



Lecture 1. Introduction to Natural Language Processing

陳信希
國立臺灣大學資訊工程學系

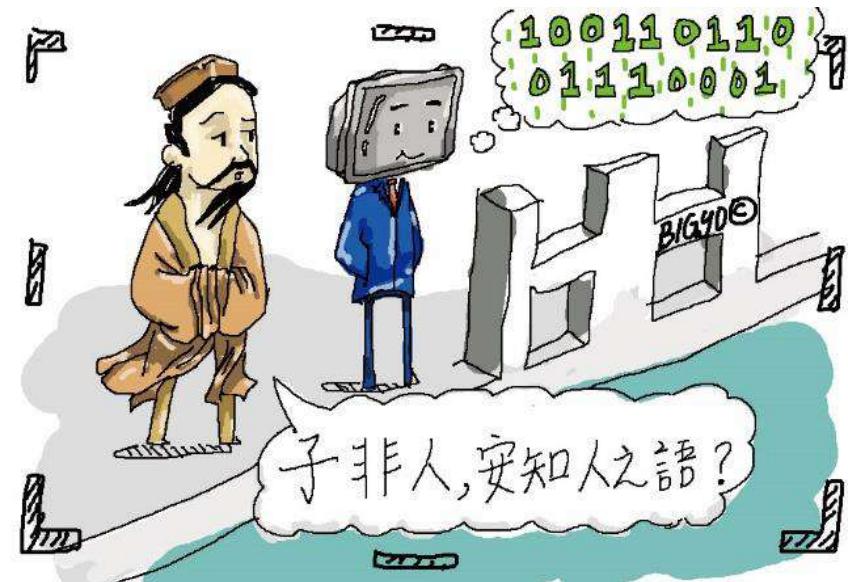
Hsin-Hsi Chen
Department of Computer Science and Information Engineering
National Taiwan University

Outlines

- What is Natural Language Processing?
- Natural Language Processing in Hype Cycle of AI
- Symbolic Representation from NLP perspective
- Distributional Representation
- Distributed Representation
- Learning by Using Both Data and Knowledge
- Resources
- Text/Reference Books
- Agenda
- TA

電腦是否擁有人的智慧？

- 語言文字是人與人之間溝通的主要工具
- 電腦能否聽、說、讀、寫、理解人的語言，為是否擁有人的智慧重要指標



(曾大祐繪)

背景：應用和挑戰

1. 機器翻譯
2. 問答系統
3. 意見探勘

機器翻譯

機器翻譯

- 不同語言間的文本自動翻譯
- 原始語言和目標語言使用詞彙和結構差異
- 原始語言文句分析，目標語言文句生成
- 由早年冷戰時期的情報收集、到近年生活面向的各種需求



生活面向的應用

- 與圖像文字或語音整合的翻譯
- 與地理位置整合的方言翻譯



<https://www.ithome.com.tw/news/93578>



<https://news.tvbs.com.tw/focus/849241>

Long Term Observations on Google Translate

- Taiwan wins gold in woman's 75kg powerlifting in Paralympics

Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raised in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

臺灣在殘奧會贏得女子75公斤級舉重項目金牌。

新華社9月14日在北京報導，中華台北旗幟北京週日在2008年殘奧會終於升起，林資惠贏得女子75公斤級舉重項目金牌。

Google 翻譯 - Windows Internet Explorer

File Edit View Favorites Tools Help

Favorites Suggested Sites Customize Links Personalize

Google 翻譯

說明

Google™

翻譯 BETA

文字和網站 翻譯後的搜尋結果 字典 工具

翻譯文字

原文：

Taiwan wins gold in women's 75 kg powerlifting in Paralympics
Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raised in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

翻譯：英文 » 中文(繁體)

台灣勝金在婦女的75公斤powerlifting在殘奧會
新華社北京9月14日（台北訊）中國台北的旗幟，終於提出了在2008年殘奧會在北京週日，與林姿慧贏得金牌，在女子75公斤級的powerlifting事件。

更好的翻譯建議

翻譯網頁

http://

英文 » 中文(繁體) 翻譯

Google 首頁 - 關於 Google 翻譯

Done CH 舊 Internet 100%

Google 翻譯

https://translate.google.com.tw/?hl=zh-TW#en/zh-TW/Taiwan%20wins%20gold%20in%20woman%E2%80%99s%2075kg%20powerlifting%20

若要在此查看我的最愛，請選取 並選取 ，然後拖曳到 [我的最愛列] 資料夾。或者，從其他瀏覽器匯入。匯入我的最愛

Google

翻譯

關閉即時翻譯

英文 中文 日文 偵測語言

中文(繁體) 英文 中文(簡體)

翻譯

Taiwan wins gold in woman's 75kg powerlifting in Paralympics Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raises in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

台灣在殘奧會女子75公斤級舉重比賽中獲得金牌 - 北京9月14日電 (中國台北) 週日，北京2008年殘奧會在北京舉行，中國台北國旗終於有所提升，林子輝獲得女子75公斤級金牌 舉重活動。

267/5000

Google 翻譯

https://translate.google.com.tw/?hl=zh-TW#en/zh-TW/Taiwan%20wins%20gold%20in%20woman%E2%80%99s%2075kg%20powerlifting%20

若要在此查看我的最愛，請選取 並選取 ，然後拖曳到 [我的最愛列] 資料夾。或者，從其他瀏覽器匯入。匯入我的最愛

Google

翻譯

關閉即時翻譯

英文 中文 日文 偵測語言

中文(繁體) 英文 中文(簡體)

翻譯

Taiwan wins gold in woman's 75kg powerlifting in Paralympics

Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raises in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

台灣在殘奧會女子舉重75公斤級比賽中獲得金牌

(北京時間9月14日) 中國台北國旗在周日的北京2008年殘奧會上終於升起，林子輝在女子75公斤級舉重比賽中獲得金牌。

267/5000

Táiwān zài cán ào huì nǚzī jǔzhòng 75 gōngjīn jí bìsài zhōng huódé jīnpái

(bēijīng shíjiān 9 yuè 14 rì) zhōngguó láibēi guóqí zài zhōu rì de bēijīng 2008 nián cán ào huì shàng zhōngyú shèng qǐ, línzǐ huī zài nǚzī 75 gōngjīn jí jǔzhòng bìsài zhōng huódé jīnpái.

Deep learning boosts Google Translate tool

- Google's online translation service, Google Translate, will soon be using a new algorithm that is entirely based on **deep learning**, the company announced on **27 September, 2016**. The algorithm, which is also described in a paper posted to the preprint server **arXiv**, is the first widely-available computer system for translating languages that relies on the increasingly popular AI technique. Compared to the firm's existing service, the algorithm **reduces errors by around 60%**, Google computer scientists say.

Google 翻譯 BETA

文字和網站 翻譯後的搜尋結果 字典 工具

翻譯文字

原文：

Taiwan wins gold in women's 75 kg powerlifting in Paralympics
Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raised in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

翻譯：英文 » 中文(繁體)

台灣勝金在婦女的75公斤powerlifting在殘奧會
新華社北京9月14日（台北訊）中國台北的旗幟，終於提出了在2008年殘奧會在北京週日，與林姿慧贏得金牌，在女子75公斤級的powerlifting事件。

英文 » 中文(繁體) 翻譯 更好的翻譯建議

(translated by Google Translate on Sept 16, 2008)

Google 翻譯

試試具有自動翻譯功能的最新瀏覽器。 下載 Google Chrome 瀏覽器 閉關

原文是英文 ▾ 翻譯成中文(繁體) ▾ 翻譯

英文 中文 日文 偵測語言

Taiwan wins gold in woman's 75 kg powerlifting in Paralympics
Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raised in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

中文(繁體) 英文 中文(簡體)

台灣勝在殘奧會舉重女子75公斤黃金
新華社北京9月14日（CNA）的中國台北標誌終於抬起日在北京2008年殘奧會，與林慈
濟輝的贏得了金牌，在女子75公斤級舉重事件。

按一下上方的文字，即可編輯及查看其他翻譯。 閉關

(translated by Google Translate on Sept 12, 2012)



試試具有自動翻譯功能的最新瀏覽器。 下載 Google Chrome 瀏覽器 關閉

翻譯

原文是 英文 ▾



翻譯成 中文(繁體) ▾



英文 中文 日文 偵測語言

Taiwan wins gold in woman's 75 kg powerlifting in Paralympics
Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raised in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

中文(繁體) 英文 中文(簡體)

台灣勝在殘奧會舉重女子75公斤黃金
新華社北京9月14日(CNA)的中國台北標誌終於抬升日在北京2008年殘奧會，與林慈濟輝的贏得了金牌，在女子75公斤級舉重事件。



新! 按一下上方的文字，即可編輯及查看其他翻譯。 關閉

(2012.9.12)



試試具有自動翻譯功能的最新瀏覽器。 下載 Google Chrome 瀏覽器 關閉

翻譯



中文 英文 日文 偵測語言 ▾



英文 中文(簡體) 中文(繁體) ▾



Taiwan wins gold in woman's 75 kg powerlifting in Paralympics
Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raised in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

台灣勝金在女子75公斤級舉重殘奧會
北京，9月14日(CNA)的中國台北標誌在2008年殘奧會在北京終於抬升週日，與林梓輝贏得金牌的女子75公斤級舉重的事件。



(2014.2.18)

英文 中文 日文 偵測語言 ▾



中文(繁體) 中文(簡體) 英文 ▾



Taiwan wins gold in woman's 75 kg powerlifting in Paralympics
Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raised in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

台灣贏得金牌的女子75公斤舉重在殘奧會
北京，9月14日(CNA)的中國台北國旗終於提出在2008年殘奧會在北京週日，林姿輝贏得了金牌，在女子75公斤級舉重賽事。



不對嗎？

(2015.2.25)

英文 中文 日文 偵測語言 ▾

中文(簡體) 英文 中文(繁體) ▾ 翻譯

Taiwan wins gold in woman's 75 kg powerlifting in Paralympics
Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raised in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.



台灣勝金在殘奧會女子75公斤舉重
北京9月14日 (CNA) 的中華台北國旗終於提出在2008年殘奧會在北京週日與林姿輝贏得舉重賽事的女子75公斤級金牌。

提出修改建議

(2016.2.24)

Taiwan wins gold in woman's 75 kg powerlifting in Paralympics
Beijing, Sept. 14 (CAN) The Chinese Taipei flag was finally raised in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

台灣在女子75公斤舉重在殘奧會贏得金子
北京9月14日 (北京時間) 中國台北旗帜在北京星期天的残奥会上终于被提起，林子辉在女子75公斤举重赛事中获得金牌。

提出修改建議

(2016.12.8)

Taiwan wins gold in woman's 75kg powerlifting in Paralympics
Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raises in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.

台灣勝金在殘奧會女子75公斤級舉重
北京9月14日 (CNA) 的中華台北國旗終於引發了2008年殘奧會在北京週日與林姿輝贏得舉重賽事的女子75公斤級金牌。

(2017.2.22)

Taiwan wins gold in woman's 75kg powerlifting in Paralympics
Beijing, Sept. 14 (CNA) The Chinese Taipei flag was finally raises in the 2008 Paralympic Games in Beijing Sunday, with Lin Tzu-hui winning a gold medal in the women's 75 kg category of powerlifting event.



267/5000

台灣在殘奧會女子舉重75公斤級比賽中獲得金牌

(北京時間9月14日) 中國台北國旗在周日的北京2008年殘奧會上終於升起 · 林子輝在女子75公斤級舉重比賽中獲得金牌。

提出修改建議

Taiwan zài cán ào huì nǚzǐ jǔzhòng 75 gōngjīn jí bǐsài zhōng huòdé jīnpái

(2018.2.25)

Google Translate 十年觀察

台灣勝金在婦女的75公斤powerlifting在殘奧會

新華社北京9月14日(台北訊)中國台北的旗幟，終於提出了在2008年殘奧會在北京週日，與林姿慧贏得金牌，在女子75公斤級的powerlifting事件。
(2008.9.16)

台灣勝在殘奧會舉重女子75公斤黃金

新華社北京9月14日(CNA)的中國台北標誌終於抬起日在北京2008年殘奧會，與林慈清輝的贏得了金牌，在女子75公斤級舉重事件。
(2012.9.12)

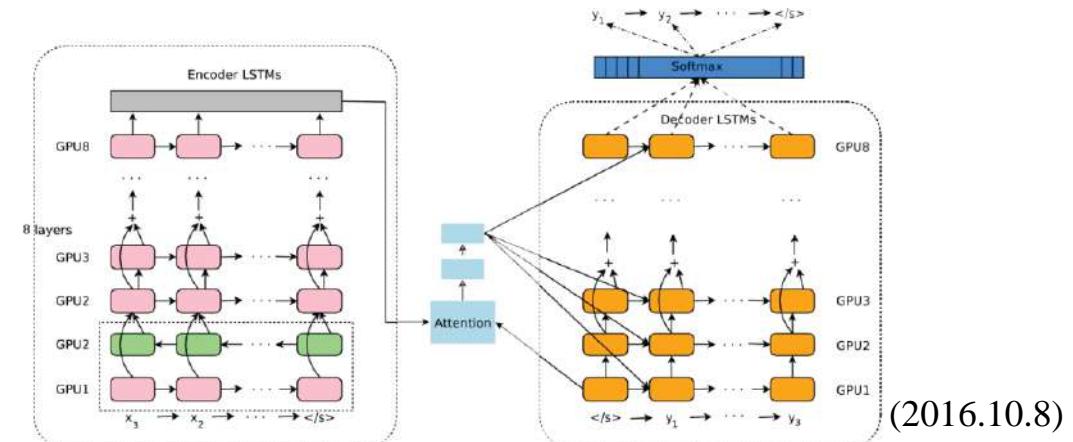
台灣勝金在女子75公斤級舉重殘奧會

北京，9月14日(CNA)的中國台北標誌在2008年殘奧會在北京終於抬起週日，與林梓輝贏得金牌的女子75公斤級舉重事件。(2014.2.18)

台灣贏得金牌的女子75公斤舉重在殘奧會

北京，9月14日 (CNA) 的中國台北國旗終於提出在2008年殘奧會在北京週日，林姿輝贏得了金牌，在女子75公斤級舉重賽事。(2015.2.25)

Google Neural Machine Translation System



台灣勝金在殘奧會女子75公斤舉重

北京9月14日 (CNA) 的中華台北國旗終於提出在2008年殘奧會在北京週日與林姿輝贏得舉重賽事的女子75公斤級金牌。 (2017.2.24)

台灣在殘奧會女子舉重75公斤級比賽中獲得金牌

(北京時間9月14日) 中國台北國旗在周日的北京2008年殘奧會上終於升起，林子輝在女子75公斤級舉重比賽中獲得金牌。 (2018.2.25)

詞彙面向

- raise (升起) → 提出 (2008) → 抬起 (2012) → 抬起 (2014) → 提出 (2015) → 提出 (2016) → 引發 (2017) → 升起 (2018)
- woman (女子) → 婦女 (2008) → 女子 (2012) → 女子 (2014) → 女子 (2015) → 女子 (2016) → 女子 (2017) → 女子 (2018)
- event (項目) → 事件 (2008) → 事件 (2012) → 事件 (2014) → 賽事 (2015) → 賽事 (2016) → 賽事 (2017) → 比賽 (2018)
- Gold (金牌) → 金/金牌 (2008) → 黃金/金牌 (2012) → 金/金牌 (2014) → 金牌/金牌 (2015) → 金/金牌 (2016) → 金牌/金牌 (2018)
- powerlifting (舉重) → powerlifting (2008) → 舉重 (2012) → 舉重 (2014) → 舉重 (2015) → 舉重 (2016) → 舉重 (2017) → 舉重 (2018)
- Lin Tzu-hui (林資惠) → 林姿慧 (2008) → 林慈清輝 (2012) → 林梓輝 (2014) → 林姿輝 (2015) → 林姿輝 (2016) → 林姿輝 (2017) → 林子輝 (2018)

結構面向

- Taiwan wins gold **in woman's 75 kg powerlifting** **in Paralympics**
- 臺灣**在殘奧會贏得女子75公斤級舉重項目**金牌
- 台灣勝金在婦女的75公斤powerlifting**在殘奧會** (2008)
- 台灣勝**在殘奧會舉重女子75公斤**黃金 (2012)
- 台灣勝金**在女子75公斤級舉重殘奧會** (2014)
- 台灣贏得**金牌的女子75公斤舉重在殘奧會** (2015)
- 台灣勝**金在殘奧會女子75公斤舉重** (2016)
- 台灣勝**金在殘奧會女子75公斤級舉重** (2017)
- 台灣**在殘奧會女子舉重75公斤級比賽中獲得金牌** (2018)

機器翻譯
是否已經達到人類翻譯的境界？

英文病例中文翻譯結果

• 首席投訴

吐血於1/22。

• 歷史簡介

這位70歲的男性有1.高血壓，2。糖尿病，3.慢性腎臟病。他定期在我們的OPD隨訪。他的長期處方包括Tapal 1 tab QD。根據他妻子的陳述，他感到從1/22早上開始噁心，然後他突然暈厥。許多全身抽搐時1/23嘔吐血。然後他被救護車送到了我們的急診室。

在我們的急診室，實驗室數據顯示血紅蛋白水平低（7.5 mg / dl）和正常範圍的心肌酶和正常範圍的PT，一鍵通對疑似上肢患者進行輸血和Pantoloc胃腸道出血。內鏡檢查結果為1/23，結果為0.4cm A1十二指腸潰瘍和胃糜爛。Bosmin注射和加熱器內窺鏡檢查過程中完成探頭檢查。在印像中十二指腸潰瘍出血，他被送進我們病房接受進一步治療管理。

• 療程和治療

入院後，我們繼續靜脈輸液和PPI。我們切換了自2010年1月26日起，PPI為口服形式。隨後血紅蛋白靜止（1/27血紅蛋白：10.3）。他於2010/01/28出院，進一步進行了GI OPD隨訪。

- 主訴

英文病例中文翻譯結果

1月22日嘔血。

- 歷史簡介

這名70歲的男性有1.高血壓，2。糖尿病，3。慢性腎病。他經常跟進我們的OPD。他的長期處方包括Tapal 1 tab QD。根據他妻子的陳述，他從1月22日早上就感到噁心，然後他突然暈厥。很多血液在1/23時嘔吐，伴有全身性驚厥。然後他被救護車送到我們的急診室。

在我們的急診室，實驗室數據顯示低水平血紅蛋白（7.5 mg / dl）和正常範圍的心肌酶和正常範圍的PT，PTT。疑上消化道出血給予輸血和Pantoloc。內窺鏡檢查在1/23進行，結果為0.4cm A1十二指腸潰瘍和胃糜爛。Bosmin注射和加熱器探針在內窺鏡檢查過程中完成。在十二指腸潰瘍出血的印象下，他被送進我們的病房接受進一步治療。

- 課程和治療

入院後，我們繼續靜脈輸液和PPI。自2010/01/26起，我們將PPI轉換為口頭形式。血紅蛋白靜止後（1 / 27Hb：10.3）。他於2010/01/28出院，隨後進一步接受了胃腸道檢查。 (系統：Google Translate，時間：2018/9/15)

Chief Complaint

Hematemesis on 1/22.

