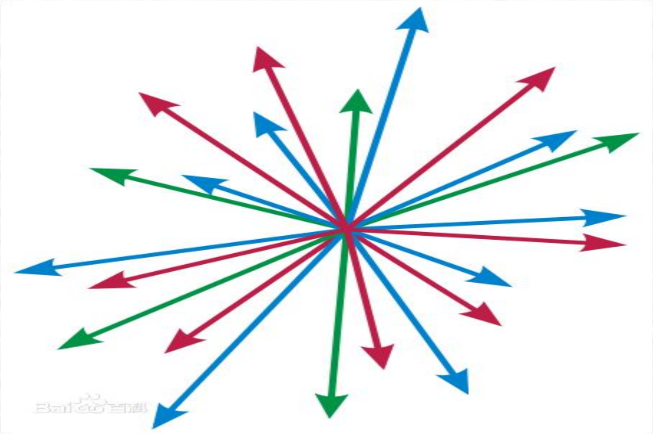
The Problems of Dot Product

- ✓ High when two vectors have large values in the same direction.
- ✓ Low for orthogonal vectors.
- ✓ Sensitive to word frequency.

Word vs. Vector

How to represent a word with a vector in deep learning?

- One hot-vector (VSM)
- Dense vector (DL)



One-hot vectors

- A vector of length |V|
- 1 for the target word and 0 for other words

$$\begin{array}{cccccccccccccccccccc} w_0 & w_1 & & & & & & & & w_j & & & & & & & & & & w_{|V|} \\ 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \end{array}$$

- if "pineapple" is vocabulary word 5
- The **one-hot vector of pineapple** is
- [0,0,0,0,1,0,0,0,0…….0]

Problems of the one hot Vector

- The real matrix is V x V. V is all the words, such as 50,000 x 50,000
- It is very **sparse,** most values are 0.
  - lots of efficient algorithms for sparse matrices.

Why Dense Vectors

- Generalize better than storing explicit counts
- Do better at capturing synonymy

What is a Dense Vector

- Pineapple:

[-0.24,-0.2,0.5,0.15,-0.01]

instead of [0,0,0,0,1,0,0,0,0…….0]

Sparse & Dense Vectors

- Sparse vectors
  - ✓ Long (length V=20000 to 50000)
  - ✓ Sparse (most are zero)

w_0	w_1									w_j										$w_{|V|}$
0	0	0	0	0	…	0	0	0	0	1	0	0	0	0	0	…	0	0	0	0

- **Dense vectors**
  - ✓ **Short (length: 200-1000)**
  - ✓ **Dense (most are non-zero)**

W0,W1,	**W_{200}**
0.32 ,0.45, -0,78, 0.11,0.32,…	**0.56**

How to generate a dense vector?

- From **the neural network models** used for language modeling.

Dense Vector

- Assumption:

the meaning of a word is represented by its context（上下文）.

- For example:
  - A bottle of tesgüino is on the table
  - Everybody likes tesgüino
  - Tesgüino makes you drunk
  - We make tesgüino out of corn.

→ An alcoholic beverage like beer

tesgüino

Bottle	0.7
Table	0.5
Like	0.45
Make	0.4
Drunk	0.8
Corn	0.78
…	…

如何确定词的上下文
Word Contexts (±7)

sugar, a sliced lemon, a tablespoonful of	**apricot**	preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	…
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
computer	0	2	1	0	1	0	
informatio	0	1	6	0	4	0	
…	…						

Apricot is similar to *pineapple*, while *computer* is similar to *information* based on their vectors.

The *long the contexts*, the more semantic representation (±4-10)

The *shorter the contexts*, the more syntactic the representation (±1-3)

如何确定文档中词的上下文 window approach

A sliding window approach

- A sequence of 2c+1 words. The middle word is called the *focus word*, and the c words to each side are the *contexts*.

For example, if c=1

$W_1 W_2 W_3 W_4 W_5 W_6 W_7 W_8 \ldots W_n$

如何产生词的稠密向量表示?

