國立高雄科技大學
National Kaohsiung University of Science and Technology

詞向量

報告人: 蘇佳益
指導老師: 陳聰毅
國立高雄科技大學建功校區電子工程系
https://github.com/chiayisu/Natural-Language-Processing

Agenda

- Word2Vector
- Negative Sampling
- Word2Vector 訓練結果
- Glove: Global Vectors for Word Representation
- 詞向量的評估
- 詞嵌入
- Name Entity Recognition (NER)
- 預訓練詞向量探討

Recap

- 計數手法的問題
  - 可能需要建立一個非常龐大的矩陣
  - 需耗費大量資源

推論手法

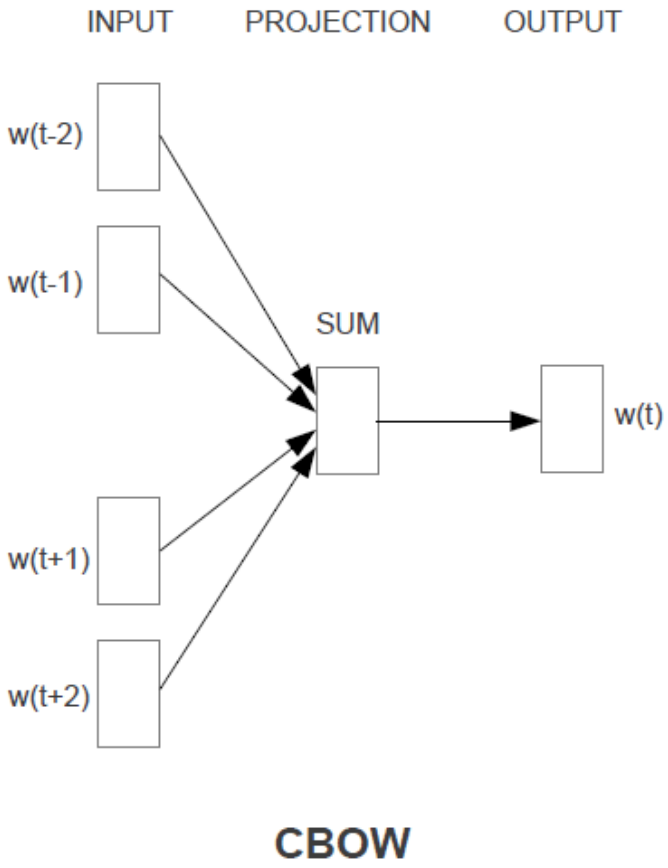
- 以類神經網路來推論
- 可以使用小批次學習並結合GPU進行平行運算以提高效率
- 給予上下文讓模型推論中心字詞
- EX: you ? Goodbye and I say hello.

Word2Vector

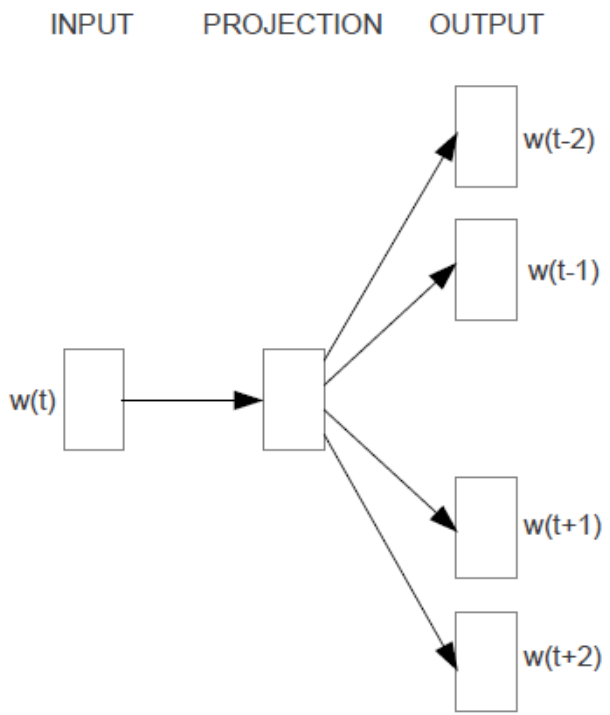
Introduction

- Bag-of-Words model (CBOW)
- Continuous Skip-gram model (Skip-Gram)