Brief History

This 70-year-old male has history of 1.Hypertension, 2.Diabetes mellitus, 3.Chronic kidney disease. He regularly followed up at our OPD. His long term prescription included Tapal 1 tab QD. according to his wife's statement, he felt nausea since 1/22 morning, and then he had sudden onset of syncope. A lot of blood was vomited on 1/23 accompanied with general convulsion. Then he was sent to our emergency department by Ambulance.

In our emergency department, laboratory data revealed low level hemoglobin (7.5 mg/dl) and normal range of cardiac enzyme and normal range of PT、PTT. Blood transfusion and Pantoloc were given for suspected upper gastrointestinal bleeding. Endoscopy was performed on 1/23 with results of a 0.4cm A1 duodenal ulcer and a gastric erosion. Bosmin injection and heater probe were done during endoscopy procedure. Under the impression of duodenal ulcer with bleeding, he was admitted to our ward for further management.

Course and Treatment

After admission, we continued intravenous fluid supply and PPI. We switched

PPI to oral form since 2010/01/26. Following hemoglobin was stationary (1/27

Hb:10.3). He was discharged on 2010/01/28 with further GI OPD follow up.

X

首席投訴

吐血於1/22。

歷史簡介

這位70歲的男性有1.高血壓，2。糖尿病，3。慢性腎臟病的病史。他定期在我們的OPD隨訪。他的長期處方包括Tapal 1 tab QD。根據妻子的陳述，自1/22上午起，他感到噁心，然後突然暈厥。全身抽搐後1/23時大量嘔吐。然後他被救護車送到了我們的急診室。

在我們的急診科，實驗室數據顯示血紅蛋白水平低 (7.5 mg / dl) ，心臟酶正常範圍，PT，PTT正常範圍。懷疑有上消化道出血的患者進行輸血和泛托洛因治療。內鏡檢查為1/23，結果為0.4cm A1十二指腸潰瘍和胃糜爛。在內窺鏡檢查過程中進行了Bosmin注射和加熱器探針。在十二指腸潰瘍出血的印象下，他被送進我們病房進行進一步治療。

療程和治療

入院後，我們繼續靜脈輸液和PPI。我們切換了自2010/01/26以來，PPI改為口服形式。隨後血紅蛋白靜止 (1/27血紅蛋白：10.3)。他於2010/01/28出院，進一步進行了GI OPD隨訪。

Shǒuxí tóusù

tǔxiě yú 1/22.

Lìshǐ jiǎnjiè

[Show more](#)

主訴

1月22日嘔血。

歷史簡介

這名70歲的男性有1.高血壓，2。糖尿病，3。慢性腎病。他經常跟進我們的OPD。他的長期處方包括Tapal 1 tab QD。根據他妻子的陳述，他從1月22日早上就感到噁心，然後他突然暈厥。很多血液在1/23時嘔吐，伴有全身性驚厥。然後他被救護車送到我們的急診室。

在我們的急診室，實驗室數據顯示低水平血紅蛋白（7.5 mg / dl）和正常範圍的心肌酶和正常範圍的PT，PTT。懷疑上消化道出血給予輸血和Pantoloc。內窺鏡檢查在1/23進行，結果為0.4cm A1十二指腸潰瘍和胃糜爛。Bosmin注射和加熱器探針在內窺鏡檢查過程中完成。在十二指腸潰瘍出血的印像下，他被送進我們的病房接受進一步治療。

課程和治療

入院後，我們繼續靜脈輸液和PPI。自2010/01/26起，我們將PPI轉換為口頭形式。血紅蛋白靜止（1 / 27Hb : 10.3）。他於2010/01/28出院，隨後進一步接受了胃腸道檢查。

英文病例中文翻譯結果

（系統：Google Translate，時間：2018/9/15）

首席投訴

吐血於1/22。

歷史簡介

這位70歲的男性有1.高血壓，2。糖尿病，3。慢性腎臟病的病史。他定期在我們的OPD隨訪。他的長期處方包括Tapal 1 tab QD。根據妻子的陳述，自1/22上午起，他感到噁心，然後突然暈厥。全身抽搐後1/23時大量嘔吐。然後他被救護車送到了我們的急診室。在我們的急診科，實驗室數據顯示血紅蛋白水平低（7.5 mg / dl），心臟酶的正常範圍以及PT，PTT的正常範圍。懷疑有上消化道出血的患者進行輸血和泛托洛因治療。內鏡檢查為1/23，結果為0.4cm A1十二指腸潰瘍和胃糜爛。在內窺鏡檢查過程中進行了Bosmin注射和加熱器探針。在十二指腸潰瘍出血的印像下，他被送進我們病房進行進一步治療。

療程和治療

入院後，我們繼續靜脈輸液和PPI。自2010年1月26日起，我們將PPI更改為口頭形式。隨後血紅蛋白靜止（1 / 27Hb : 10.3）。他於2010/01/28出院，進一步進行了GI OPD隨訪。

（系統：Google Translate，時間：2019/11/19）

首席投訴

吐血於1/22。

歷史簡介

這位70歲的男性有1.高血壓，2。糖尿病，3。慢性腎臟病的病史。他定期在我們的OPD隨訪。他的長期處方包括Tapal 1 tab QD。根據妻子的陳述，自1/22上午起，他感到噁心，然後突然暈厥。全身抽搐後1/23時大量嘔吐。然後他被救護車送到了我們的急診室。在我們的急診科，實驗室數據顯示血紅蛋白水平低（7.5 mg / dl），心臟酶的正常範圍以及PT，PTT的正常範圍。懷疑有上消化道出血的患者進行輸血和泛托洛因治療。內鏡檢查為1/23，結果為0.4cm A1十二指腸潰瘍和胃糜爛。在內窺鏡檢查過程中進行了Bosmin注射和加熱器探針。在十二指腸潰瘍出血的印像下，他被送進我們病房進行進一步治療。

療程和治療

入院後，我們繼續靜脈輸液和PPI。自2010年1月26日起，我們將PPI更改為口服形式。隨後血紅蛋白靜止（1 / 27Hb : 10.3）。他於2010/01/28出院，進一步進行了GI OPD隨訪。

（系統：Google Translate，時間：2019/11/19）

首席投訴

吐血於1/22。

歷史簡介

這位70歲的男性有1.高血壓，2。糖尿病，3。慢性腎臟病的病史。他定期在我們的OPD隨訪。他的長期處方包括Tapal 1 tab QD。根據妻子的陳述，自1/22上午起，他感到噁心，然後突然暈厥。全身抽搐後1/23時大量嘔吐。然後他被救護車送到了我們的急診室。

在我們的急診科，實驗室數據顯示血紅蛋白水平低（7.5 mg / dl），心臟酶正常範圍，PT，PTT正常範圍。懷疑有上消化道出血的患者進行輸血和泛托洛因治療。內鏡檢查為1/23，結果為0.4cm A1十二指腸潰瘍和胃糜爛。在內窺鏡檢查過程中進行了Bosmin注射和加熱器探針。在十二指腸潰瘍出血的印像下，他被送進我們病房進行進一步治療。

療程和治療

入院後，我們繼續靜脈輸液和PPI。我們切換了自2010/01/26以來，PPI改為口服形式。隨後血紅蛋白靜止（1/27血紅蛋白 : 10.3）。他於2010/01/28出院，進一步進行了GI OPD隨訪。

（系統：Google Translate，時間：2020/3/2）

英文病例

- History
- Past History:
 - 1.DM:denied,HTN:denied
 - 2.OP hx:denied major operation or trauma hx in the past.
 - 3.other systemic disease:denied
 - 4.Asthma history:denied
- Personal History:
 - 1.food/drug allergy hx:denied
 - 2.denied alcohol drinking;denied smoking
 - 3.trarel history in recent 3 months denied
 - 4.Uccocation:retire.
 - 5.social relation :active
 - 6.race閩南人
- 診斷:
 - 1.ARF
 - 2.Hyperkalemia
 - 3.Abnomal tiver function.

翻譯結果比較

人工翻譯

- 過去病史:

1. 沒有糖尿病,高血壓
2. 對於過去沒有手術和創傷的病史
3. 沒有其他疾病
4. 沒有氣喘

- 個人病史:

1. 對食物藥物沒過敏
2. 沒有喝酒沒有抽菸
3. 最近3個月沒有旅行
4. 工作狀況是退休了
5. 社交的關係是主動的
6. 種族閩南人

機器翻譯 (Google Translate, 2018.9.15)

- 過去的歷史:

- 1.DM:denied,HTN:denied
- 2.OP hx：過去否認主要手術或創傷。
- 3.其他全身性疾病：否認
- 4.Asthma 史：否認

- 個人歷史:

- 1.食物/藥物過敏hx：否認
- 2.拒絕飲酒;拒絕吸煙
- 3.trarel歷史近3個月否認
- 4.Uccipation：退休。
- 5.社會關係：積極的
- 6.race閩南人

機器翻譯 (Google Translate, 2019.11.16)

- 過去的歷史：

- 1.DM：被拒絕， HTN：被拒絕
- 2.OP hx：過去拒絕進行大手術或創傷。
- 3.其他全身疾病：被拒絕
- 4.哮喘病史：被拒絕

- 個人經歷：

- 1.食物/藥物過敏hs：拒絕
- 2.拒絕飲酒；拒絕吸煙
- 3.最近3個月的trarel歷史被拒絕
- 4.職業：退休。
- 5.社會關係：活躍
- 6.種族閩南人

翻譯比較

- 診斷：

1. 急性腎衰竭
2. 高血鉀
3. 不正常的肝功能

- 診斷：

- 1.ARF
- 2.Hyperkalemia
- 3.Abnormal liver function

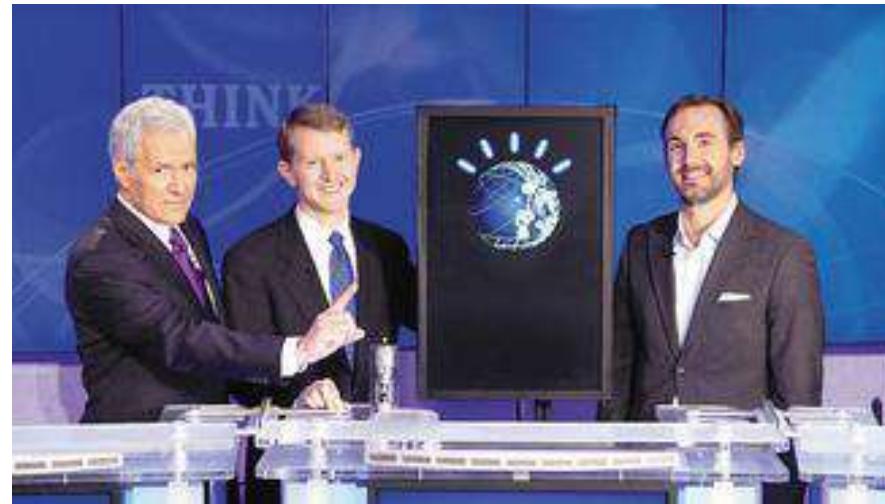
- 診斷：

- 1.ARF
- 2.高鉀血症
- 3.異常功能。

問答系統

人與電腦機智問答比賽 (2011年2月14-16日)

- DeepQA超級電腦(華生，Watson)、簡寧斯(Ken Jennings)、洛特(Brad Rutter)



(美聯社)

- 第一名(華生)：一百萬美元，第二名(簡寧斯)：30萬美元，第三名(洛特)：
20萬元

經典問答題

- Q：指出一美國都市，此城市最大機場以二戰英雄命名，第二大機場以戰役命名
- A：芝加哥（華生：多倫多）
- Q：老牌餅乾Oreo何時首次推出？
- A：1910年代（簡寧斯：20年代，華生：1920年代）

問答系統挑戰

- 分析問題，找出在問甚麼
- 分析內容，擷取正確答案
- 計算支持或反駁資訊的信心度
- 自然語言處理、資訊檢索、機器學習、知識表示和推理、及大規模平行計算等
- 背後的知識庫：結構化/半結構化/非結構化

問題

文件檢索

答案選擇

諾貝爾獎（瑞典語：Nobelpriset，挪威語：Nobelprisen），是根據瑞典化學家阿佛烈·諾貝爾的遺囑於1901年開始頒發的獎項。諾貝爾獎分設物理、化學、生理學或醫學、文學、和平和經濟學六個獎項（經濟學獎於1968由瑞典中央銀行增設，全稱「瑞典銀行紀念諾貝爾經濟科學獎」，通稱「諾貝爾經濟學獎」）。諾貝爾獎普遍被認為是所頒獎的領域內最重要的獎項。

誰創立諾貝爾獎

問甚麼：人名

阿佛烈·伯恩哈德·諾貝爾（瑞典語：Alfred Bernhard Nobel，1833年10月21日—1896年12月10日）是瑞典化學家、工程師、發明家、軍工裝備製造商和矽藻土炸藥的發明者。他曾擁有Bofors軍工廠，主要生產軍火；還曾擁有一座鋼鐵廠。在他的遺囑中，他利用他的巨大財富創立了諾貝爾獎，各種諾貝爾獎項均以他的名字命名。

諾貝爾獎（瑞典語：Nobelpriset，挪威語：Nobelprisen）是一項由瑞典皇家科學院、瑞典學院、卡羅琳學院和挪威諾貝爾委員會頒發給對化學、物理、文學、和平和生理及醫學這五方面有著傑出貢獻的人士或組織的獎項[1]。諾貝爾獎是根據阿佛烈·諾貝爾在1895年的遺囑而設立的，並由諾貝爾基金會管理阿佛烈·諾貝爾的遺產及諾貝爾獎的頒發。

代名詞
指涉

在他的遺囑中，他利用他的巨大財富創立了諾貝爾獎，各種諾貝爾獎項均以他的名字命名。

諾貝爾獎是根據阿佛烈·諾貝爾在1895年的遺囑而設立的，並由諾貝爾基金會管理阿佛烈·諾貝爾的遺產及諾貝爾獎的頒發。

語音問答/智慧音箱



INTRODUCING
amazon echo

Always ready, connected,
and fast. **Just ask.**



意見探勘

文本分析

- 對新聞、科技論文、電子郵件、網頁、部落格貼文、微網誌、病例等不同類型來源的數位資料，進行文件探勘

- 市場產品資訊
- 政治意見追蹤
- 社群網路分析
- 熱門議題分析



The screenshot shows the homepage of the journal "Science". The header reads "Science The World's Leading Journal of Original Scientific Research, Global News, and Commentary". Below the header, there are links for "Science Home", "Current Issue", "Previous Issues", "Science Express", "Science Products", "My Science", and "About the Journal". The main content area displays an article titled "China's Publication Bazaar" by Marla Hydeman. The article summary states: "Science has exposed a thriving academic black market in China involving shady agencies, corrupt scientists, and compromised editors—many of them operating in plain view. The commodity: papers in journals indexed by Thomson Reuters' Science Citation Index, Thomson Reuters' Social Sciences Citation Index, and Elsevier's Engineering Index." There are also links for "Article Views", "Summary", "Full Text", "Full Text (PDF)", "Podcast Interview", "Article Tools", "Save to My Folders", "Download Citation", and "Alert Me When Article Is".