- Word2Vector
- Glove
- …

所有的**词**都用**向量**来表示→计算机可以**处理**

Word2Vectors: Skip-Gram & CBOW

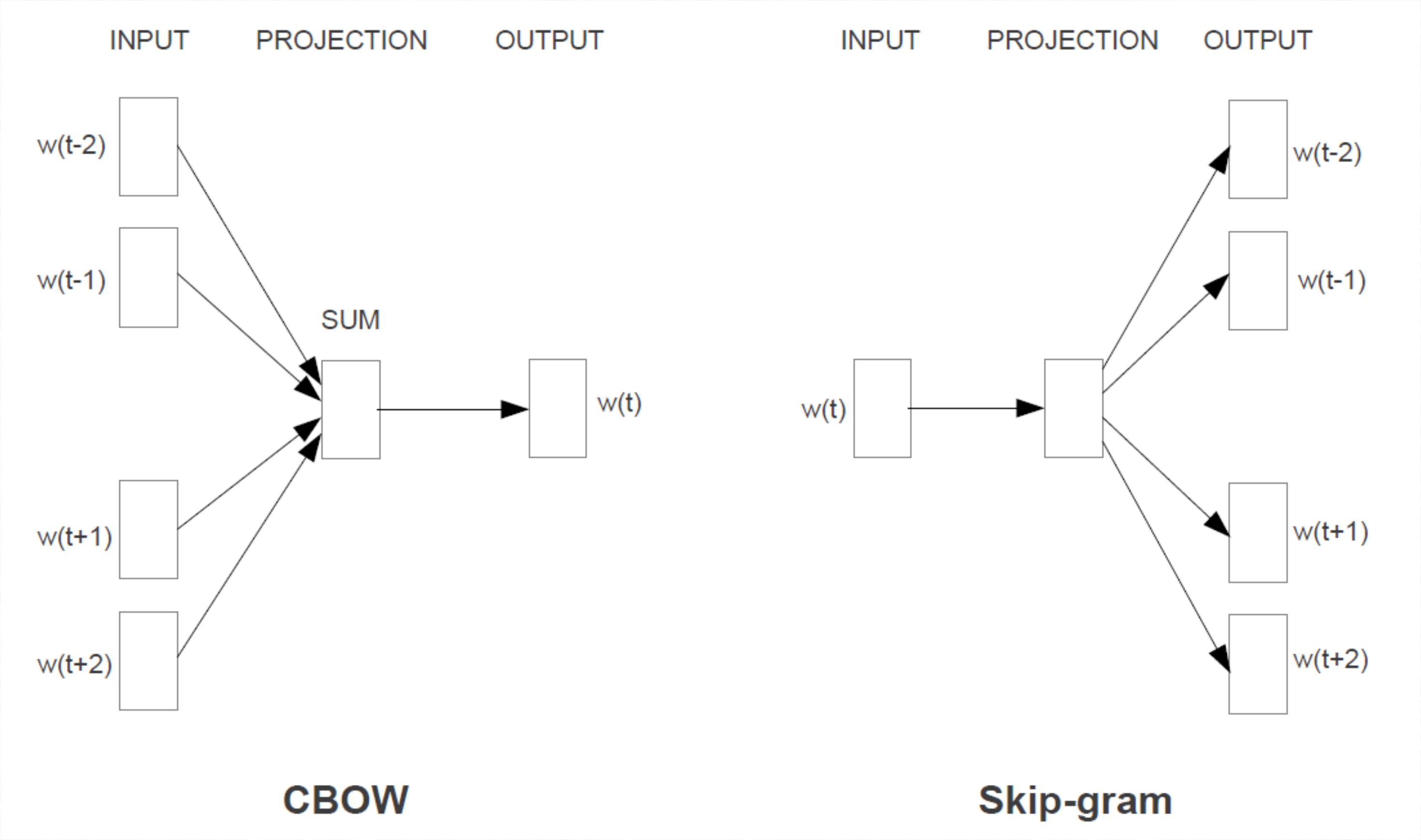
- By Google in 2013.
- Learn embeddings (*the vector representation*) as part of the process of word prediction.
- Advantage:

✓ Fast, easy to train

✓ Available online

✓ Including sets of pretrained embeddings

CBOW & Skip-gram



CBOWs

- Predict the current word based on
  - a context window of 2*C* words
- For example C=2, we are given word:

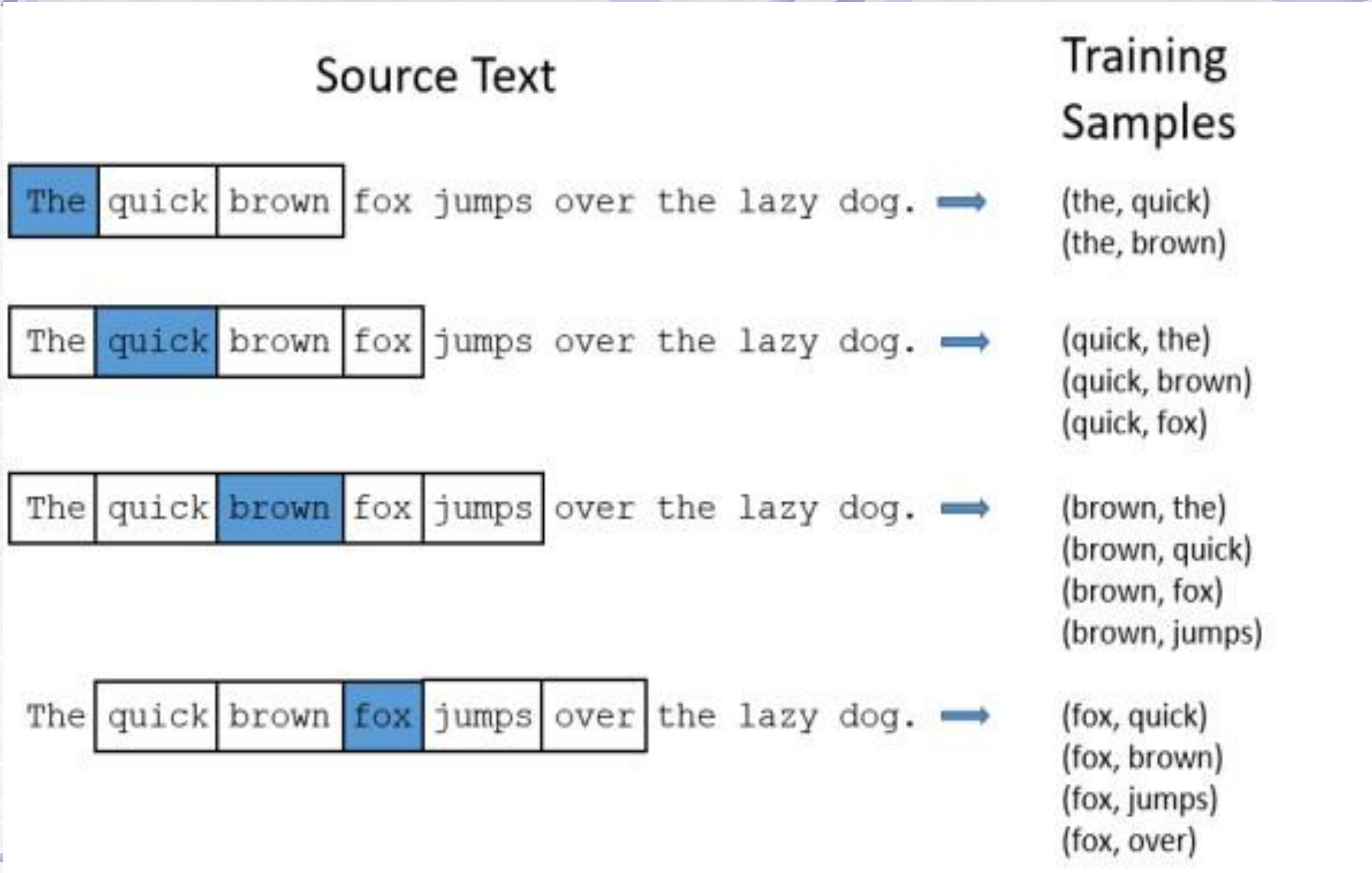
$$\left[w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2} \right]$$

to predict the w_t

Skip-Grams

- Predict each neighboring word
  - in a context window of 2*C* words
  - from the current word.
- For example C=2, we are given word w_t and **predicting these 4 words:**

$$\left[w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}\right]$$


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)