CBOW及Skip-Gram的架構



CBOW



Skip-gram

CBOW演算法標記

- w_i：所有語詞中的第i個語詞
- $\omega \in R^{n*|V|}$：輸入矩陣
- $U \epsilon R^{|V|*n}$：輸出矩陣

CBOW演算法

1. 將上下文表示成One-Hot Vector $\left(x^{(c-m)}, \ldots, x^{(c-1)}, \ldots, x^{(c+m)} \in R^{|V|}\right)$
2. 將One-Hot Vector 轉成Embedded Word Vector ($\hat{v} = \omega x^{(c-m)} + \cdots + \omega x^{(c+m)} \in R^{n}$)
3. 將Embedded Word Vector取平均$\hat{v} = \frac{\hat{v}}{2m} \in R^{n}$
4. 產生一個分數矩陣z =$U * \hat{v} \in R^{|V|}$
5. 將分數矩陣轉成機率分布 y = softmax(z) $\in R^{|V|}$
6. 利用梯度下降法更新輸入及輸出矩陣

Skip-Gram 演算法

1. 產生中心語詞的One-Hot Vector x ∈ $R^{|V|}$
2. 將One-Hot Vector 轉成Embedded Word Vector $v_c = \omega x \in R^n$
3. 產生一個分數矩陣z =$U * v_c \in R^{|V|}$
4. 將分數向量轉成機率分布$\hat{y} = softmax(z)$
5. 利用梯度下降法更新輸入及輸出矩陣

如何更新輸入及輸出矩陣

- CBOW

➢ Let J = $-\log P(w_c|w_{c-m},\dots,w_{c+m})$

$= -logP(u_c|\hat{v})$

$= -log\frac{\exp(u_c\hat{v})}{\sum_{j=1}^{|V|}\exp(u_c\hat{v})}$

$= -u_c\ \hat{v} + \log(\sum_{j=1}^{|V|}\exp(u_c\hat{v})$

- Skip-Gram

➢ Let J = $-\log P(w_{c-m},\dots,w_{c+m}|w_c)$

$= -\log\left(\prod_{j=0,j\neq m}^{2m} P\left(w_{c-m+j}\middle|\, w_c\right)\right.$

$= -\log\left(\prod_{j=0,j\neq m}^{2m} P\left(u_{c-m+j}\middle|\, v_c\right)\right.$

$= -\log\left(\prod_{j=0,j\neq m}^{2m}\frac{\exp\left(u_{c-m+j}^T v_c\right)}{\sum_{k=1}^{|V|}\exp(u_k^T v_c)}\right)$

$=-\sum_{j=0,j\neq m}^{2m} u_{c-m+j}^T v_c + 2m\log\left(\sum_{k=1}^{|V|}\exp(u_k^T v_c)\right)$

計數手法 vs. 推論手法

- 計數手法
  - 加入新語詞須從頭計算
  - 產生出來的向量只能用在相似度任務
- 推論手法
  - 加入新語詞不須從心計算
  - 在多種不同任務都有不錯的表現
  - 可藉由基本數學運算找出語詞間的關係
    - E.g. Vector(queen) = Vector(king) – Vector(men) + Vector(women)

Negative Sampling

Softmax 函數的壞處

- 需要正規化
- 演算法複雜度為 $O(|V|)$
- 因為：通常語詞為幾十億個
- 所以：計算$O(|V|)$次非常耗費運算資源

Introduction to Negative Sampling

- 將多值分類問題轉成二值分類的問題
- 目標：最大化正確語詞的機率；最小化負例語詞機率
- 只需採樣少數負例 (5-15個)
- 只需要循環整個負例的分布

目標函數推倒相關標記及定義

- (w, c) ：語詞及上下文
- $P(D = 1|w, c)$：正例在正例文本資料的機率
- $P(D = 0|w, c)$：負例在負例文本資料的機率
- θ：輸入及輸出矩陣
- 正例在文本資料的機率公式定義如下：

➢ $P(D = 1|w, c, \theta) = \sigma(v_c^T v_w) = \frac{1}{1+e^{-(v_c^T v_w)}}$

目標函數

- 目標：最大化正例在正例文本資料的機率及負例在負例文本資料的機率
- 公式推倒如下：
- $\theta = \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} p(D=1|w,c,\theta) \prod_{(w,c)\in \widehat{D}} p(D=0|w,c,\theta)$