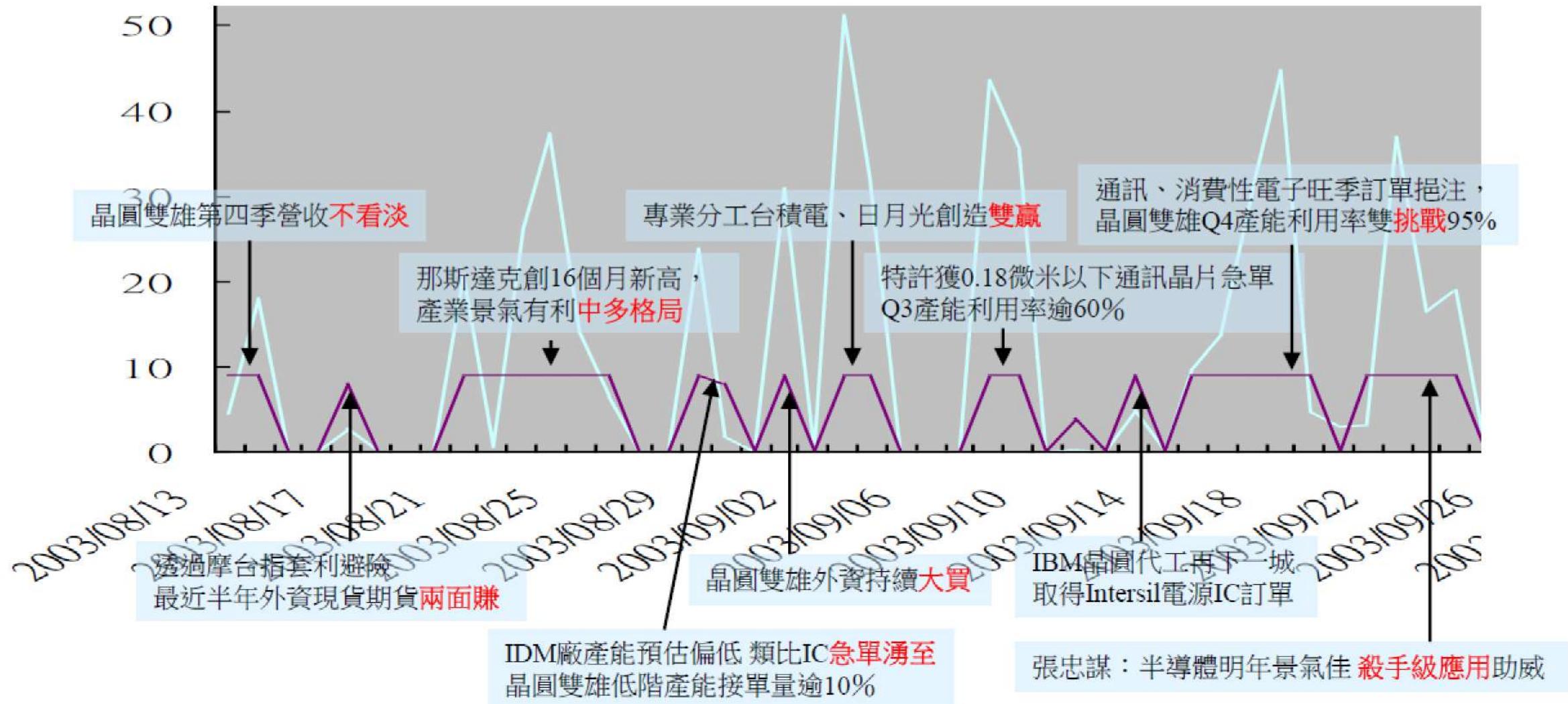


Instagram

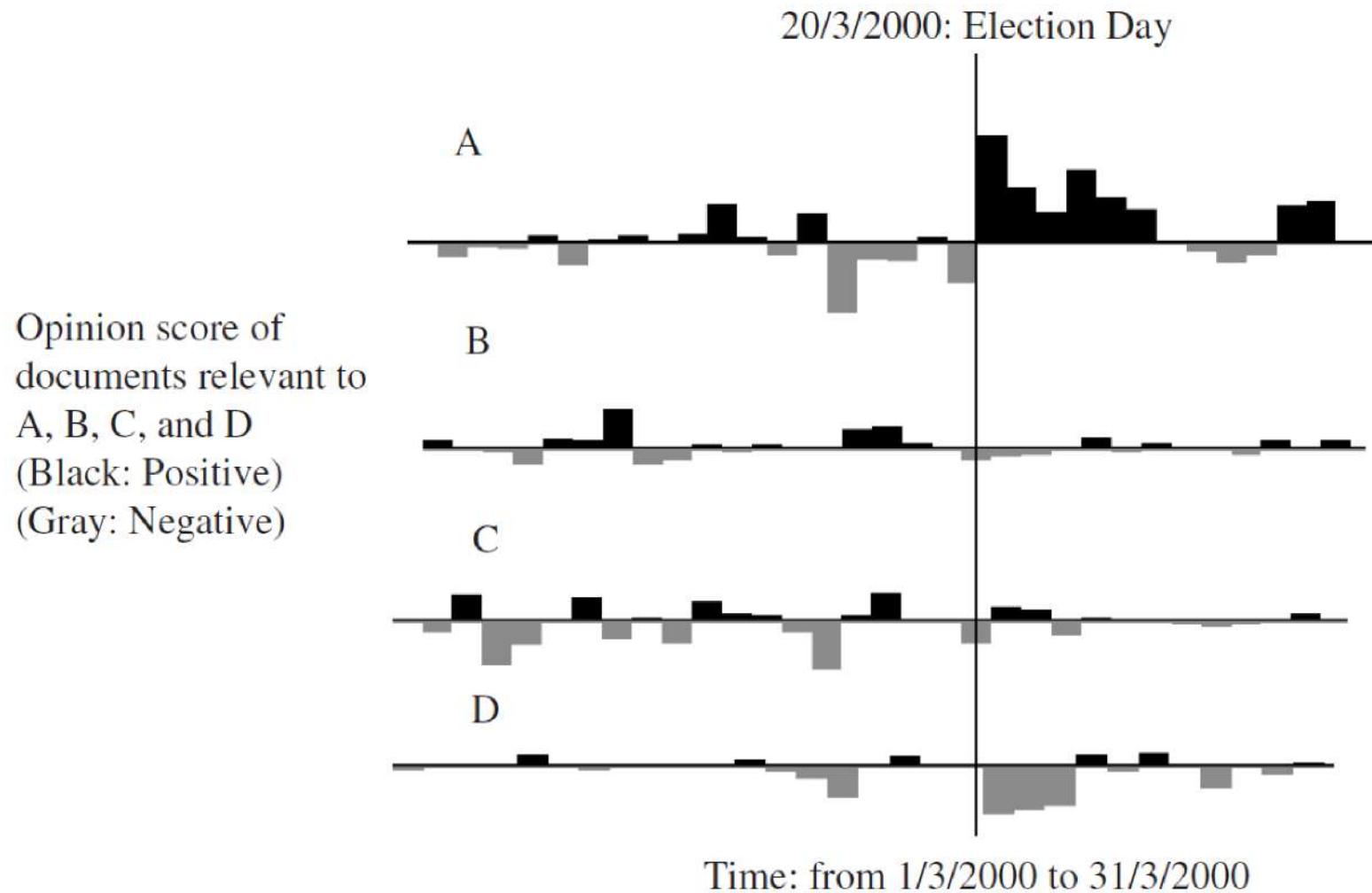
出遊找旅館：旅館評論

狀元樓是南京老牌酒店了，在夫子廟入口的地方，遍布我喜歡的小吃店。客房格局挺古老的，面積不大，不過景觀很好，可以看見秦淮河。服務態度很好，會用力的幫着推轉門（就不能裝個自動的嘛）。洗澡水很舒服。

市場分析：標的追蹤



總統大選：多標的輿情分析



機器人和人辯論賽

- 規則

- 4分鐘開場、4分鐘反駁、和2分鐘總結
- 記者和IBM員工等40名現場觀眾投票

- 議題

- 政府應該增加太空探索的費用
 - Project Debater 持正方觀點，人類辯手持反方觀點
 - Project Debater 比 Dan Zafrir 多獲得 9 票，贏得比賽
- 遠端醫療應該在醫療占更大比例
 - Project Debater 持正方觀點，人類辯手反方觀點
- 結果一敗一勝

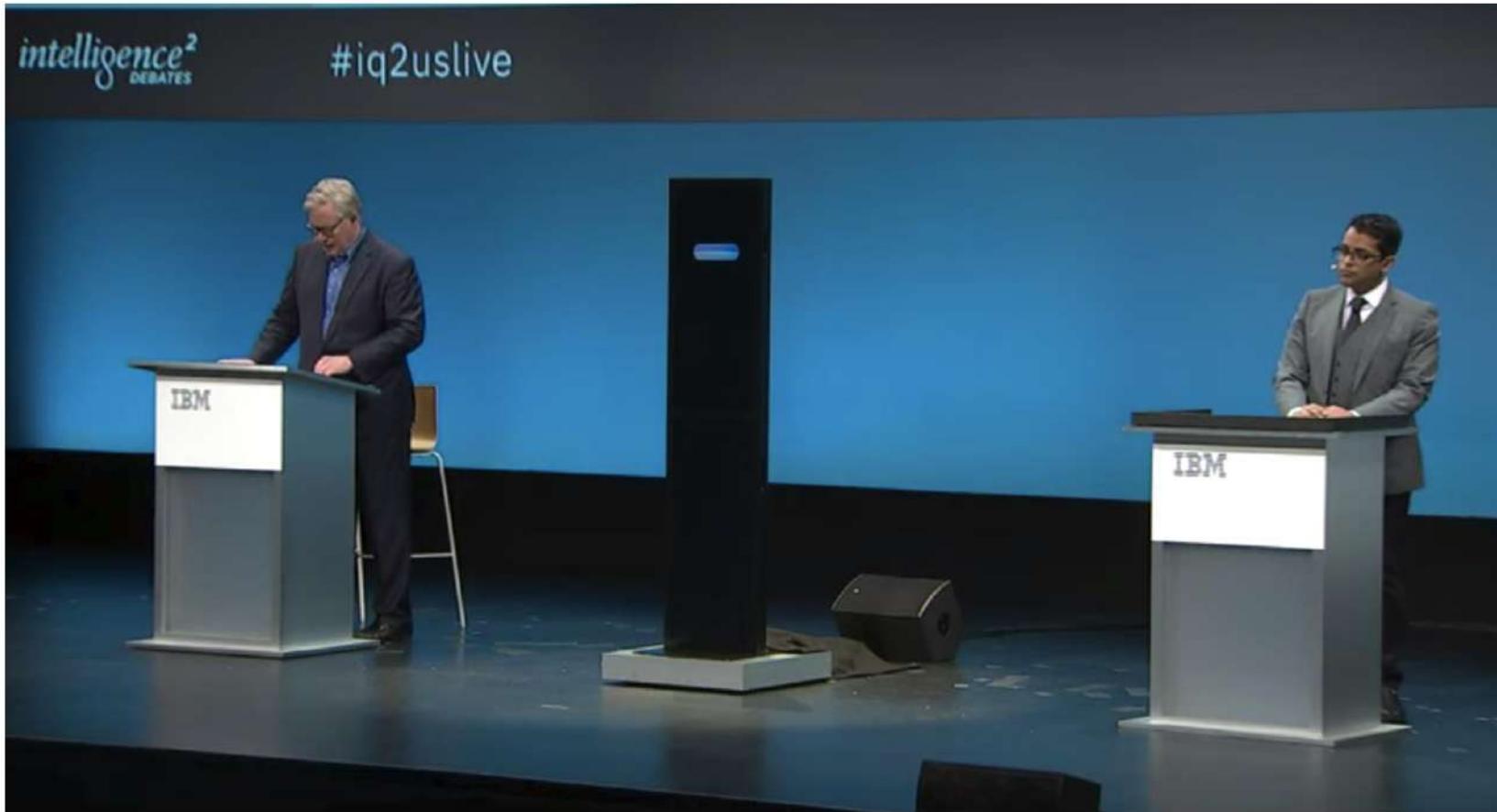


以色列國際辯論協會主席 Dan Zafrir



2016 年以色列國家辯論冠軍 Noa Ovadia

Project Debater 辭論

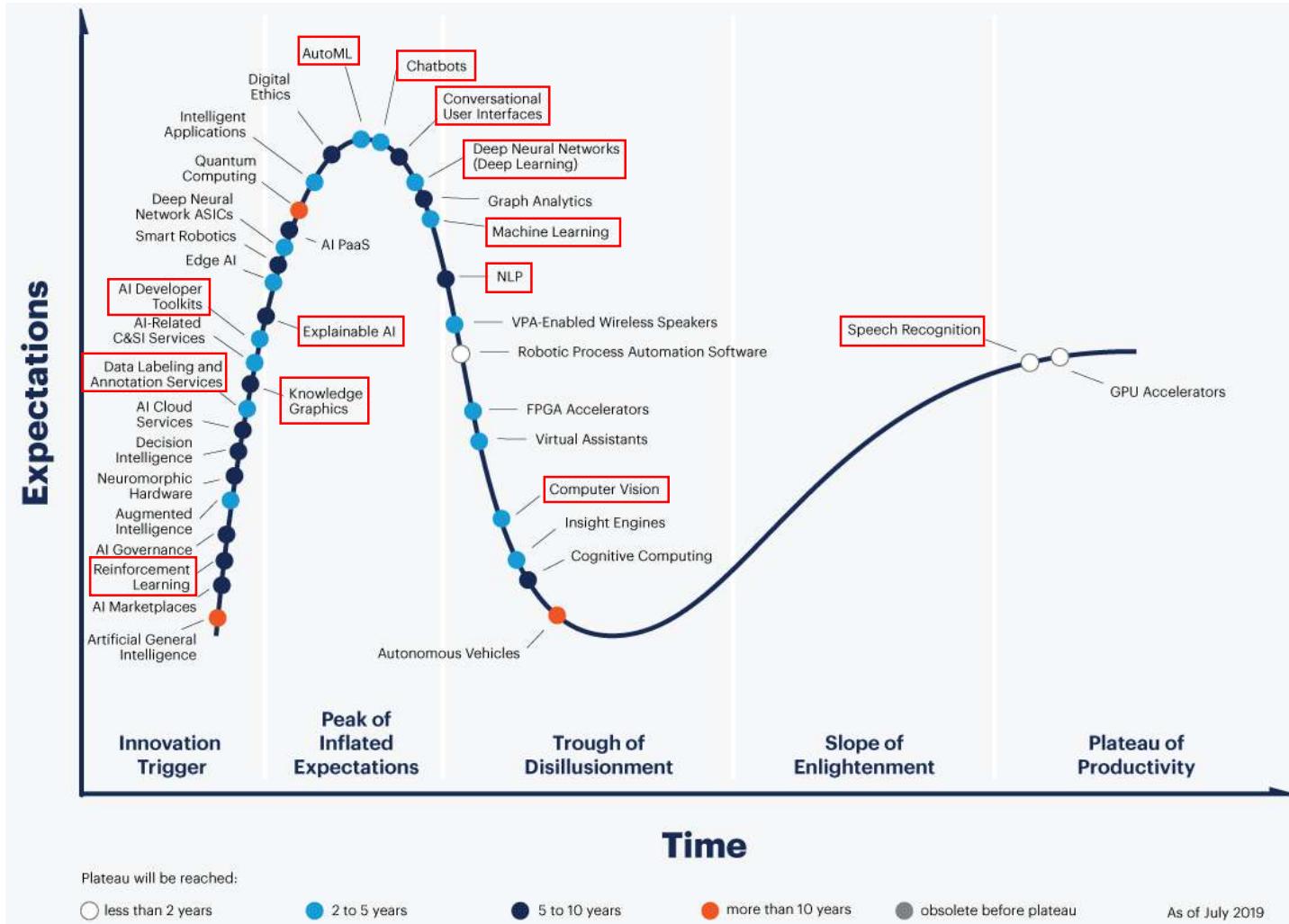


Project Debater 辭論。（左為美國知名的辯論節目智慧廣場 Intelligence Squared U.S. 主持人 John Donvan 主持，中為 Project Debater，右為國際專業辯手 Harish Natarajan）2019/02/12

1:https://buzzorange.com/techorange/2019/02/12/project-debater-ibm/?fbclid=IwAR02FEXWAI0TAPA0t9ZqW88kze6HmM1HTuLv_4ZFxFkJzRJTvUmLijmRdKg
2:<https://www.youtube.com/watch?v=m3u-1yttrVw&t=5s>

人工智慧技術成熟度曲線 (2019)

(Hype Cycle for Artificial Intelligence, 2019)



(科技誕生的促動期)

(泡沫化的底谷期)

(實質生產的高原期)

(過高期望的峰值)

(穩步爬升的光明期)

資料來源：2019 Gartner

The Hype Cycle (技術成熟度曲線)

- Technology Trigger (科技誕生的促動期)
 - 媒體大肆報導，產品的知名度無所不在。隨著科技缺點、問題、限制出現，失敗案例大於成功案例。
- Peak of Inflated Expectations (過高期望的峰值)
 - 公眾過分關注演繹出一系列成功的故事。對於失敗，有些公司採取了補救措施，而大部分卻無動於衷。
- Trough of Disillusionment (泡沫化的底谷期)
 - 經過扎實的試驗而存活的科技，適用範圍及限制有客觀和實際的了解，成功且存活的經營模式逐漸成長。
- Slope of Enlightenment (穩步爬升的光明期)
 - 新科技的誕生，在市面上受到主要媒體與業界高度的注目。
- Plateau of Productivity (實質生產的高原期)
 - 科技產生的利益與潛力被市場實際接受，支援經營模式的工具和方法論經過數代的演進，進入非常成熟的階段。

Hype Cycle for Artificial Intelligence, 2019

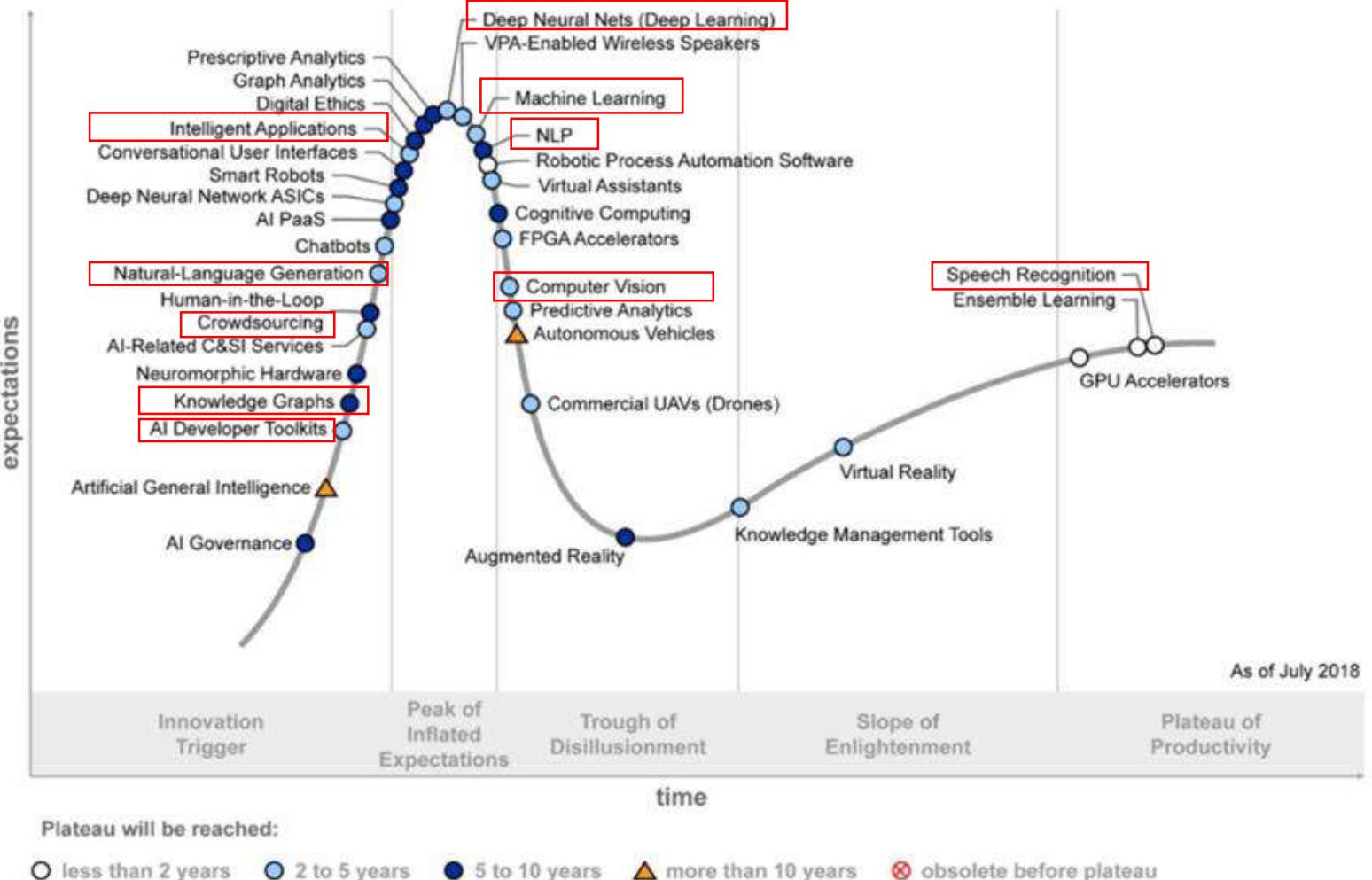
- On the Rise 科技誕生的促動期
 - Artificial General Intelligence
 - AI Marketplaces
 - Reinforcement Learning
 - AI Governance
 - Augmented Intelligence
 - Neuromorphic Hardware
 - Decision Intelligence
 - AI Cloud Services
 - Data Labeling and Annotation Services
 - Knowledge Graphs
 - AI-Related C&SI Services
 - AI Developer Toolkits
 - Explainable AI
- At the Peak 過高期望的峰值
 - Edge AI
 - Smart Robots
 - Deep Neural Network ASICs
 - AI PaaS
 - Quantum Computing
 - Intelligent Applications
 - Digital Ethics
 - AutoML
 - Chatbots
 - Conversational User Interfaces
 - Deep Neural Networks (Deep Learning)
 - Graph Analytics
 - Machine Learning
 - NLP