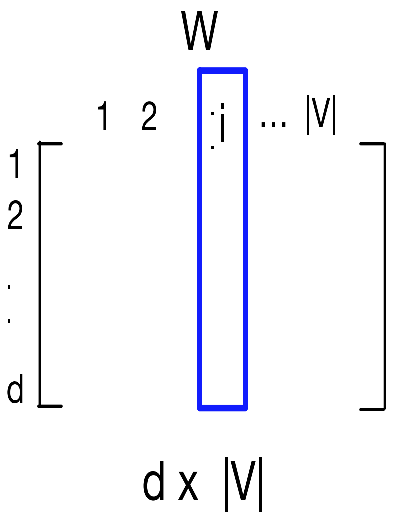
Skip-grams learn 2 embeddings for each w

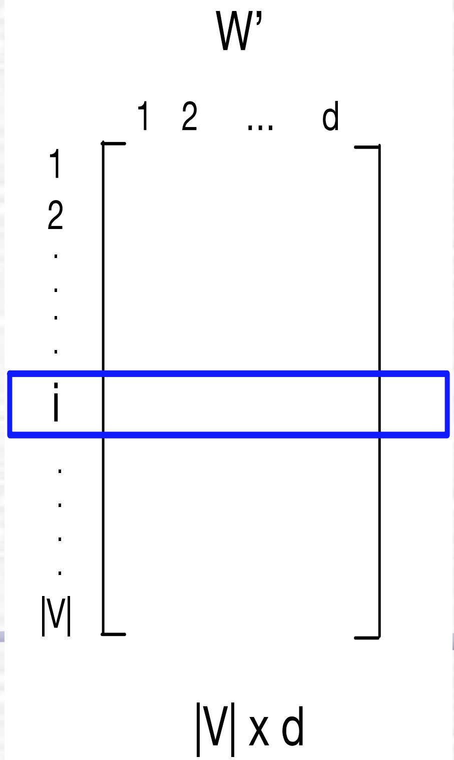
input embedding *v*, in the input matrix *W*

- Column *i* of the input matrix *W* is the $1 \times d$ embedding v_i for word *i* in the vocabulary.

output embedding *v′*, in output matrix W’

- Row *i* of the output matrix *W′* is a $d \times 1$ vector embedding v'_i for word *i* in the vocabulary.