  $= \underset{\theta}{\operatorname{argmax}} \prod_{(w,c)\in D} p(D=1|w,c,\theta) \prod_{(w,c)\in \widehat{D}} (1-p(D=1|w,c,\theta))$

  $= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log(p(D=1|w,c,\theta)) + \sum_{(w,c)\in \widehat{D}} \log(1-p(D=1|w,c,\theta))$

  $= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log(\frac{1}{1+\exp(-u_w^T v_c)}) + \sum_{(w,c)\in \widehat{D}} \log(1-\frac{1}{1+\exp(-u_w^T v_c)})$

  $= \underset{\theta}{\operatorname{argmax}} \sum_{(w,c)\in D} \log(\frac{1}{1+\exp(-u_w^T v_c)}) + \sum_{(w,c)\in \widehat{D}} \log(\frac{1}{1+\exp(u_w^T v_c)})$

目標函數

- 因為：最大化Log likelihood 等於最小化負的Log likelihood
- 因此：目標函數修改如下：
- $\mathrm{J} = -\sum_{(w,c)\in D} \log\left(\frac{1}{1+\exp\left(-u_w^T v_c\right)}\right) - \sum_{(w,c)\in \widehat{D}} \log\left(\frac{1}{1+\exp\left(u_w^T v_c\right)}\right)$
- $\widehat{D}$ 為負例文本資料

目標函數

- Skip-Gram 模型的目標函數可修改如下：
- $-\log(\sigma\left(u_{c-m+j}^{T} * v_c\right) - \sum_{k=1}^{K} \log(\sigma(-\hat{u}_k^T * v_c)$
- CBOW 模型的目標函數可修改如下：
- $-\log(\sigma(u_c^T * v_c) - \sum_{k=1}^{K} \log(\sigma(-\hat{u}_k^T * v_c)$
- 負例的採樣通常會將unigram模型中所計算出來的機率乘上0.75次方

Word2Vector 訓練結果

語意及句法測試

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

詞對關聯性測試

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Glove: Global Vectors for Word Representation

Pennington et al.

Glove: Introduction

- 分散式表示的方法
  - 計數手法 (e.g. co-matrix)
  - 推論手法 (e.g. skip-gram)
- 計數手法
  - 善用全域資訊
  - 然而：只有在相似度任務表現較佳
- 推論手法
  - 無法善用全域資訊
  - 然而：在多個任務都有良好的表現
- GLOVE 結合兩種方法的優點，其表現也優於上述兩種方法。

Glove Model: Motivation

	x = solid	x = gas	x = water	x = random
$P(x\|\text{ice})$	large	small	large	small
$P(x\|\text{steam})$	small	large	large	small
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	large	small	~1	~1

Glove Model: Motivation (CONT.)

	x = solid	x = gas	x = water	x = fashion
$P(x\|\text{ice})$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(x\|\text{steam})$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$\frac{P(x\|\text{ice})}{P(x\|\text{steam})}$	8.9	8.5×10^{-2}	1.36	0.96

標記介紹

- X : 共生矩陣
- X_{ij}： 語詞 j 出現在語詞i的上下文次數
- X_i : 所有語詞出現在語詞i的上下文次數 $(\sum_k X_{ik})$
- P_{ij} : 語詞j出現在語詞i的上下文的機率$P(j|i) = \frac{X_{ij}}{X_i}$

表格總結

- 假設 I = ice, j = steam, k = 任何測試的語詞
- k 如果與冰相關但與蒸氣無關其 $\frac{P_{ik}}{P_{jk}}$ 將會較大
- k 如果與冰無關但與蒸氣相關 $\frac{P_{ik}}{P_{jk}}$ 將會較小
- k 如果同時與冰和蒸氣無關或相關 $\frac{P_{ik}}{P_{jk}}$ 將會接近於1
- 根據上述總結， Pennington et al. 發現共生矩陣的重要性。

公式推倒

- $F\left(w_i, w_j, \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$ ，w 為詞向量，$\tilde{w}$ 將會在後面探討， F目前取決於許多不同的因素。
- $F\left(w_i - w_j, \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$，因為向量空間為線性，所以我們可以將向量做香減。F目前取決於兩個語詞的向量差。
- $F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$ ，因為左邊為向量，右邊為純量。其結構會模糊掉我們所要的線性結構，所以我們將左邊變點積。
- F應該要滿足於同態的性質，因此公式應該等於下列公式：
- $F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{F\left(w_i^T \tilde{w}_k\right)}{F(w_j^T \tilde{w}_k)}$, where $F\left(w_i^T \tilde{w}_k\right) = P_{ik} = \frac{X_{ik}}{X_i}$