Hype Cycle for Artificial Intelligence, 2019

- Sliding Into the Trough 泡沫化的底谷期
 - VPA-Enabled Wireless Speakers
 - Robotic Process Automation Software
 - FPGA Accelerators
 - Virtual Assistants
 - Computer Vision
 - Insight Engines
 - Cognitive Computing
 - Autonomous Vehicles
- Entering the Plateau 實質生產的高原期
 - Speech Recognition
 - GPU Accelerators

人工智慧技術成熟度曲線 (2018)

(Hype Cycle for Artificial Intelligence, 2018)



資料來源：2018 Gartner

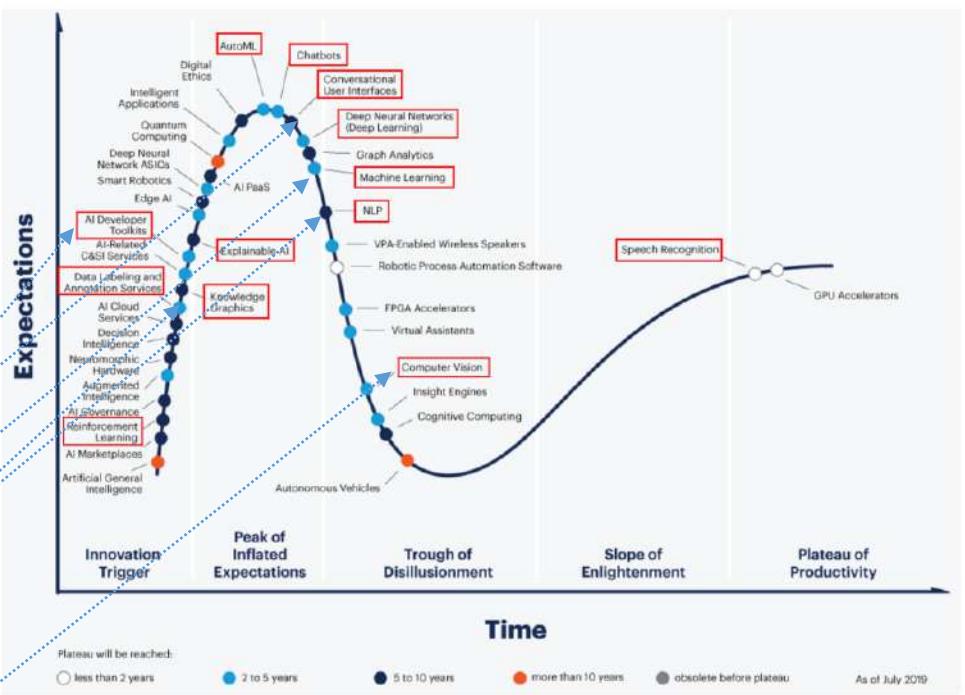
(科技誕生的促動期)

(泡沫化的底谷期)

(實質生產的高原期)

(過高期望的峰值)

(穩步爬升的光明期)



資料來源：2019 Gartner

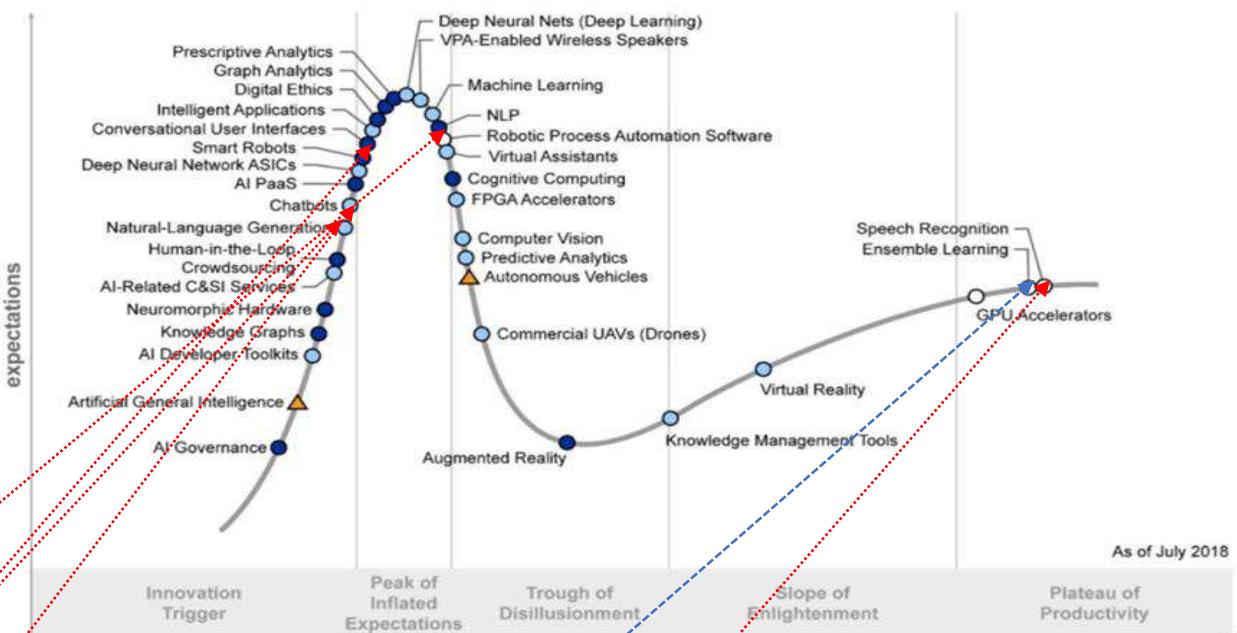


資料來源：2018 Gartner

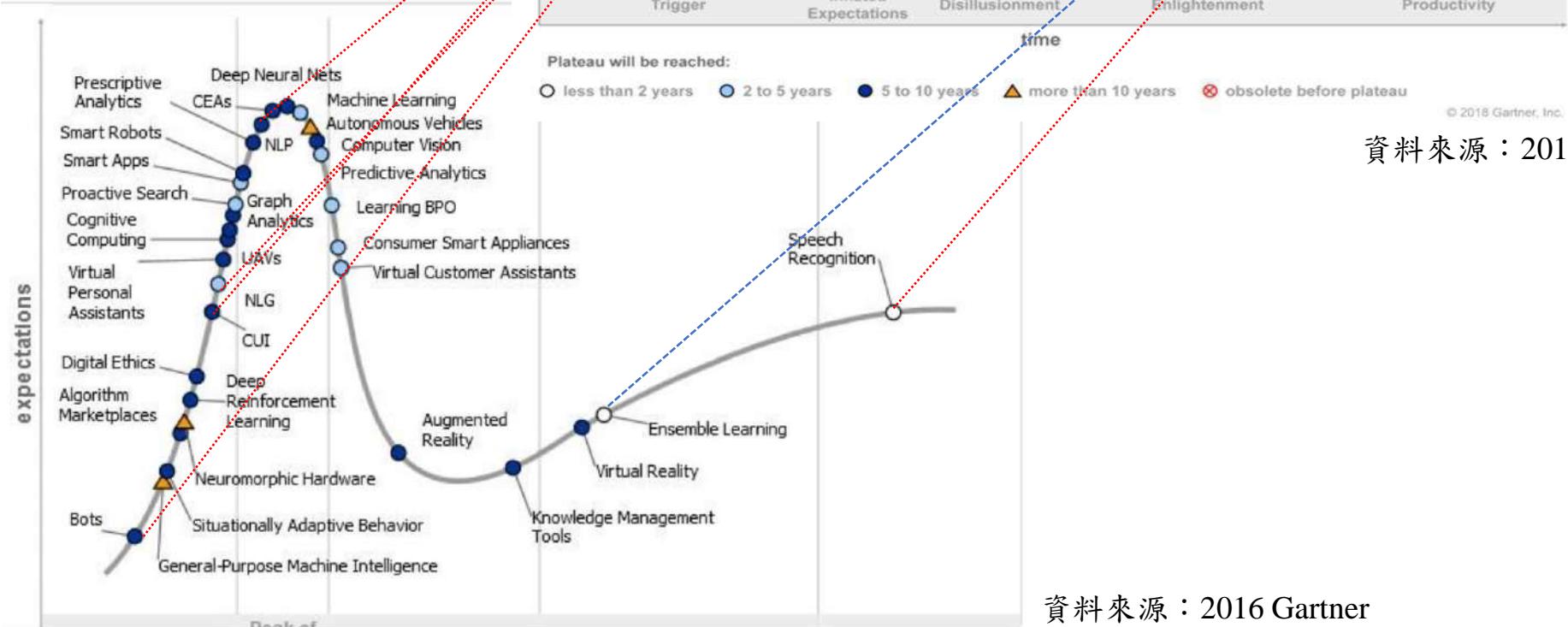
Plateau will be reached:

- less than 2 years 2 to 5 years 5 to 10 years more than 10 years obsolete before plateau

Hype Cycle for Artificial Intelligence (2018)



Hype Cycle for Smart Machines (2016)



資料來源：2018 Gartner

資料來源：2016 Gartner

(科技誕生的促動期)

(過高期望的峰值)

(實質生產的高原期)
(穩步爬升的光明期)

International Joint Conference on AI

- IJCAI at 50!
 - ~580 participants in 1969
 - 2x in 1981, 10x in 1985
 - decrease later ...
 - ... and rise again
 - > 3,000 registrants (15% more than last year at end)
- Main Track Submissions
 - 2007: 1,365
 - 2009: 1,291 (5% **decrease**)
 - 2011: 1,325 (3% increase)
 - 2013: 1,473 (11% increase)
 - 2015: 1,996 (36% increase)
 - 2016: 2,296 (15% increase)
 - 2017: 2,540 (11% increase)
 - 2018: 3,470 (37% increase)
 - 2019: 4,752 (37% increase)



IJCAI 2017		
Topics	Number of Submissions	
Machine Learning	734	41.92%
KR, Reasoning, and Logic	192	10.97%
Agent-based and MAS	149	8.51%
Multidisc. Topics and Apps	147	65.85%
NLP	145	8.28%
Planning and Scheduling	90	5.14%
Robotics and Vision	82	4.68%
Constraints and Satisfiability	81	4.63%
Uncertainty in AI	78	4.45%
Comb. & Heuristic Search	53	3.02%

IJCAI 2018				
Topics	Number of Submissions		Number of Acceptances	
Machine Learning	1808	32.34%	356	31.70%
Computer Vision	613	10.96%	131	11.67%
Machine Learning Applications	609	10.89%	122	10.86%
Multi-agent Systems	498	70.15%	100	8.90%
Natural Language Processing	480	8.59%	102	9.08%
Knowledge Representation	412	7.37%	81	7.21%
Humans and AI	260	4.65%	38	3.38%
Planning and Scheduling	235	4.20%	44	3.92%
Search and Game Playing	190	3.40%	42	3.74%
Uncertainty in AI	188	3.36%	41	3.65%
Constraints and Satisfiability	162	2.90%	41	3.65%
Robotics	136	2.43%	25	2.23%

Accepted papers, by area in IJCAI 2019

- Machine learning: 2,516 submitted, accepted 438
- Computer vision: 833 submitted, 117 accepted
- Machine learning applications: 785 submitted, 144 accepted
- Multi-agent systems: 518 submitted, 121 accepted
- Natural language processing: 630 submitted, 103 accepted
- Knowledge representation: 349 submitted, 88 accepted
- Humans and AI: 280 submitted, 50 accepted
- Planning and scheduling: 217 submitted, 48 accepted
- Search and game playing 222 submitted, 43 accepted
- Uncertainty in AI: 185 submitted, 43 accepted
- Constraints and satisfiability: 128 submitted, 30 accepted
- Robotics: 127 submitted, 20 accepted
- Multidisciplinary Topics and Applications: 617 submitted, 119 accepted

Artificial Intelligence Journal Award

IJCAI 2017

- Prominent Paper Award
 - **BabelNet**: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network
 - **YAGO2**: A spatially and temporally enhanced knowledge base from Wikipedia

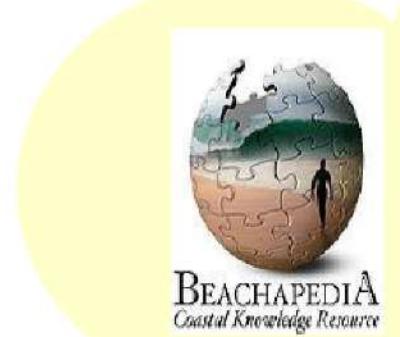
Knowledge Graphs

IJCAI 2019

- Classic Paper Award
 - Published at least 15 years ago
 - Between MDPs and semi-MDPs: A framework for temporal abstraction in **reinforcement learning**
- Prominent Paper Award
 - Published not more than 7 years ago
 - The **dropout learning** algorithm

dropout: 降低overfitting的方法

True Knowledge®
The Internet Answer Engine™
BETA



Linked movie database



**TextRunner/
ReVerb**

WolframAlpha™ computational knowledge engine



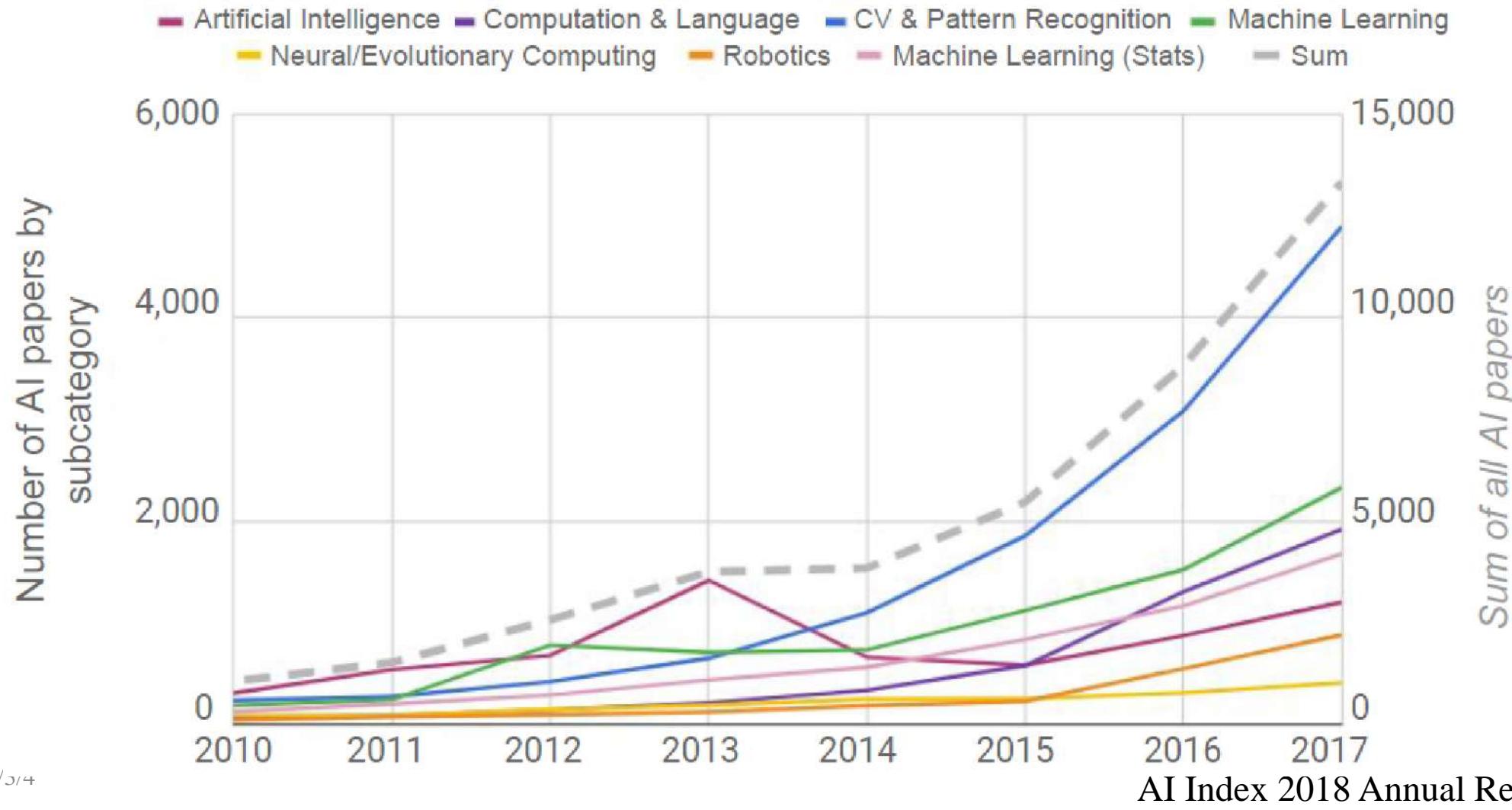
As of September 2011

The 57th Annual Meeting of the Association for Computational Linguistics (ACL)