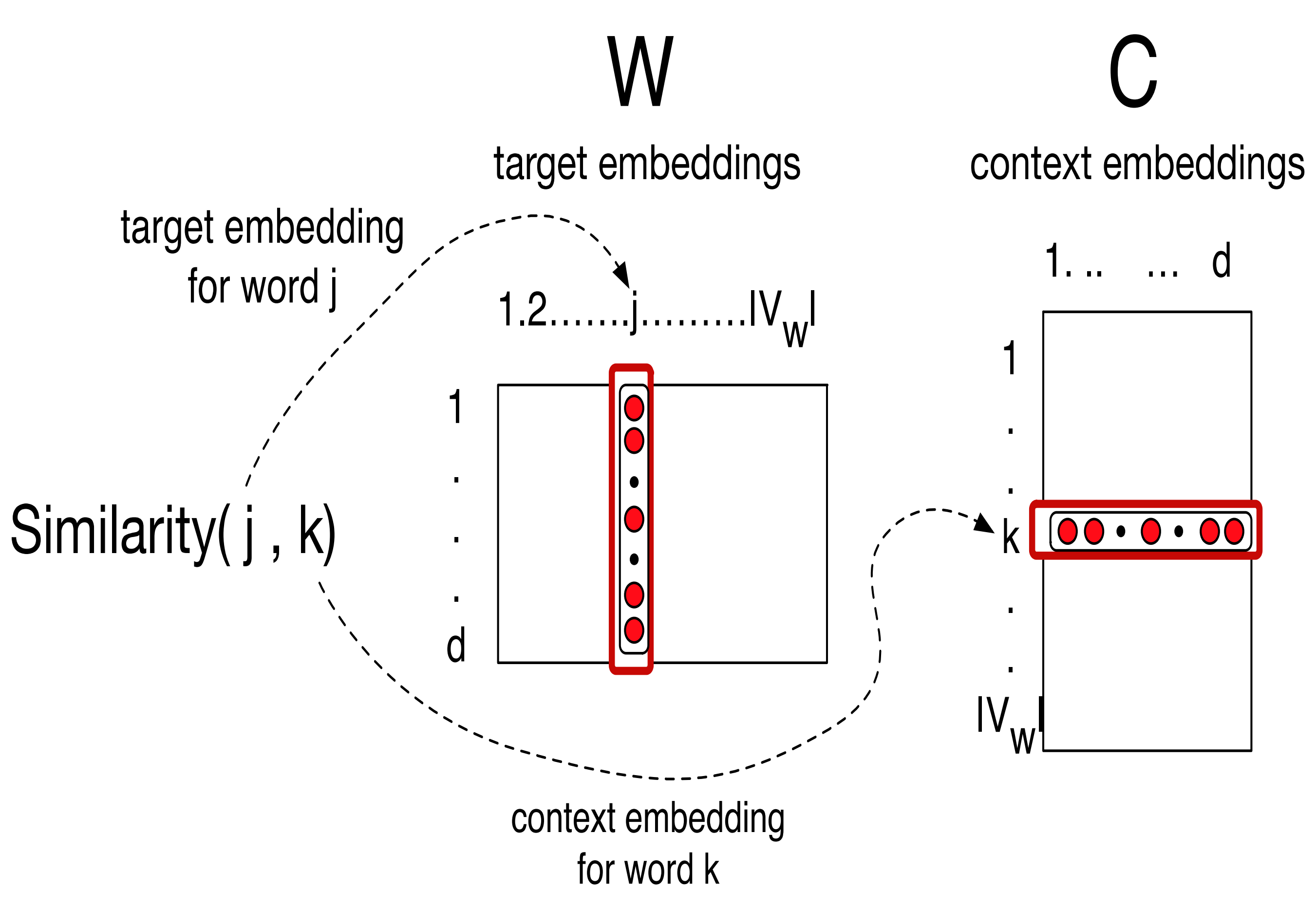


Setup

- Walking through corpus pointing at word $w(t)$, whose index in the vocabulary is j, so we'll call it w_j $(1 < j < |V|)$.

- Let's predict $w(t+1)$, whose index in the vocabulary is k $(1 < k < |V|)$. Hence our task is to compute $P(w_k | w_j)$.

Intuition: similarity as dot-product between a target vector and context vector



Similarity is computed from dot product

- Remember: two vectors are **similar** if they have a **high dot product**
  - Cosine is just a normalized dot product
- So:
  - $\text{Similarity}(j,k) \propto c_k \cdot v_j$

Turning dot products into probabilities

- Similarity(j,k) = $c_k \cdot v_j$
- Use softmax to turn into probabilities

$$p(w_k|w_j) = \frac{exp(c_k \cdot v_j)}{\sum_{i \in |V|} exp(c_i \cdot v_j)}$$

Learning

- Start with some initial embeddings (e.g., random)
- iteratively make the embeddings for a word
  - **more like** the embeddings of its neighbors
  - **less like** the embeddings of other words.