公式推倒

- 因此，F = EXP函數，公式如下：

$$w_i^T \tilde{w}_k = \ln P_{ik} = \ln X_{ik} - \ln X_i \quad (1)$$

- 因為 $\ln X_i$與k無關，所以我們可以將它變成一個Bias b_i。此外，公式裡也增加一個 $\tilde{w}_k$ 的Bias 項。因此，公式如下：

$$w_i^T \tilde{w}_k + b_k + b_i = \ln X_{ik} \quad (2)$$

- 問題：
  - Log的值不能為0，否則將會發散。
- 解決辦法：
  - $\ln X_{ik}$ +1
  - 然而：這方法將會與共生矩陣相同，每個語詞的權重都相同。
  - 因此：演算法將會根據公式 (2)設計。

公式推倒

- Pennington et al. 採用 weighted least square regression模型及權重函數$f(X_{ij})$來解決此問題。因此，損失函數如下：
- $J = \sum_{i,j}^{V} f(X_{ij}) \left(w_i^T \tilde{w}_k + b_k + b_i - \ln X_{ik}\right)^2$, V為語詞的數量。(3)
- $f(X_{ij})$ 應該滿足於下列規則：

1. f(0) = 0
2. f(x) 應該為非遞減函數，如此一來，頻率較低之語詞的權重比較不會過高。
3. f(x) 也不應該讓頻率較高之語詞的權重過大。

公式推倒

- 雖然有許多滿足於上述條件之公式，但是表現較好的公式如下：

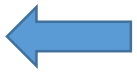
$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\propto} & \text{if } x < x_{max} \\ 1, otherwise \end{cases}$$

- 根據實驗，當 $\propto$ 等於 $\frac{3}{4}$ 有較好之結果。而$\frac{3}{4}$ 也與Mikolov et al.的Word2Vector的論文相同。

與其他模型的關聯性

- Mikolov et al. 在Word2Vector的論文裡採用了softmax函數來計算語詞 j 出現在語詞i的上下文的機率。因此，公式如下：

$$Q_{i,j} = \frac{\exp\left(u_j^T \hat{v}_i\right)}{\sum_{j=1}^{|V|} \exp\left(u_j^T \hat{v}_i\right)}$$



The most costly one

- Cross-Entropy 計算如下：

$$J = -\sum_{i \in corpus} \sum_{j \epsilon context} \ln Q_{i,j}$$

- 因為語詞 i 和語詞 j可能出現多次，因此我們可以將他用共生矩陣來表示，以提高效率。

$J = \sum_{i=1}^{w} \sum_{j=1}^{w} X_{ij} \ln Q_{i,j}$, where X is a co-occurrence matrix.

與其他模型的關聯性

- Cross-Entropy Loss的壞處
  - 花費較多的運算量
  - 原因：需要正規化
- Pennington et al. 採用 least square 目標函數。因此，不須做正規化。公式如下：
- $J = \sum_{i=1}^{w} \sum_{j=1}^{w} X_{ij} \left(\widetilde{P_{ij}} - \widetilde{Q_{i,j}} \right)^2$, where $\widetilde{P_{ij}} = X_{ij}$ and $\widetilde{Q_{i,j}} = \exp\left(u_j^T \hat{v}_i \right)$ which are all unnormalized term.

與其他模型的關聯性

- 因為X_{ij}通常值非常大，所以這也讓我們很難做最佳化參數。因此，我們將公式修改成下列公式：
- $J = \sum_{i=1}^{w} \sum_{j=1}^{w} X_{ij} \left(\ln(\widetilde{P_{ij}}) - \ln(\widetilde{Q_{i,j}}) \right)^2$

  $= \sum_{i=1}^{w} \sum_{j=1}^{w} X_{ij} \left(u_j^T \hat{v}_i - \ln(X_{ij}) \right)^2$ (4)
- Mikolov et al. 在 Word2Vector論文裡也提到藉由資料的過濾能夠提升整體表現，因此Pennington et al. 採用權重函數來解決此問題，公式(4)修改如下：

$$\sum_{i=1}^{w} \sum_{j=1}^{w} f(X_{ij}) X_{ij} \left(u_j^T \hat{v}_i - \ln(X_{ij}) \right)^2$$