- Submissions
 - A 75% increase over ACL 2018
 - Submissions from 74 countries/regions, including a few from Antarctica (南極洲)
 - 2,694 valid submissions
- More than 3,000 participants
- Top 5 dominating areas
 - Information Extraction and Text Mining (9%)
 - Machine Translation (8%)
 - Machine Learning (7%)
 - Dialogue and Interactive Systems (7%)
 - Generation (6%)

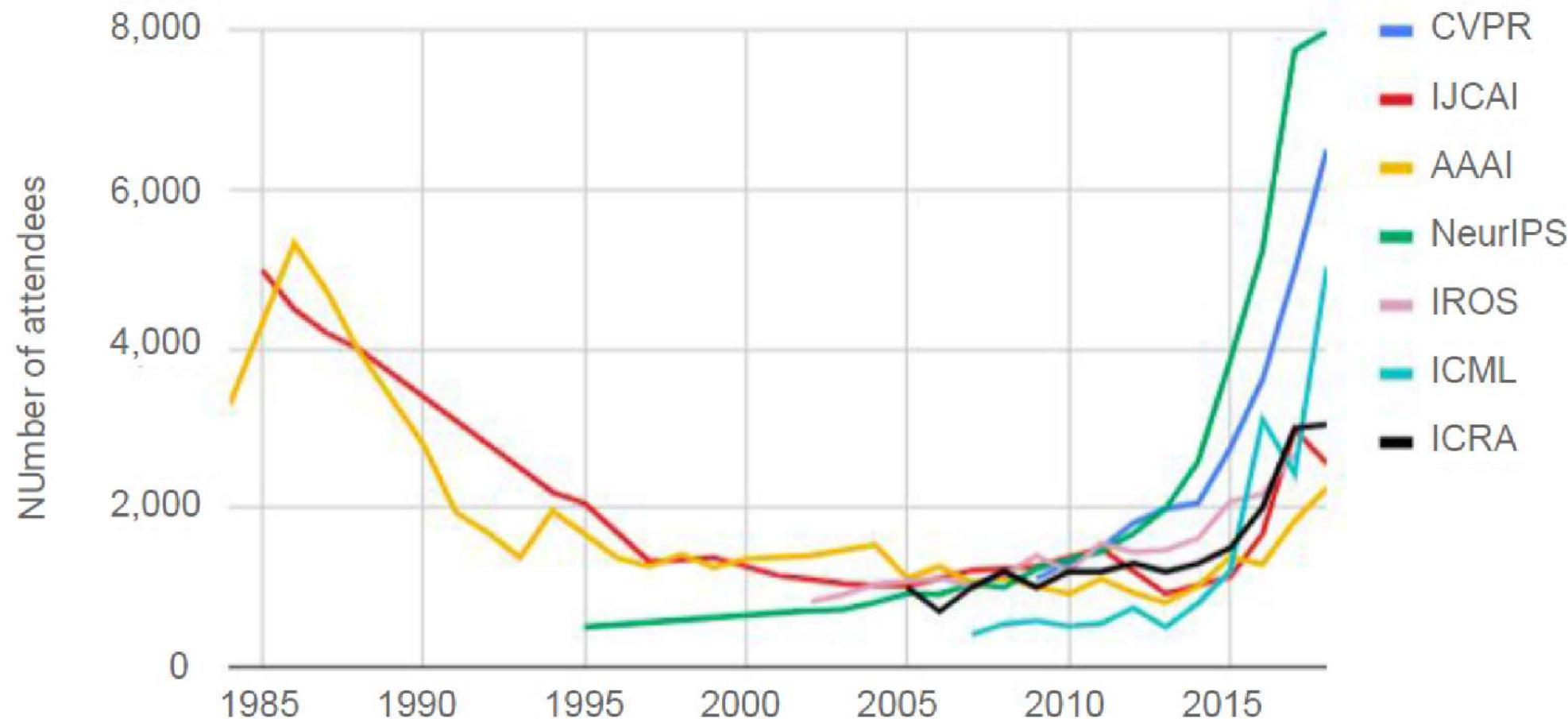


Number of AI papers on arXiv by Subcategory (2010—2017)

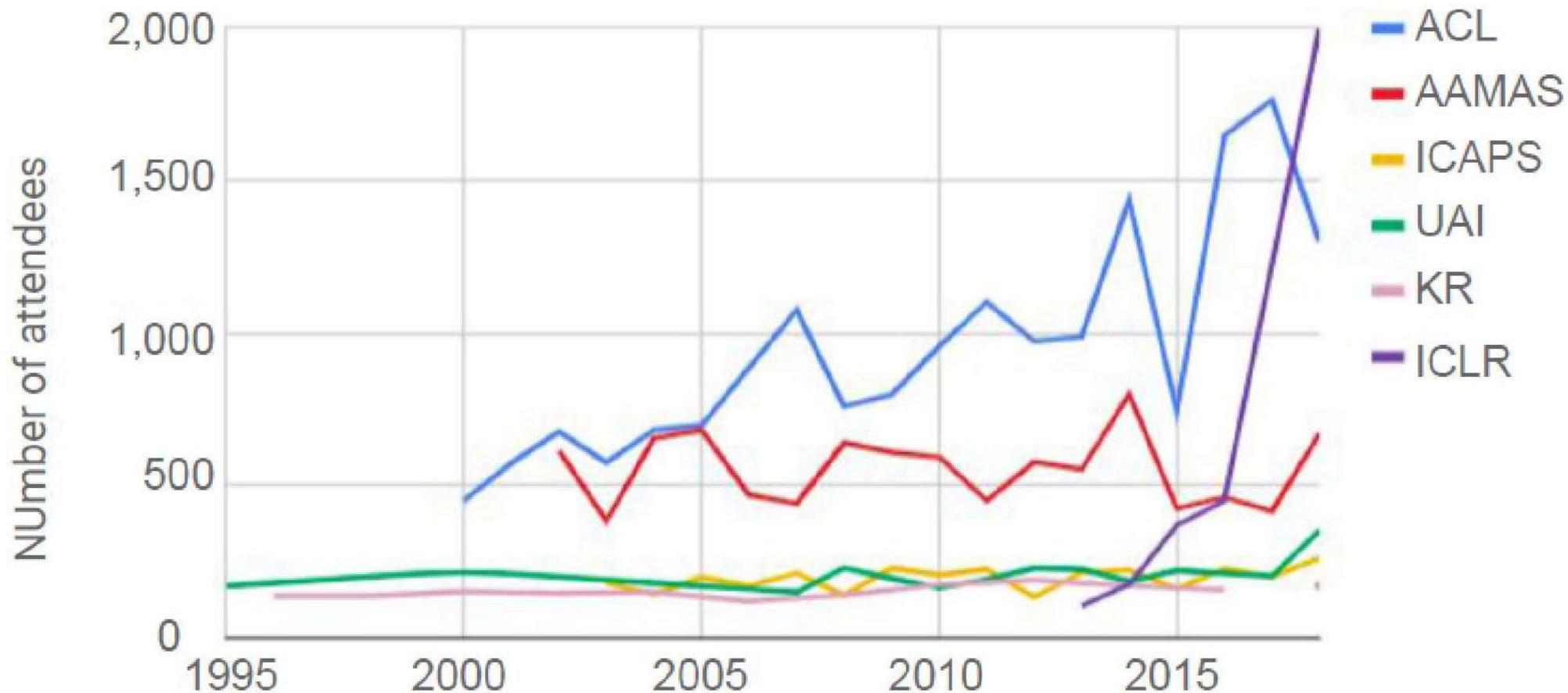


Attendance at Large Conferences (1984—2018)

大型 AI 會議是指 2017 年有超過 2000 位參會者的會議

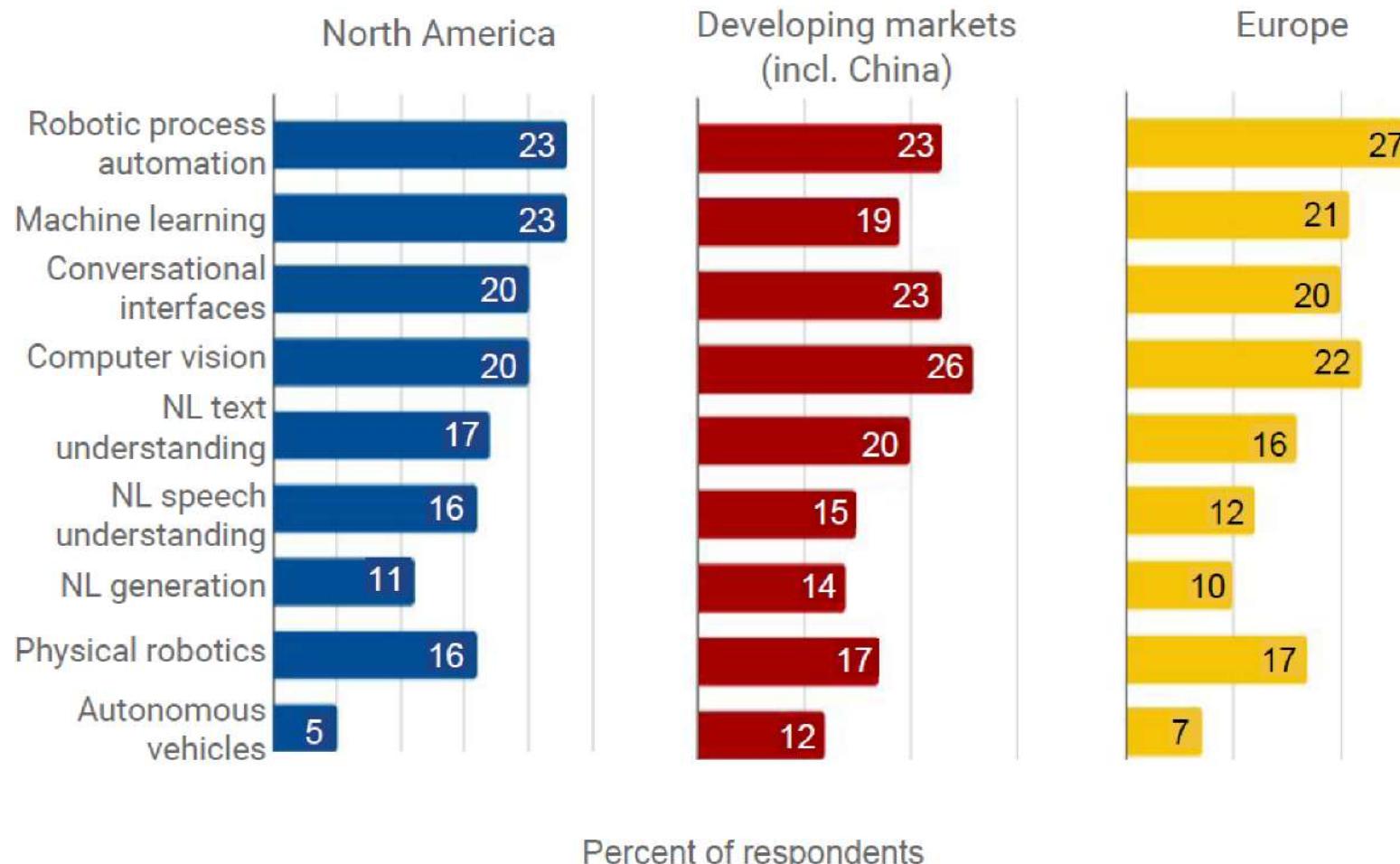


Attendance at Small Conferences (1995—2018)

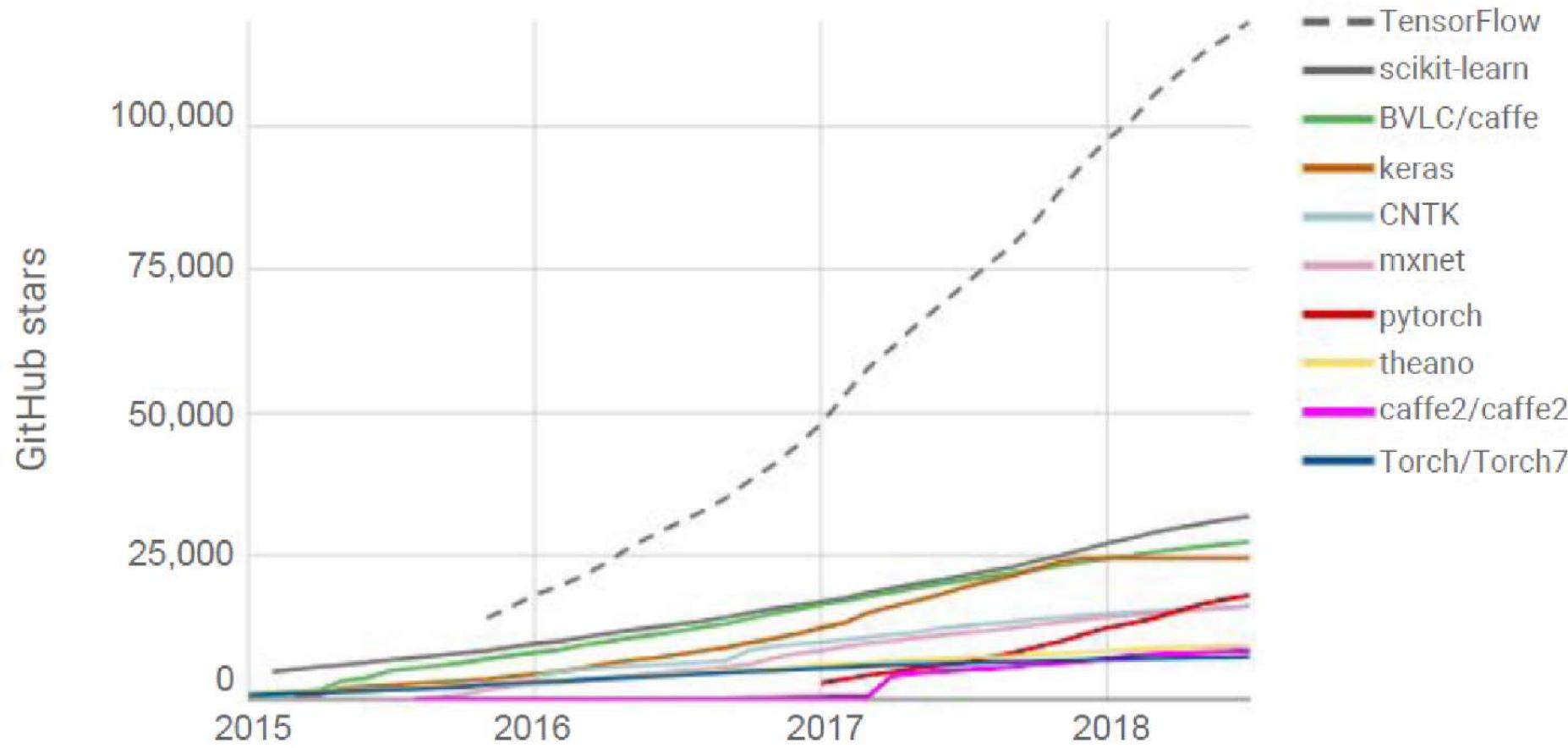


Capabilities Embedded in at Least One Company Function (2018)

(Source: McKinsey & Company)



Cumulative GitHub stars by AI library (2015—2018)(Source: GitHub)



Learning by Using Both Data and Knowledge in NLP

What is Representation?

- Mentioning a concept is fundamental in conversation
- Symbolic representation is a natural way in real world



A large, colorful word cloud centered around the word "cat" in various languages. The words are arranged in a cluster, with "cat" being the largest and most central word. Other prominent words include "kissa", "catt", "pisică", "mačka", "KOT", "gato", "kočka", "chat", "cat", "qattus", "kat", and "kötür". The words are in different colors and sizes, and some have small arrows pointing to their meanings or related words in other languages.