Problem with the Learning

- The denominator: have to compute over **every word in vocab**

$$p\left(w_k \middle| w_j\right) = \frac{exp\left(c_k \cdot v_j\right)}{\sum_{i \in |V|} exp\left(c_i \cdot v_j\right)}$$

- Instead: just sample <u>a few of those negative words (non neighbor words)</u>

Goal in learning

- Make the word like the context words

```
lemon,   a [tablespoon of apricot preserves or] jam
              c1          c2     w       c3         c4
```

$$\sigma(x) = \frac{1}{1+e^{x}}$$

- We want this to be high:

$$\sigma(c1 \cdot w) + \sigma(c2 \cdot w) + \sigma(c3 \cdot w) + \sigma(c4 \cdot w)$$

- not like *k* randomly selected "noise words"

```
[cement metaphysical dear coaxial   apricot attendant whence forever puddle]
 n1     n2               n3   n4            n5        n6     n7       n8
```

- We want this to be low:

$$\sigma(n1 \cdot w) + \sigma(n2 \cdot w) + \ldots + \sigma(n8 \cdot w)$$

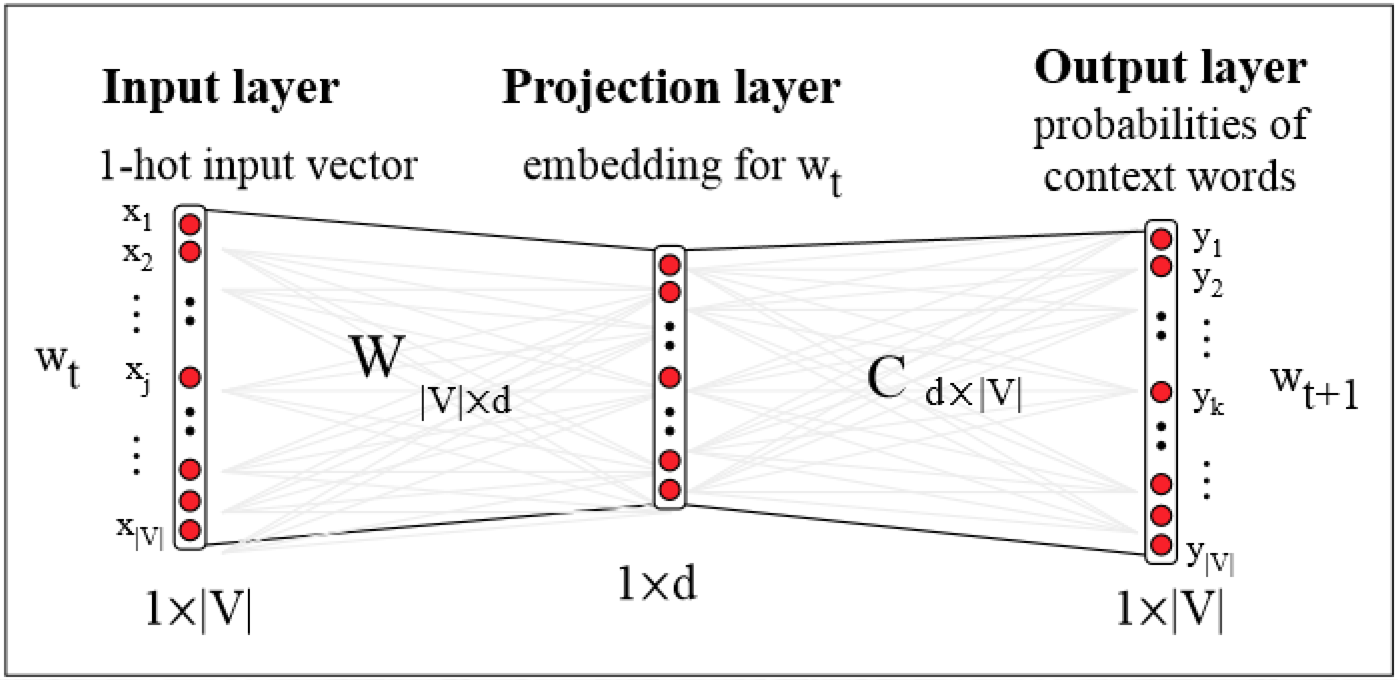
Skipgram with negative sampling: objective function

用SGD 算法优化这个目标函数

$$\log \sigma(c \cdot w) + \sum_{i=1}^{K} \mathbb{E}_{w_i \sim p(w)} [\log \sigma(-w_i \cdot w)]$$

we want to maximize the dot product of the word with the actual context words, and minimize the dot products of the word with the k negative sampled non-neighbor words. The noise words wi are sampled from the vocabulary V according to their weighted unigram probability;

Visualizing the network



稠密词向量的特性

- What are the properties of word vectors?