- 因此，與公式(3)為相同的公式。

Word Analogy Explanation

- 兩種類比的問題
  - 語意上的類比
  - 語法上的類比
- 語意的類比問題
  - a is to b as c is to _
  - San Francisco is to California as Chicago is to _
- 語法上的類比問題
  - Good is to better as bad is to _
- 此問題主要根據下列代數運算來找出答案
  - $w_b - w_a + w_c$

Experiment on Word Analogies Task

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

Experiment on Word Similarity Task

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Corpora and Training Detail

- Corpora
  - 2010 wiki (1 billion tokens)
  - 2014 wiki (1.6 billion tokens)
  - Gigaword 5 (4.3 billion tokens)
  - Combination of the Gigaword5 + and wiki 2014 (6 billion tokens)
- Tokenization
  - Stanford Tokenizer
- Co-Occurrence Matrix X
  - In this step, the paper used decreasing weighted function which means the d word apart will contribute 1/d to the total count.

Corpora and Training Detail

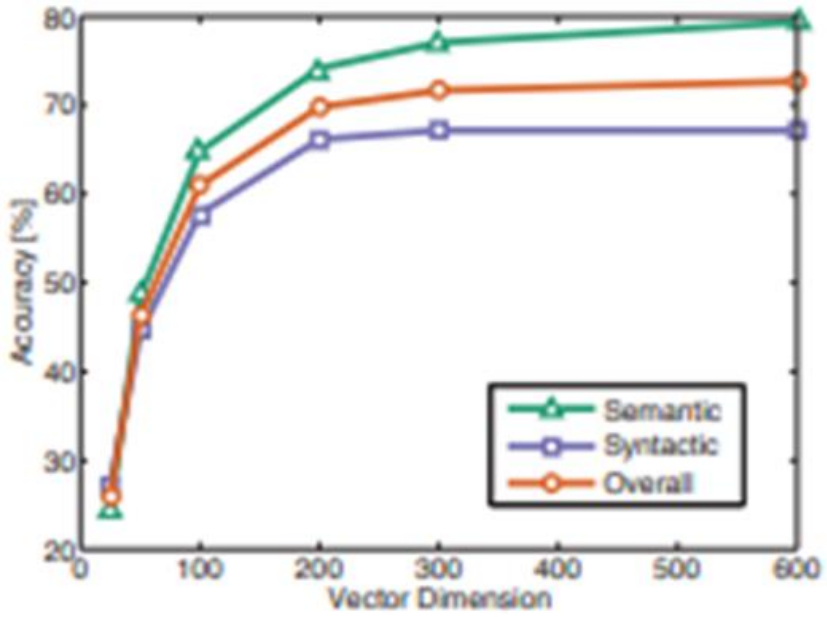
- Parameter Setting
  - $x_{max} = 100$
  - $\alpha = 3/4$
  - Learning rate = 0.05
  - Dimension is less that 300
  - The windows size is 10 to the right and left.

Discussion on $\widetilde{W}$ and W

- 兩個參數為相同的參數除了其亂數所初始化的值
- 為何要有$\widetilde{W}$?
  - 使用多個參數並將其結果相加有助於降低 overfitting 的問題及雜訊。
- 因此：Pennington et al. 把 $\widetilde{W}$ + W 當成詞向量。