Pitcher of New York Yankees

Person Name



市委常委政法委書記



Reporter of
XinHua News Agency



CEO of Boeing China

王建民

Chien-Ming Wang



Research Fellow of
Chinese Academy of Social Sciences



Japanese Department Chair



Associate Fellow
of Academia Sinica



分院院長

(Wei, Lin and Chen, WI 2006)

Ambiguity



bat

蝙蝠



球棒



球拍

Representation Schemes in NLP

- Symbolic Representation
- Distributional Representation
- Distributed Representation

Meaningful Units in Natural Languages

- Morpheme: -ly, -ful, -able, un-, 們
- Character: a-z, A-Z, 台, 灣, 大, 學
- Word: bat, cat, 台灣大學
- N-gram: 台灣, 灣大, 大學, 台灣大, 灣大學
- Multiple Word Expression: 網絡紅人, 網紅
- Named Entity: 台灣積體電路股份有限公司, 台積電, 王建民
- Clause
- Sentence
- Paragraph
- Passage
- Document
- Multi-Document

Document

Sentences

人類語言是人和人互動，傳遞資訊很重要的媒介，人類的知識也是透過語言文字記錄下來。電腦科學的研究，長久以來就把電腦是否具被人類語言處理能力，視為電腦是否具有人的智慧的重要指標之一。自然語言處理探討人類語言的分析與生成，終極目標是電腦與使用者直接以人的語言互動。

由於語言文字是知識呈現的重要媒介，自然語言處理的素材相當多，特別是網際網路興起後，大量數位化的內容不僅唾手可得，而且反應真實世界不同型態語言的使用。新聞、電子郵件、網頁、維基百科、部落格貼文、微網誌、論壇、科技論文、電子病歷、瀏覽紀錄、...等不同類型來源的數位資料，都是可能分析探討的對象。

在 Jeopardy 人與電腦益智問答比賽，華生 DeepQA 系統贏過兩位益智問答高手簡寧斯和洛特，自然語言處理是這個智慧問答系統的核心技術之一。此外，智慧問答系統已經被導入一般生活應用中，Apple 的 Siri、Google 的 Assistant、Microsoft 的 Cortana、Amazon 的 Alexa 等都是商業化的產品。機器翻譯系統，例如 Google Translate，將一種語言所撰寫的資訊轉換成另外一種語言呈現，降低資訊傳遞上語言的障礙，一直是人工智能的典型應用範例。除了直接的語言文字應用外，輿情分析、病歷探勘、金融科技、健康照護、法律諮詢、烹飪教學等都有自然語言處理的影子。

Paragraph

Paragraph

Paragraph

Categorization for Lexical Items

- Parts of Speech (Lexical Categories)

狀元樓是南京老牌酒店了，在夫子廟入口的地方，
遍布我喜歡的小吃店。客房格局挺古老的，面積
不大，不過景觀很好。

狀元(Na) 樓(Nc) 是(SHI) 南京(Nc) 老牌(VH) 酒店(Nc) 了(T) ，(COMMACATEGORY)

在(P) 夫子廟(Na) 入口(Nc) 的(DE) 地方(Na) ，(COMMACATEGORY)

遍布(VH) 我(Nh) 喜歡(VK) 的(DE) 小吃店(Nc) 。(PERIODCATEGORY)

客房(Nc) 格局(Na) 挺(Dfa) 古老(VH) 的(DE) ，(COMMACATEGORY)

面積(Na) 不(D) 大(VH) ，(COMMACATEGORY)

不過(Cbb) 景觀(Na) 很(Dfa) 好(VH) ，(COMMACATEGORY)

Na: 普通名詞, Nc: 地方詞, VK: 狀態句賓動詞, VH: 狀態不及物動詞, D: 副詞, Dfa: 動詞前程度副詞

(CKIP Segmente)

Categorization for Lexical Semantics

- Semantic Categories



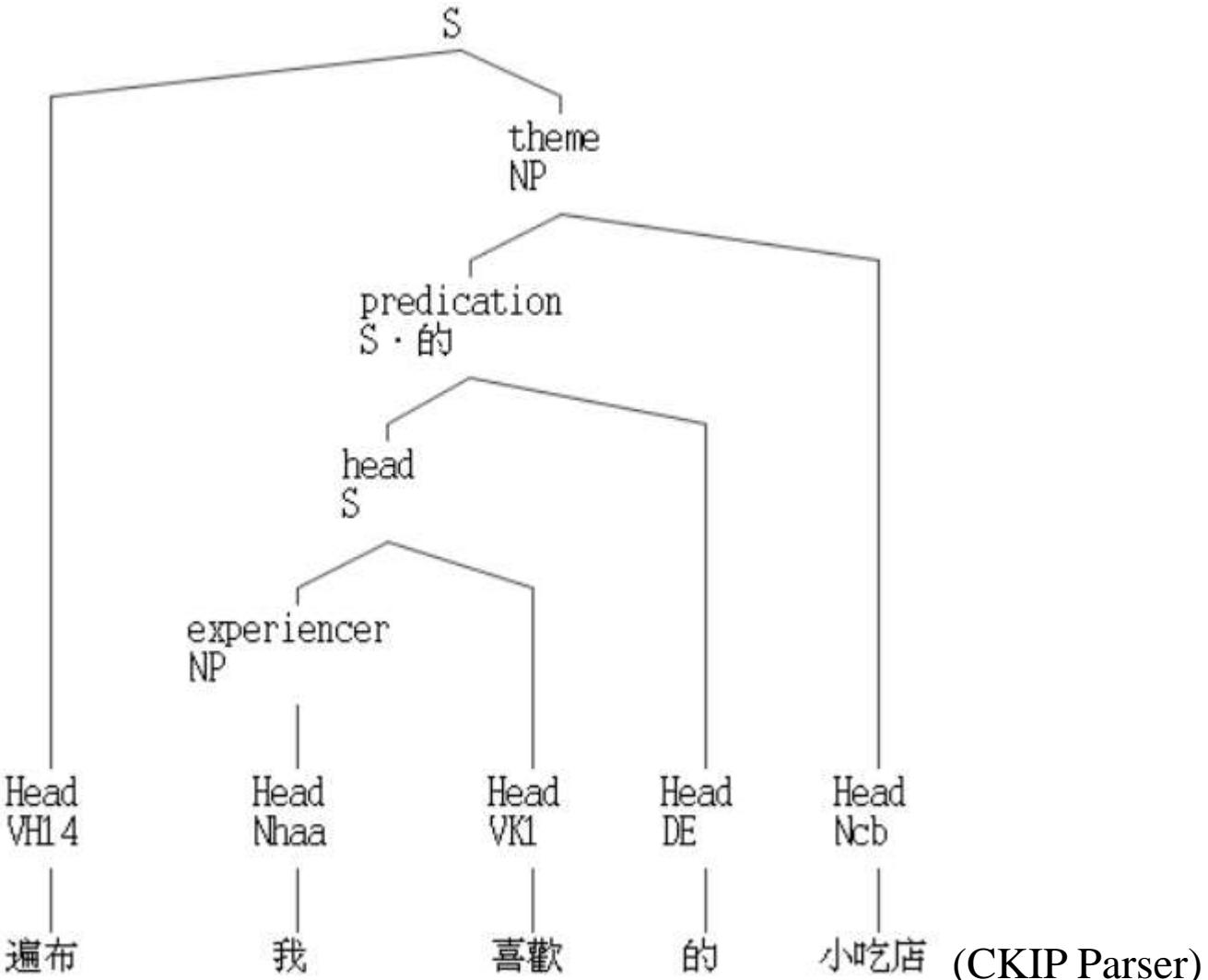
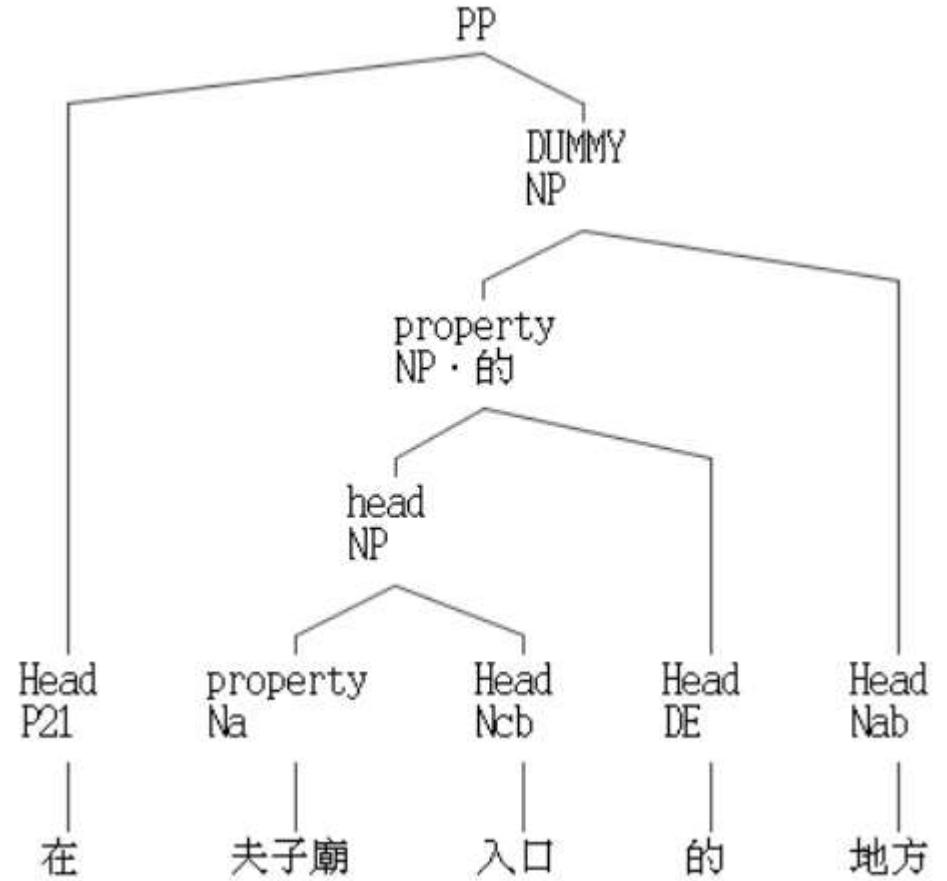
夕陽、斜陽、殘陽、落日 (Bd02-3, 同義詞詞林)



bat, chiroptera (synset in WordNet)

Categorization for Constituent Relations

- Syntactic Categories



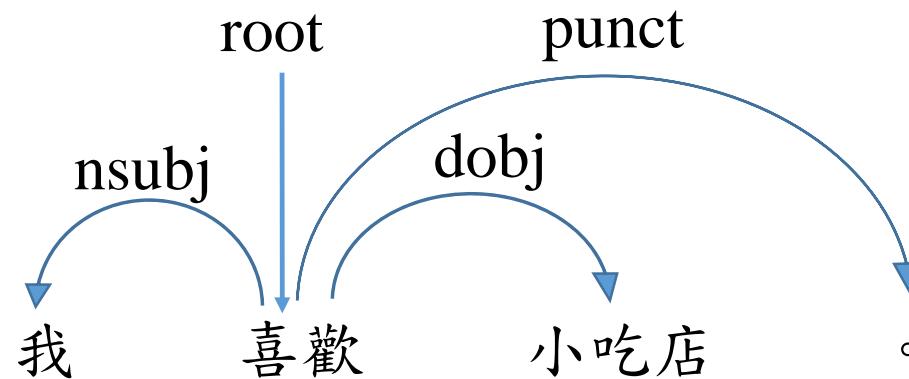
(CKIP Parser)

Categorization for Word Relations

- Dependency Relations

我喜歡小吃店。

nsubj(喜歡-2, 我-1)
root(ROOT-0, 喜歡-2)
dobj(喜歡-2, 小吃店-3)
punct(喜歡-2, 。-4)



Categorization for Sentence Relations

- Discourse Relations

- 他首先證實傅爾和中谷義雄的理論。其次，他發現經絡不僅是電流的良導體，也是電磁波的良導體。*Temporal* (時序)
- 因為颱風來襲，所以學校停止上課。*Contingency* (因果)
- 法國品牌的汽車在本土市場的佔有率雖然過半，但市場份額持續萎縮。*Comparison* (轉折)
- 伏爾泰是啟蒙運動的領導者，一位偉大的思想家。*Expansion* (推展)

Categorization for Opinions

- Opinions (Positive vs. Negative)

國信狀元樓酒店

4分 2013年8月24日 評論

狀元樓是南京老牌酒店了，在夫子廟入口的地方，遍布我喜歡的小吃店。客房格局挺古老的，面積不大，不過景觀很好，可以看見秦淮河。服務態度很好，會用力的幫着推轉門（就不能裝個自動的嘛）。洗澡水很舒服。

Categorization for Emotions

- Emotions
 - Anger
 - Disgust
 - Fear
 - Joy
 - Sadness
 - Surprise

今天跟你約吃飯 不知為什麼特別緊張

(I had a meal with you today. Don't know why I was so nervous.)

也許因為一陣子沒見吧

(Maybe it's because we hadn't seen each other for a long time.)

謝謝你請我吃飯 還送我禮物

(Thank you for treating me and giving me a present.)

雖然一直叫你不要送我東西

(Although I told you not to give me anything.)

但收到的時候還是很開心

(But I was very happy receiving it.)

當打開禮物的時候 整個傻眼

(When I opened the gift, I was astonished.)

居然送我 iPod 太誇張了

(You gave me an iPod. It's so unbelievable.)

Tom so crazy but I'm so happy

Categorization for Stances

- Stances (favor vs. against)

Atheism	無神論
Climate change	氣候變化
Feminist movement	女權主義運動
Hillary Clinton	希拉蕊·克林頓
Legalization of abortion	墮胎合法化
	核能發電
	廢除死刑
	同性婚姻

NLP Tasks

- Morpheme → Morphological Analyzer
- Word → Tokenizer/Chinese Segmentation
- Named Entity → Named Entity Recognition
- Part of Speech → Part of Speech Tagger
- Syntactic Category → Syntactic Parser/Chunker
- Dependency Category → Dependency Parser
- Semantic Category → Semantic Role Labelling
- Discourse Relation → Discourse Parser
- Opinion → Opinion Mining
- Emotion → Emotion Analysis
- Stance → Stance Detector

Lexical Resources

- English
 - WordNet
 - PropBank
 - FrameNet
 - PPDB
- Chinese
 - 中研院中文詞知識庫
 - 同義詞詞林
 - 知網 Hownet

Based on Symbolic Representation

WordNet: A Lexical Database for English

Ambiguity

Multiple POS

Multiple Senses

WordNet Search - 3.1
- [WordNet home page](#) - [Glossary](#) - [Help](#)

Word to search for: bat

Display Options: (Select option to change) ▾

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **bat**, [chiropteran](#) (nocturnal mouselike mammal with forelimbs modified to form membranous wings and anatomical adaptations for echolocation by which they navigate) 蝙蝠
- S: (n) **bat**, [at-bat](#) ((baseball) a turn trying to get a hit) "he was at bat when it happened"; "he got four hits in four at-bats" 打數
- S: (n) [squash racket](#), [squash racquet](#), **bat** (a small racket with a long handle used for playing squash) 球拍
- S: (n) [cricket bat](#), **bat** (the club used in playing cricket) "a cricket bat has a narrow handle and a broad flat end for hitting" 球板
- S: (n) **bat** (a club used for hitting a ball in various games) 球棒

Verb 用球棒（或球拍）擊（球）

- S: (v) **bat** (strike with, or as if with a baseball bat) "bat the ball"
- S: (v) **bat**, [flutter](#) (wink briefly) "bat one's eyelids" 眨眼
- S: (v) **bat** (have a turn at bat) "Jones bats first, followed by Martinez"
- S: (v) **bat** (use a bat) "Who's batting?"
- S: (v) [cream](#), **bat**, [clobber](#), [drub](#), [thrash](#), [lick](#) (beat thoroughly and conclusively in a competition or fight) "We licked the other team on Sunday!" 面不改容

Sense Relations

球棒

- **S: (n) bat** (a club used for hitting a ball in various games)
 - *direct hyponym / full hyponym* 下位詞 棒球球棒
 - **S: (n) baseball bat, lumber** (an implement used in baseball by the batter)
 - **S: (n) paddle** (small wooden bat with a flat surface; used for hitting balls in various games) 槌板
 - **S: (n) table-tennis racquet, table-tennis bat, pingpong paddle** (paddle used to play table tennis) 乒乓球球拍
 - *direct hypernym / inherited hypernym / sister term* 上位詞
 - **S: (n) club** (stout stick that is larger at one end) "he carried a club in self defense"; "he felt as if he had been hit with a club" 棍，棒
 - ***derivationally related form***
 - **W: (v) bat** [Related to: **bat**] (strike with, or as if with a baseball bat) "bat the ball"

PropBank

- Each verb sense has numbered argument: arg0, arg1, arg2, ..., argm

Frameset: f1

ARG0: caller

ARG1: telephone

ARG2: party called

Frame:

(IP (NP-SBJ (PN 我)))

(VP (ADVP (AD 已經)))

(VP (VV 打))

(NP-OBJ (NN 電話))

(PP (P 紿))

(NP-PN (NR 斯特恩))))))

ARG0: 我

ARG1: 電話

ARG2: 紿 斯特恩

ARGM-ADV: 已經

REL: 打

[arg0 我][argm-adv 已經][rel 打][arg1 電話][arg2 紿 斯特恩]
[arg1 這些算盤][arg0 產業界自己][rel 打]的[argm-ext 最精]
[arg1 托馬斯]被[arg0 泰森的鐵拳][rel 打]得[arg2 爬不起來]
[arg0 他][argm-tmp 晚上]則到體育場[rel 打][arg1 籃球]

FrameNet

Cook
Food
Container
Heating Instrument



Fry (炸), bake (烘), boil (煮), and broil (烤)



Cooking Creation

This frame describes food and meal preparation. A Cook creates a Produced_food from (raw) Ingredients. The Heating_instrument and/or the Container may also be specified.

Caitlin BAKED some cookies from the pre-packaged dough.

FEs:

Core:

Cook [Cook]

Semantic Type: Sentient

The Cook prepares the Produced_food.

Drew COOKED dinner for his friends.

Drew BAKED an apple pie.

Produced_food [Food]

The Produced_food is the result of a Cook's efforts.

Drew PREPARED dinner for his friends.

Drew BAKED an apple pie for dessert.

Non-Core:

Container [Container]

Semantic Type: Container

This FE identifies the Container that holds the food being produced.

BAKE the quiche in a pie tin.

Things that apply the heat directly are Heating_Instruments, e.g. crock-pot, electric griddle.

PPDB

- Paraphrase Database

The screenshot shows the Paraphrase.org website interface. At the top, there is a search bar with the word "dog", a language dropdown set to "English", and a "Go" button. To the right of the search bar is a "Download PPDB" button. On the left, there is a sidebar titled "Result for dog" with checkboxes for "Noun, singular or mass" (which is checked), "Noun, plural", "Adjective", and "Verb, past tense". Below this is a "Filter results" button. The main area displays "5 search results" for "dog". Each result is numbered (1 to 5) and lists a paraphrase, its part of speech, and upvote/downvote counts.