Properties of embeddings

- Nearest words to some embeddings

For example

Input word: 文化

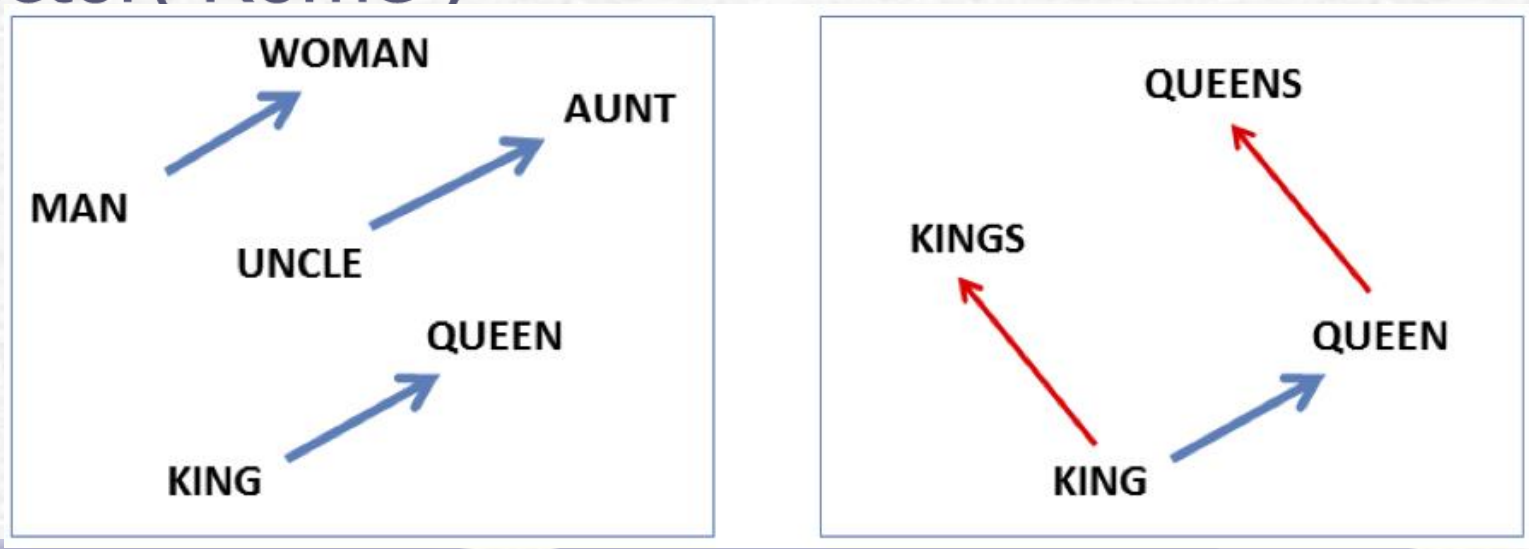
Output words:

博大精深	0.839596
民间艺术	0.735981
体育运动	0.725998
东西方	0.678683

Embeddings capture **relational meaning**

vector('*king*') - vector('*man*') +vector('*woman*') ≈ vector('queen')

vector('*Paris*') - vector('*France*') +vector('*Italy*') ≈ vector('Rome')



Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship	spray paint grafitti taggers	capitulation capitulated capitulating

Table 6: Examples of the closest tokens given various well known models and the Skip-gram model trained on phrases using over 30 billion training words. An empty cell means that the word was not in the vocabulary.

Source Code of word2vectors

- Google: C implementation

http://word2vec.googlecode.com/svn/trunk

- Gensim: Python (https://radimrehurek.com/gensim/index.html)
- Java implementation (http://github.com/NLPchina/Word2VEC_java)
- **C++**： https://github.com/jdeng/word2vec

Summary

- How the computer processes a text.
- How a document represented by a vector
- how a word represented by a vector
- How to calculate the similarity of a word pair
- What is a word embedding?
- What is word2vectors?