Model Analysis: Vector Length and Context Size

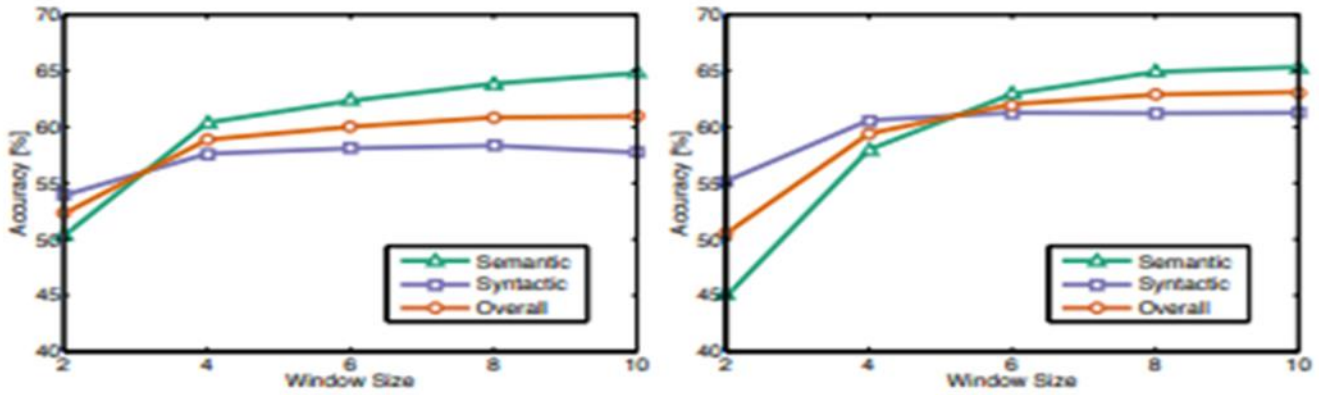
- 下面圖形顯示如果項量的維度大於200將會變平滑的曲線。



(a) Symmetric context

Model Analysis: Vector Length and Context Size

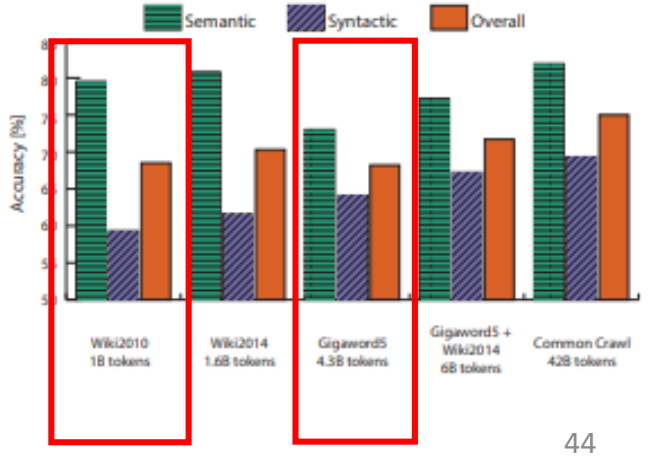
- 下圖顯示
  - 語法任務在小視窗及非對稱的視窗表現較佳
  - 語意任務在較大的視窗表現較佳
- 在語法任務上最好要考慮全域資訊。



(b) Symmetric context (c) Asymmetric context

Model Analysis: Corpus Size

- 正確率
  - 文本資料集越大，語法任務的正確率越高
  - 然而：語法任務的正確率並不完全取決於文本資料集大小
- Example
  - 較大資料集的Giagaword 文本的正確率並未比較小資料集的維基文本高
  - 原因： Giagaword (News)文本並未隨著新知識更新而更新



詞向量的評估

評估詞向量的方法

- Intrinsic Evaluation
- Extrinsic Evaluation

Intrinsic Evaluation

- 使用子系統來評估詞向量
- 運算較快速
- 需要與真實的系統有所關連
- 使用Intrinsic Evaluation 的原因
  - 使用整個系統來評估詞向量是非常花費時間及資源的

Intrinsic Evaluation – Word Analogy

Input	Result Produced
Chicago : Illinois : : Houston	Texas
Chicago : Illinois : : Philadelphia	Pennsylvania
Chicago : Illinois : : Phoenix	Arizona
Chicago : Illinois : : Dallas	Texas
Chicago : Illinois : : Jacksonville	Florida
Chicago : Illinois : : Indianapolis	Indiana
Chicago : Illinois : : Austin	Texas
Chicago : Illinois : : Detroit	Michigan
Chicago : Illinois : : Memphis	Tennessee
Chicago : Illinois : : Boston	Massachusetts

Input	Result Produced
bad : worst : : big	biggest
bad : worst : : bright	brightest
bad : worst : : cold	coldest
bad : worst : : cool	coolest
bad : worst : : dark	darkest
bad : worst : : easy	easiest
bad : worst : : fast	fastest
bad : worst : : good	best
bad : worst : : great	greatest

Extrinsic Evaluation

- 採用真實的任務來評估詞向量
- 運算較慢
- 較難評估詞向量的表現

詞嵌入

詞嵌入介紹

- One-Hot 編碼可能會讓向量維度變得非常龐大
- 因此：我們需要計算龐大向量的矩陣乘積
- 然而：計算龐大的矩陣只為了取出特定的值
- 詞嵌入就是用來改進One-Hot 編碼的缺點，利用字典的概念來取出特定的矩陣。

Name Entity Recognition (NER)

Name Entity Recognition (NER)