Rank	Paraphrase	Part of Speech	Upvotes	Downvotes
1	puppy	Noun, singular or mass	3	0
2	doggie	Noun, singular or mass	1	1
3	lapdog	Noun, singular or mass	0	0
4	watchdog	Noun, singular or mass	0	0
5	doggy	Noun, singular or mass	0	1

同義詞詞林

- 12大類、94中類、1428小類、3925個詞群
- 12大類
 - A 人、B 物、C 時間和空間、D 抽象事物、E 特徵、F 動作、G 心理活動、
 - H 活動、I 現象和狀態、J 關聯、K 助語、L 敬語
- 中類
 - Aa 泛稱
 - 01 人 人民 眾人
 - 02 我 我們
 - 03 你 你們
 - ...
 - 漢語同義詞詞林擴展版 (<http://www.ltp-cloud.com/download/>)

中研院中文詞知識庫

- 記載八萬多目詞的注音、語法、論元結構、及語義。
 - 詞：豐富
 - 注音: ㄊㄥ ㄉㄨˋ
 - 拼音: feng1 fu4
 - 語義特徵: +events
 - 詞性: VH16
 - 論元結構: theme, causer
 - 詞：醫生
 - 注音: ㄧ ㄕㄥ
 - 拼音: yi1 sheng1
 - 語義特徵: +mankind
 - 詞性: Nab

知網 Hownet

<http://www.keenage.com>

- 中英雙語的詞彙知識庫。
- 以抽象的語義特徵（義元）來定義詞彙的意義。
- 除了詞性及英文翻譯之外，還可以得到近義詞、上下位詞、部分與全體關係、事件、及語意角色。
- 詞：醫生
 - 英文翻譯：doctor, physician, surgeon
 - {human|人: HostOf={Occupation|職位 },domain={medical|醫 },{doctor|醫治:agent=~ } }

Symbolic Computation

- Matching
 - Resource Lookup by Keyword Matching

```
<Word item = "打">
  <WordFreq>1461</WordFreq>
  <WordSense id="1">
    <English>generalized verb of doing with specific meaning determined by its object,strike,hit,fight,construct,forge,mix</English>
    <Phone>ㄉㄚˇ</Phone>
    <PinYin>da3</PinYin>
    <SyntacticFunction>
      <POS>VC2</POS>
      <Freq>1448</Freq>
    </SyntacticFunction>
    <TopLevelDefinition>{do|做}</TopLevelDefinition>
    <BottomLevelExpansion>{do|做}</BottomLevelExpansion>
  </WordSense>
  <WordSense id="2">
    <English>dozen</English>
    <Phone>ㄉㄚˇ</Phone>
    <PinYin>da3</PinYin>
    <SyntacticFunction>
      <POS>Nfg</POS>
      <Freq>13</Freq>
    </SyntacticFunction>
    <TopLevelDefinition>quantity={打}</TopLevelDefinition>
    <BottomLevelExpansion>quantity={打}</BottomLevelExpansion>
  </WordSense>
```

單賓 「打手飾」、「打毛衣」

標準量詞 「一打毛巾」

```
<WordSense id="3">
    <English>beat</English>
    <Phone>ㄉㄚˋ</Phone>
    <PinYin>da3</PinYin>
    <SyntacticFunction>
        <POS>VC2</POS>
        <Freq>0</Freq>
    </SyntacticFunction>
    <TopLevelDefinition>{beat|打}</TopLevelDefinition>
    <BottomLevelExpansion>{beat|打}</BottomLevelExpansion>
</WordSense>
<WordSense id="4">
    <English>from</English>
    <Phone>ㄉㄚˋ</Phone>
    <PinYin>da3</PinYin>
    <SyntacticFunction>
        <POS>P17</POS>
        <Freq>0</Freq>
    </SyntacticFunction>
    <TopLevelDefinition>LocationIni={}</TopLevelDefinition>
    <BottomLevelExpansion>LocationIni={}</BottomLevelExpansion>
</WordSense>
<WordSense id="5">
    <English>since</English>
    <Phone>ㄉㄚˋ</Phone>
    <PinYin>da3</PinYin>
    <SyntacticFunction>
        <POS>P17</POS>
        <Freq>0</Freq>
    </SyntacticFunction>
    <TopLevelDefinition>TimeIni={}</TopLevelDefinition>
    <BottomLevelExpansion>TimeIni={}</BottomLevelExpansion>
</WordSense>
```

單賓

「打壞人」

介詞

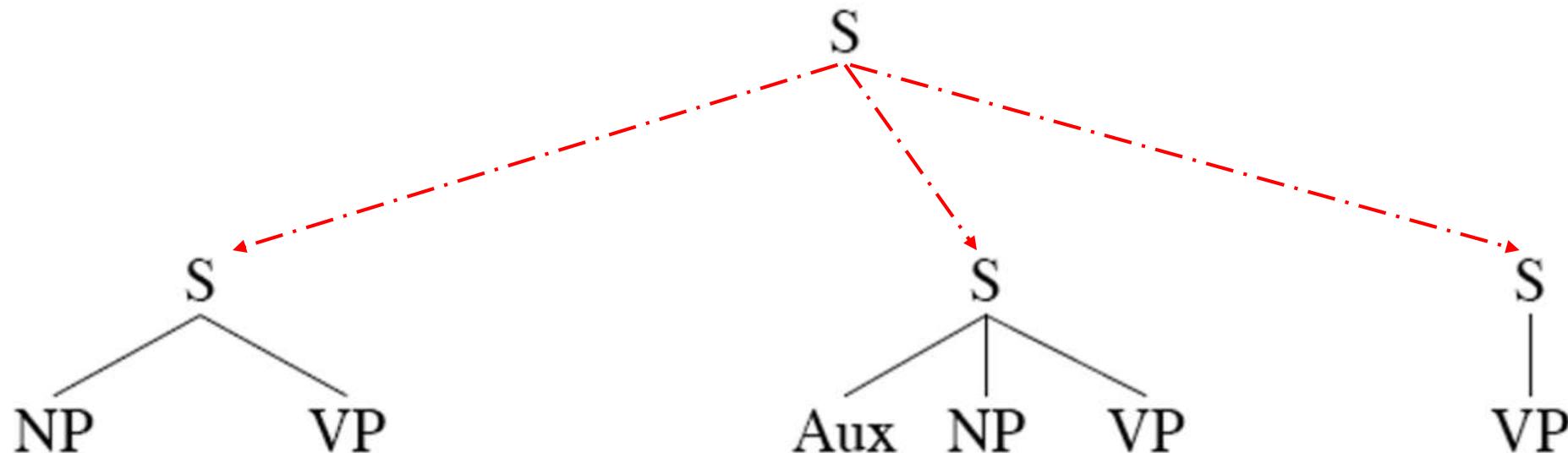
「您打那裡來？」

介詞

「打現在起，我要發奮用功。」

Symbolic Computation

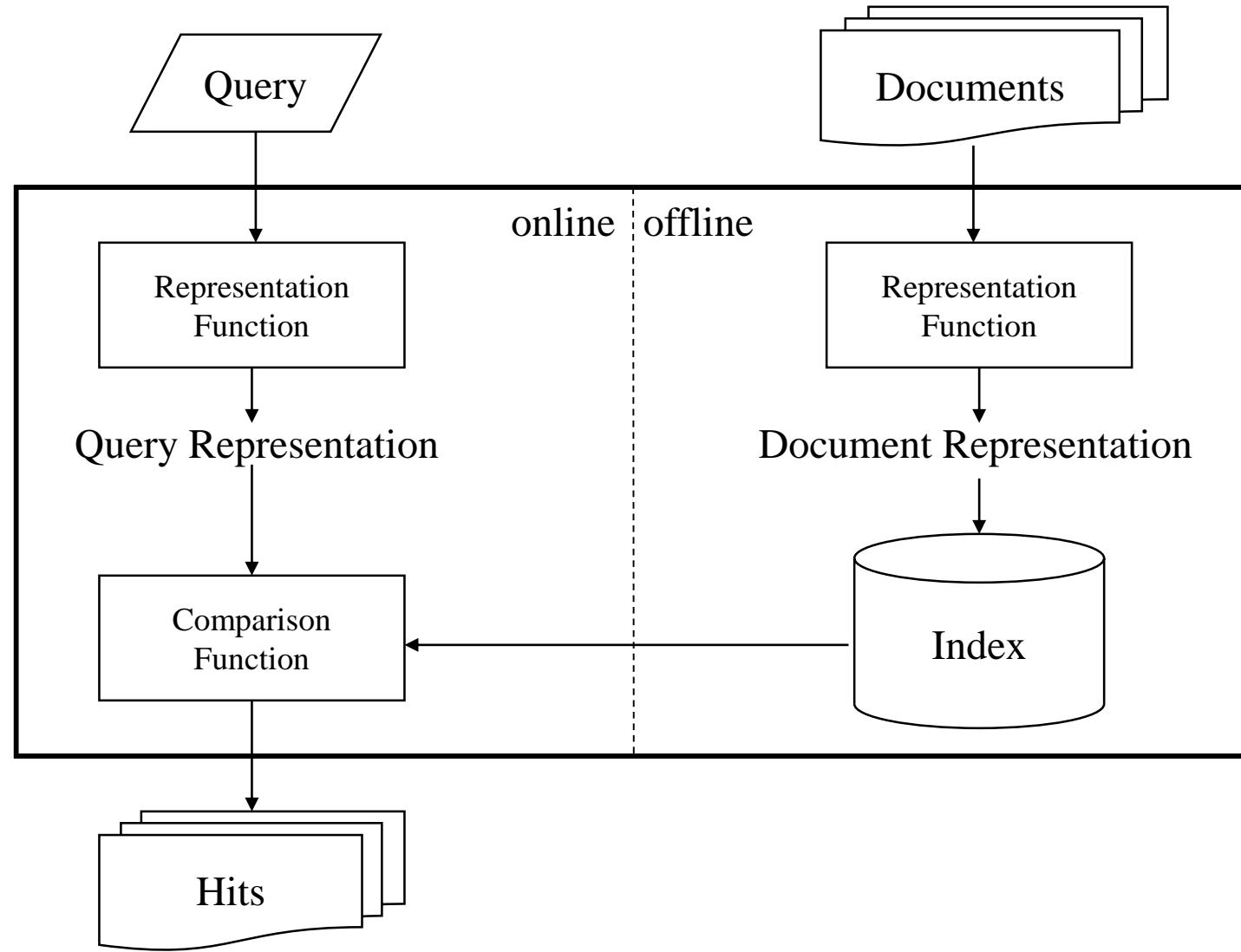
- Rule Application by Category Matching



Search Scenario

- From Information Need to Query
 - Information need: 旅遊規劃-台灣最美的夕陽景點
 - Query: 台灣夕陽
 - Expansion: 台灣落日
- 夕陽=落日
 - By thesaurus made by human (e.g., 同義詞詞林)
 - By thesaurus learned from query log, document collection, etc.
 - By feedback information
- From Keyword Matching to Concept Matching

Search Scenario



Representation Schemes in NLP

- Symbolic Representation
- Distributional Representation
- Distributed Representation

Meaning

- Meaning is denoted by symbols interpreted by human in conversation
- The interpretation of symbolic meaning by machines is via matching.
- From Symbol to distributional representation and distributed representation
 - Meaning as use!
 - You shall know a word by the company it keeps. (J.R. Firth, 1957)
 - Distributional hypothesis: words with similar meanings are likely to appear in similar contexts
- The interpretation of meaning in distributional representation by machines is via Vector Arithmetic

Distributional vs Distributed Representation

- **Count**-based representation
 - Explicit representation: construct a high dimensional sparse matrix M , where each row represents a word w in the vocabulary and each column a potential context.
 - Context: sentence, paragraph, document
 - Singular Value Decomposition (SVD) of term-context matrix
 - Term-document matrix when context is document
- **Prediction**-based (neural) embeddings
 - Continuous Bag-of-Words (CBOW)
 - Skip-Grams
 - Global Vectors (GloVe)

Distributional Word Representation

$$A = \begin{matrix} & T_1 & T_2 & T_3 & \cdots & T_t \\ D_1 & \left| \begin{matrix} a_{11} & a_{12} & \cdots & a_{1t} \end{matrix} \right. \\ D_2 & \left| \begin{matrix} a_{21} & a_{22} & \cdots & a_{2t} \end{matrix} \right. \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ D_n & \left| \begin{matrix} a_{n1} & a_{n2} & \cdots & a_{nt} \end{matrix} \right. \end{matrix}$$

- A document i.t.o. terms
- A term i.t.o. documents
- Dimension reduction on word co-occurrence matrix ($A^T A$)
 - Latent Semantic Analysis (LSA)
 - Principal Component Analysis (PCA)
- Latent Dirichlet Allocation (LDA)

Representation Schemes in NLP

- Symbolic Representation
- Distributional Representation
- **Distributed Representation**

Distributed Word Representation

- Neural networks (neural embeddings, embeddings)
 - CBOW (Continuous Bag-of-Word)
 - Skip-Gram
 - GloVe (Global Vectors for Word Representation)
 - CWINDOW
 - structured skip-gram
 - word to vector → phrase to vector → document to vector

Neural Probabilistic Language Model

- A conditional probability distribution over words in V for the next word w_t

$$p(w_t | w_{t-1}, \dots, w_{t-(n-1)}) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}$$

where

$$y = b + Wx + Utanh(d + Hx)$$

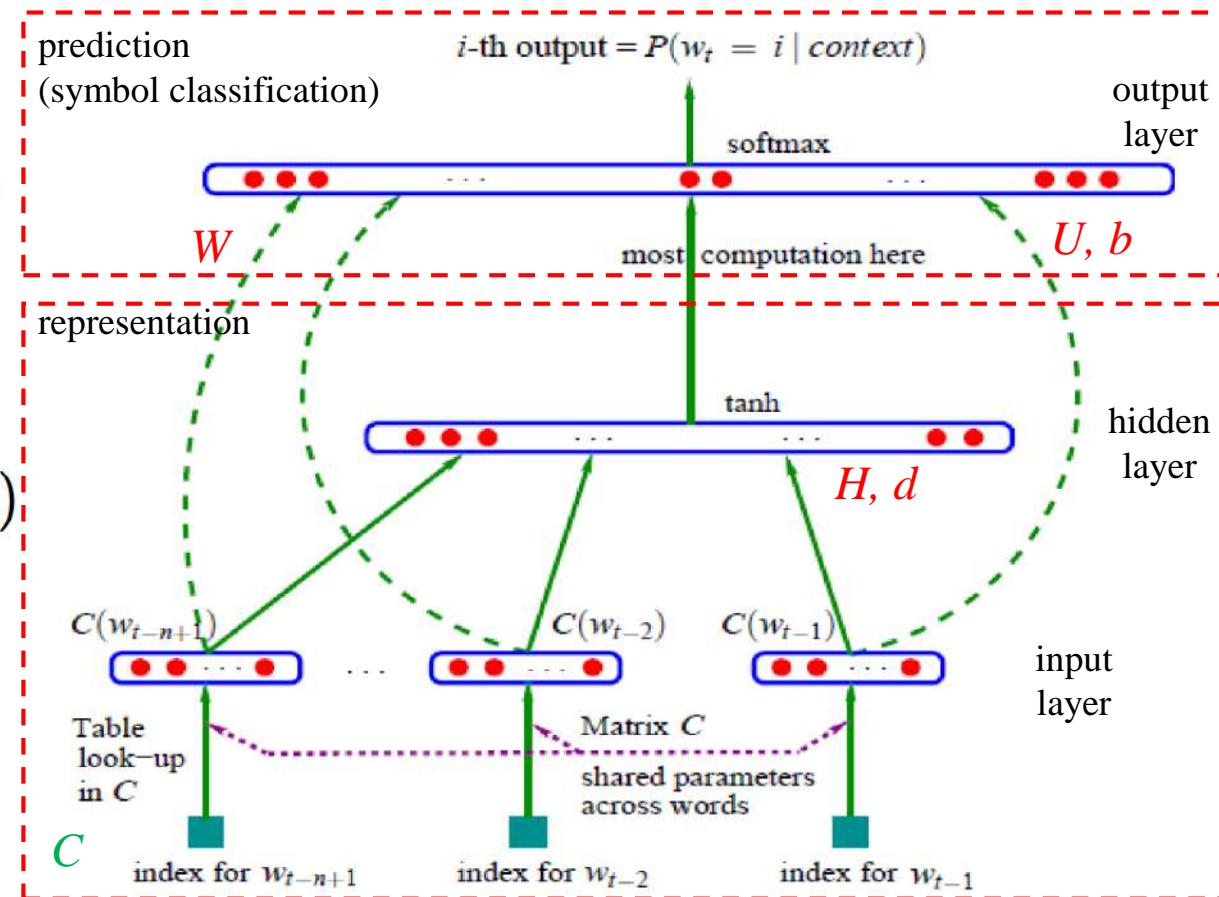
$$x = (C(w_{t-1}), C(w_{t-2}), \dots, C(w_{t-(n-1)}))$$

$$\theta = (b, d, W, U, H, C)$$

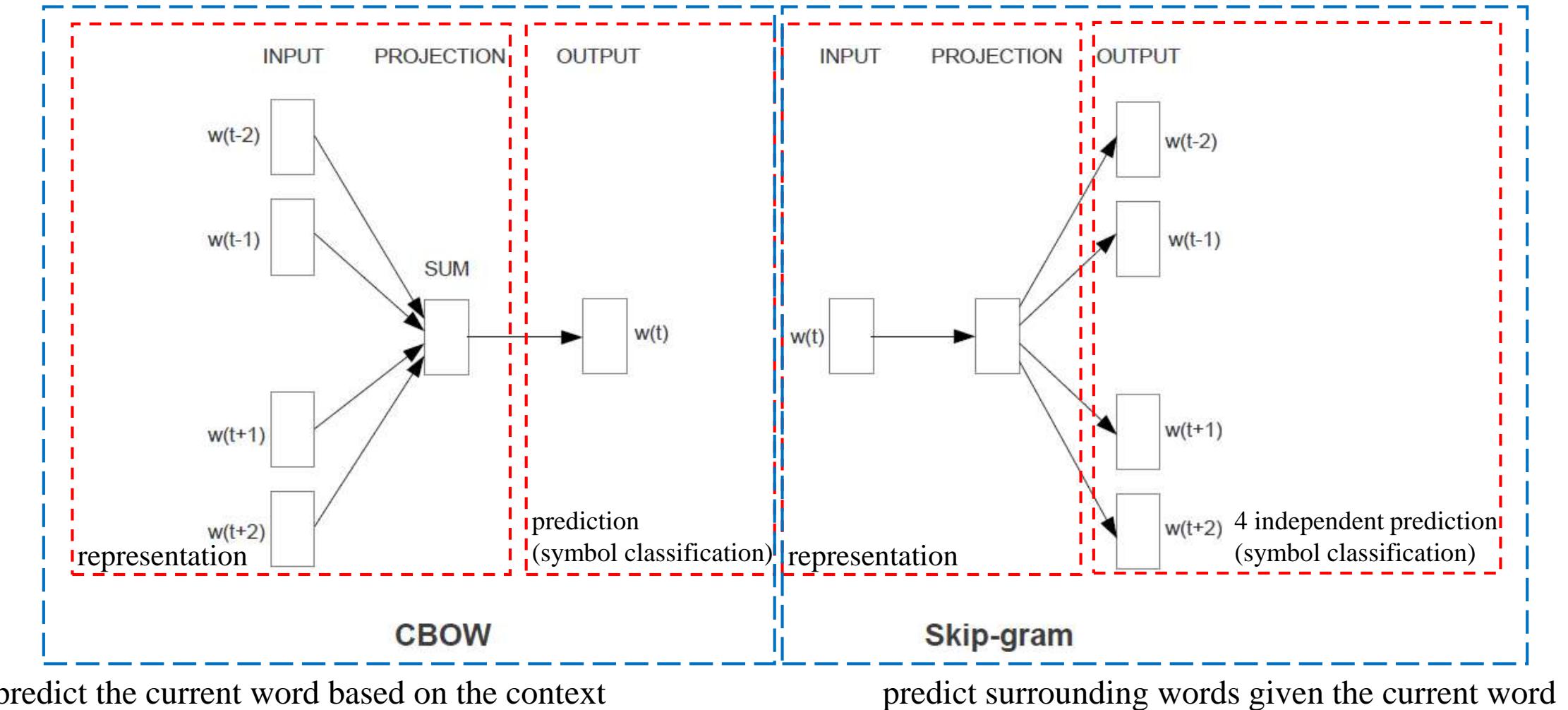
red: model parameters, green: vector representation