- Google有限公司ORG（英語LANGUAGE：Google LLCORG；中文LANGUAGE：谷歌ORG[7][註 2]）是源自美國GPE的跨國科技公司，為Alphabet IncORG.的子公司，業務範圍涵蓋網際網路廣告、網際網路搜尋、雲端運算等領域，開發並提供大量基於網際網路的產品與服務[9CARDINAL]，其主要利潤來自於AdWordsORG等廣告服務[10CARDINAL][11]。GoogleORG由在史丹佛大學ORG攻讀理工博士的賴利·佩吉PERSON和謝爾蓋·布林PERSON共同建立，因此兩CARDINAL人也被稱為「Google Guys」[12CARDINAL][1CARDINAL3][14CARDINAL]。

https://zh.wikipedia.org/wiki/Google

- Purposes
  - Tracking particular entities in documents
  - Answer of the QA System
  - Slot-Filling

Named Entity Recognition on Word Sequences

- Name Entity Recognition is a classification problem. Classifier will predicts the class that the word belongs to.

➢ Chia-Yi	B-person
➢ Su	I-person
➢ United	B-nation
➢ State	I-nation
➢ Hewlett	B-ORG
➢ Packard	I-ORG
➢ Enterprise	I-ORG
➢ Say	O

BIO-Encoding

Why is NER hard?

- Hard to know if something is an entity
  - 未來學校的教學非常不同

    Is “未來學校” the school name or merely future school?
- Ambiguity of the entity class & Dependence on Context
  - 比爾蓋茲之前是首富 ⟷ 比爾蓋茲公司今年有賺錢
- Hard to know the unknown class
  - To find out more about Zig Ziglar and read features by other Creators Syndicate writers and

    Zig Ziglar ?

What ?

Actually Person

Binary Word Windows Classification

- 自然語言通常非常模糊，所以很難用單一字詞來分類。
- 下面為自我反義的語詞範例：
  - Sanction means both “permit” and “punish”.
  - Seed means both “produce” and “remove”.
- 語詞的意思須根據其上下文來決定

Windows Classification

- 在分類問題上通常會使用整個上下文來做分類而不是採用單一語詞。
- 方法： 將在上下文裡每個語詞的詞向量取平均並相加
- 然而：此方法可能模糊掉詞向量裡所學習到的資訊

Resolution?

Windows Classification: Softmax and Concatenation of Word Vector

- 由於將詞向量相加取平均可能會模糊掉詞向量裡的重要資訊
- 因此：我們會將每個語詞的詞向量做concatenation
- 假設我們想知道”Taiwan”在句子“NKUST in Taiwan has many department.”的命名實體
- 假設視窗大小為2，詞向量的維度為4。

➢ $x_{\text{Taiwan}} = [x_{\text{NKUST}}, x_{\text{in}}, x_{\text{Taiwan}}, x_{\text{has}}, x_{\text{many}}]^T \in R^{5d}$, d = 4.

- Softmax 及 Cross-Entropy的計算方法與之前相同。

預訓練詞向量探討

Derivative of Gradient wrt Words

- $x_{window} = [x_{\text{NKUST}}, x_{\text{in}}, x_{\text{Taiwan}}, x_{\text{has}}, x_{\text{many}}]$
- Let $\nabla_x J = W^T \delta = \delta_{window}$

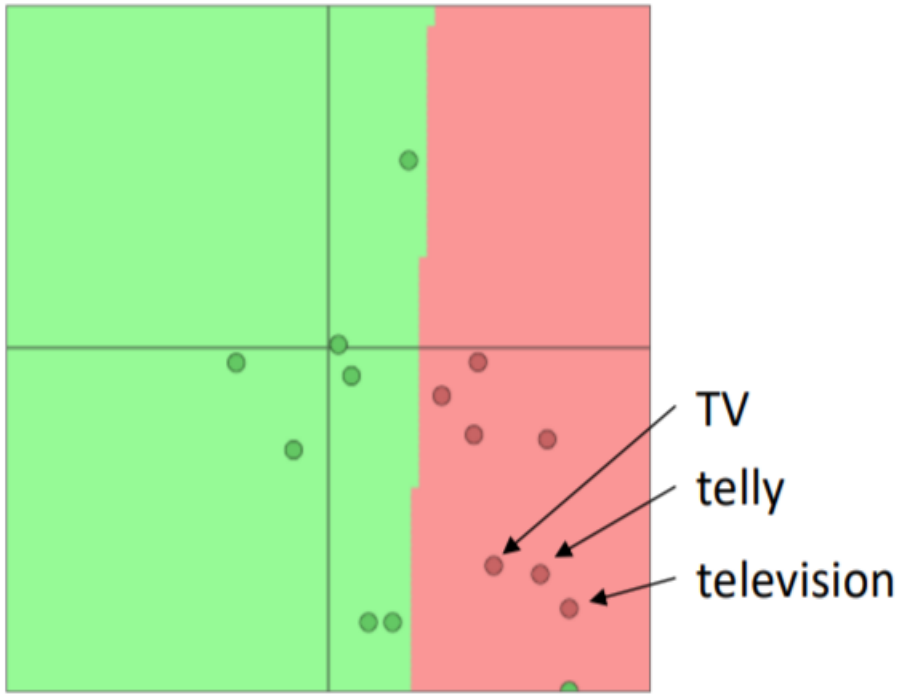
➢ $\delta_{window} = \begin{bmatrix} \nabla x_{\text{NKUST}} \\ \nabla x_{\text{in}} \\ \nabla x_{\text{Taiwan}} \\ \nabla x_{\text{has}} \\ \nabla x_{many} \end{bmatrix} \in R^{5d}$

Updating word vector is basically good.
but

- We connect only windows to the classifier instead of the whole word vector.
- We will update one word vector each time.
- This is called sparse update.

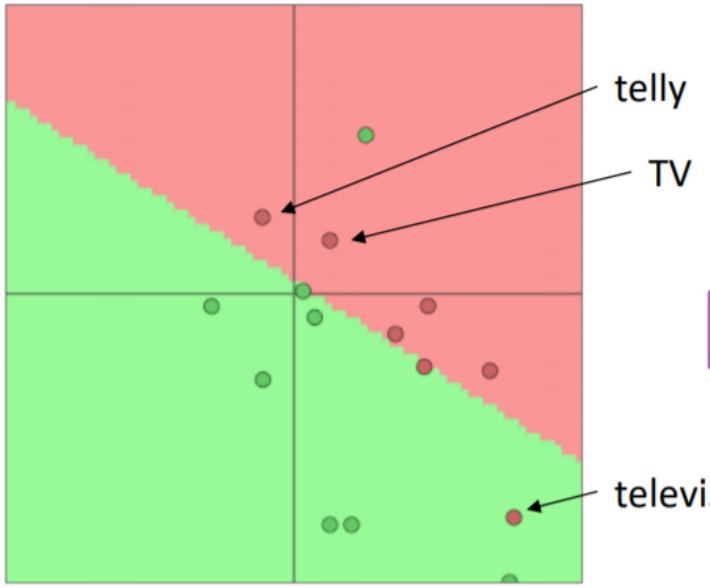
Pitfall while retraining word vector

- 假設我們要分類出電影評論的情感
  - ➢假設“TV” 及 “Telly”為訓練資料集的語詞
  - ➢假設”television”為測試資料集的語詞
  - ➢語訓練資料集如下



What happen when word vector is updated?

- 更新後的模型如下



Only Telly and TV are updated

This can be bad!

Common Questions for Word Vector

- 是否需要採用預訓練的詞向量?
  - ➢是
  - ➢要用幾十億個語詞來訓練詞向量演算法是非常容易的。因此，詞向量演算法還可能擁有不在訓練資料及內的相關知識。
  - ➢如果我們的語料庫非常龐大(幾十億個語詞)，我們可以訓練自己的詞向量演算法。
  - ➢機器翻譯通常不需要採用語訓練詞向量演算法。
- 是否需要更新語訓練詞向量?
  - ➢ 取決於資料大小
  - 如果資料集非常小，就不需更新預訓練詞向量的權重。
  - 如果資料集非常大，更新其權重可能會有較優的結果。

References

- 斎藤康毅, Deep Learning 2: 用Python進行自然語言處理的基礎理論實作.
- Manning et al., CS224n Natural Language Processing with Deep Learning, Stanford University.
- Mikolov et al., Efficient Estimation of Word Representations in Vector Space
- Mikolov et al., Distributed Representation of Words and Phrases and their Compositionality
- Alex Minnaar Word2Vec Tutorial Part II: The Continuous Bag-of-Words Model
- Pennington et al., GloVe: Global Vectors for Word Representation