Maximize the average (regularized) log-likelihood

$$L = \frac{1}{T} \sum_t \log f(w_t, w_{t-1}, \dots, w_{t-(n-1)}; \theta)$$



Word Representations in Vector Space



predict the current word based on the context

predict surrounding words given the current word

Continuous Bag-of-Words (CBOW)

- Predict the current word based on the context
- $y = b + Wx$
- $\theta = (b, W, C)$

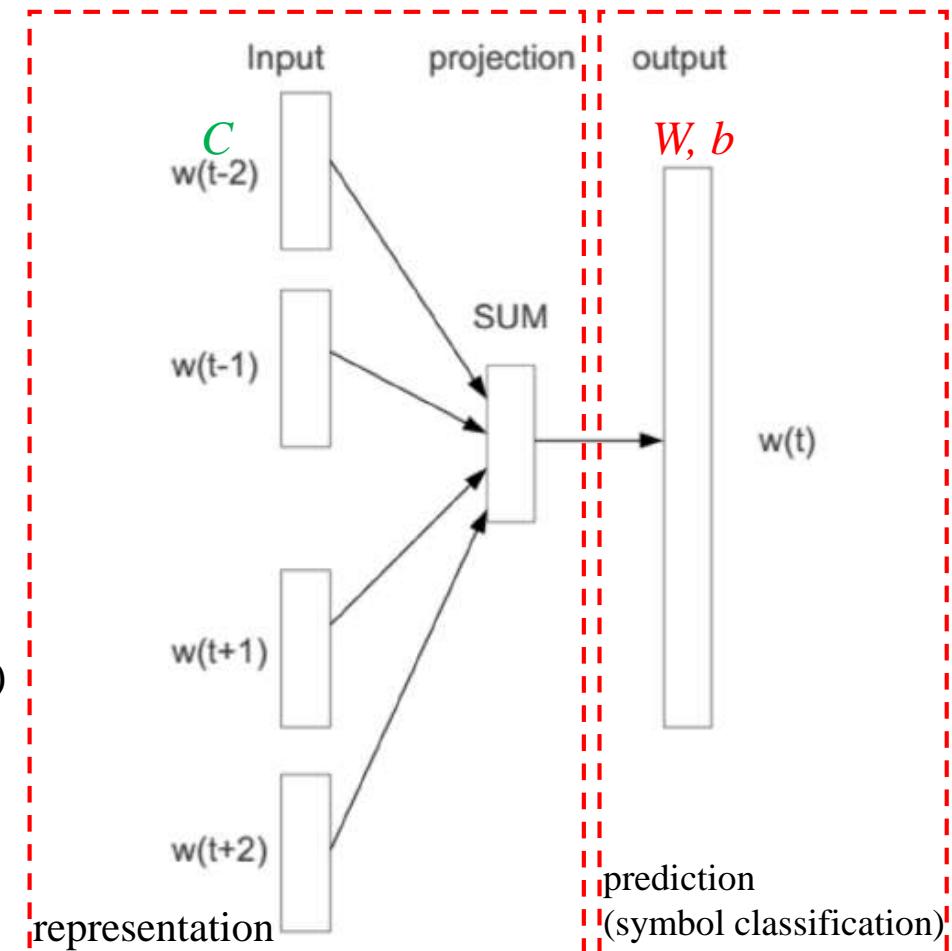
Notes:

The non-linear hidden layer is removed

The projection layer is shared for all words

All words get projected into the same position (their vectors are averaged)

The order of words in the history does not influence the projection



Skip-gram Model

- Predict the surrounding words
- $f = \prod_{-l \leq j \leq l, j \neq 0} p(w_{t+j} | w_t)$

where

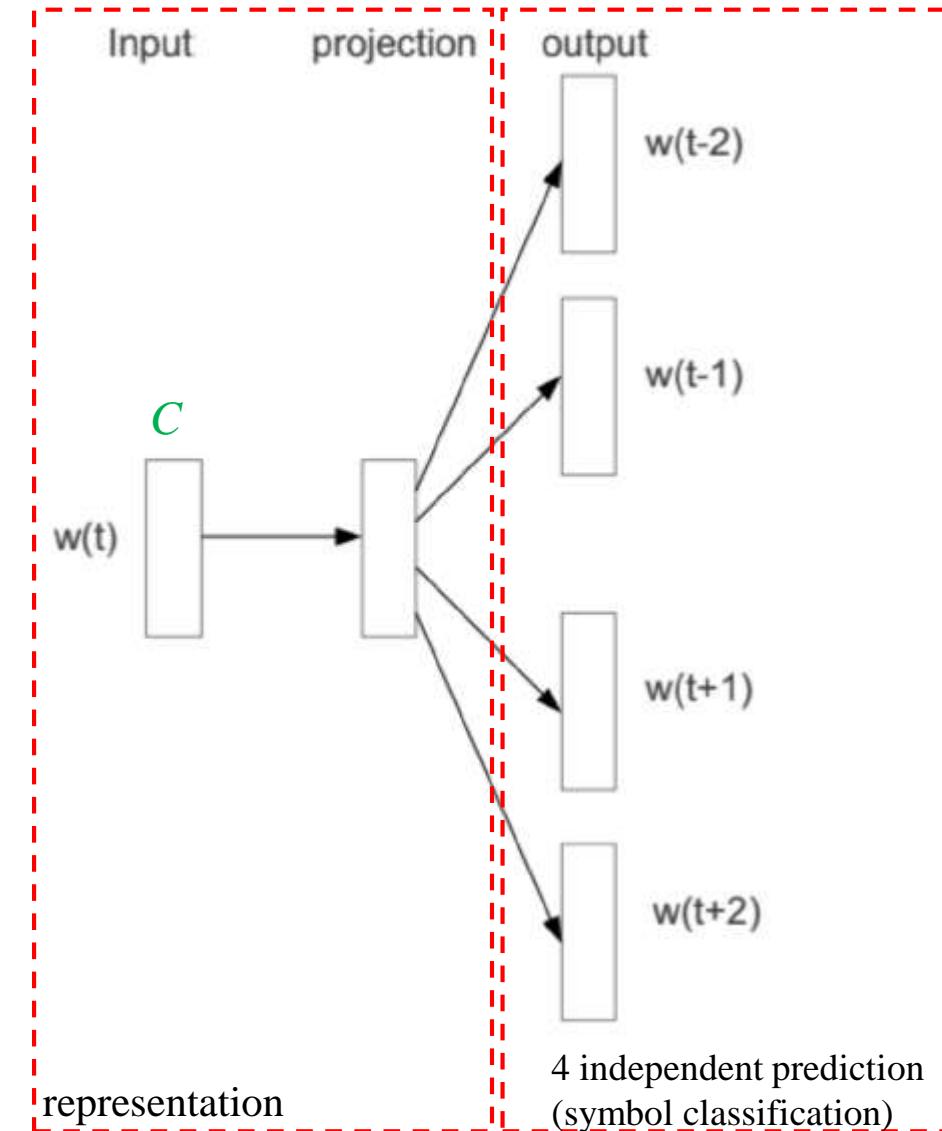
$$p(w_{t+j} | w_t) = \frac{\exp(y_{w_{t+j}}^T y_{w_t})}{\sum_i^{|V|} \exp(y_{w_i}^T y_{w_t})}$$

- $y_i = C(w_i)$
- $\theta = \textcolor{green}{C}$

Use each current word as an input to a log-linear classifier with continuous projection layer

Predict words within a certain range before and after the current word

Distributed representations of words and phrases and their compositionality
(Mikolov et al, NIPS 2013)



Distributional Representation vs. Distributed Representation

- Both count-based and prediction-based approaches
 - Rely on the same linguistic theory
 - Use the same data
 - Are mathematically related
 - Word2Vec is implicitly factorizing a matrix which is closely related to the well-known word-context PMI matrix

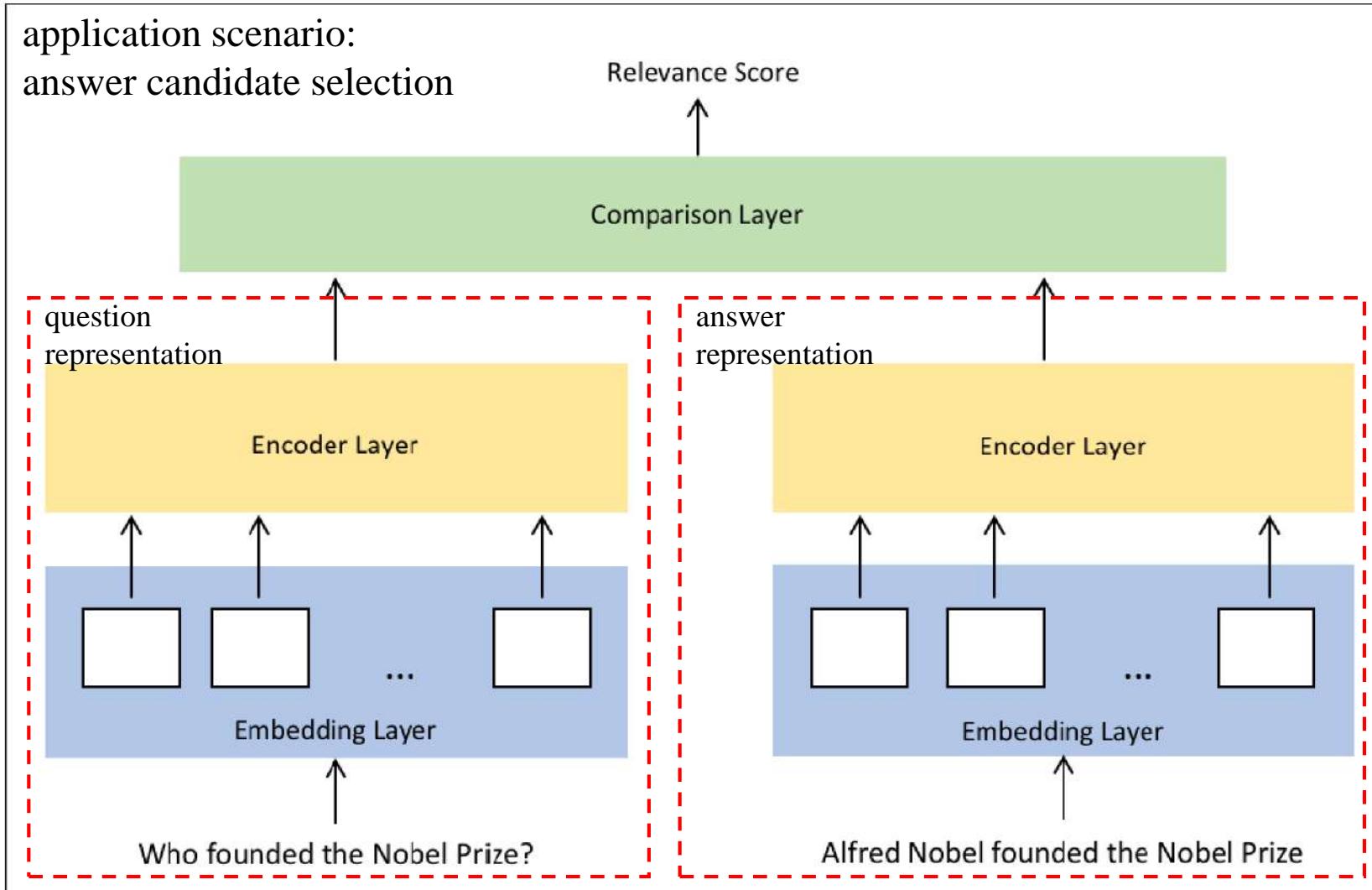
$$\mathbf{E}^W \in \mathbb{R}^{|V_W| \times d_{\text{emb}}}$$

$$\mathbf{E}^C \in \mathbb{R}^{|V_C| \times d_{\text{emb}}}$$

$$\mathbf{E}^W \times \mathbf{E}^C^\top = \mathbf{M}' \in \mathbb{R}^{|V_W| \times |V_C|}$$

$$w \cdot c = \mathbf{M}'_{[w,c]} = \text{PMI}(w, c) - \log k.$$

Sentence Similarity Computation Model



Ronan Collobert, Jason Weston, Leon
Bottou, Michael Karlen, Koray
Kavukcuoglu, and Pavel Kuksa. 2011.
**Natural Language Processing (Almost)
from Scratch.** Journal of Machine Learning
Research, 12 (2011), 2493–2537.

Experimental Setup: for Each Task

Task	Benchmark	Data set	Training set (#tokens)	Test set (#tokens)	(#tags)
POS	Toutanova et al. (2003)	WSJ	sections 0–18 (912,344)	sections 22–24 (129,654)	(45)
Chunking	CoNLL 2000	WSJ	sections 15–18 (211,727)	section 20 (47,377)	(42) (IOBES)
NER	CoNLL 2003	Reuters	“eng.train” (203,621)	“eng.testb” (46,435)	(17) (IOBES)
SRL	CoNLL 2005	WSJ	sections 2–21 (950,028)	section 23 + 3 Brown sections (63,843)	(186) (IOBES)

State-of-The-Art Systems on Four NLP Tasks

System	Accuracy
Shen et al. (2007)	97.33%
Toutanova et al. (2003)	97.24%
Giménez and Màrquez (2004)	97.16%

(a) POS

System	F1
Shen and Sarkar (2005)	95.23%
Sha and Pereira (2003)	94.29%
Kudo and Matsumoto (2001)	93.91%

(b) CHUNK

System	F1
Ando and Zhang (2005)	89.31%
Florian et al. (2003)	88.76%
Kudo and Matsumoto (2001)	88.31%

(c) NER

System	F1
Koomen et al. (2005)	77.92%
Pradhan et al. (2005)	77.30%
Haghghi et al. (2005)	77.04%

(d) SRL

Generalization Performance of Benchmark NLP Systems with a Vanilla Neural Network (NN)

Approach	POS (PWA)	Chunking (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99

Colin Raffel, Noam Shazeer, Adam Roberts,
Katherine Lee, Sharan Narang, Michael
Matena, Yanqi Zhou, Wei Li, Peter J. Liu,
**Exploring the Limits of Transfer Learning
with a Unified Text-to-Text Transformer,**
arXiv:1910.10683v2, 24 Oct 2019

T5: Text-to-Text Transfer Transformer

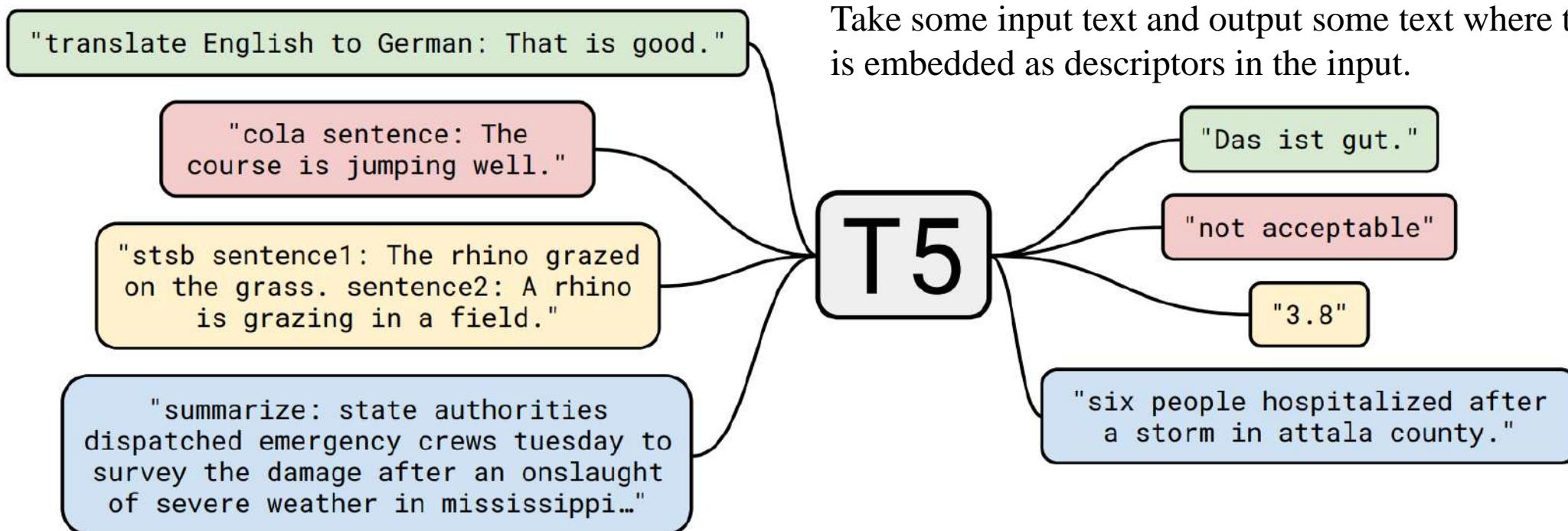


Figure 1: A diagram of our text-to-text framework. Every task we consider – including translation, question answering, and classification – is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. “T5” refers to our model, which we dub the “Text-to-Text Transfer Transformer”.

The same model is used for a wide variety of tasks

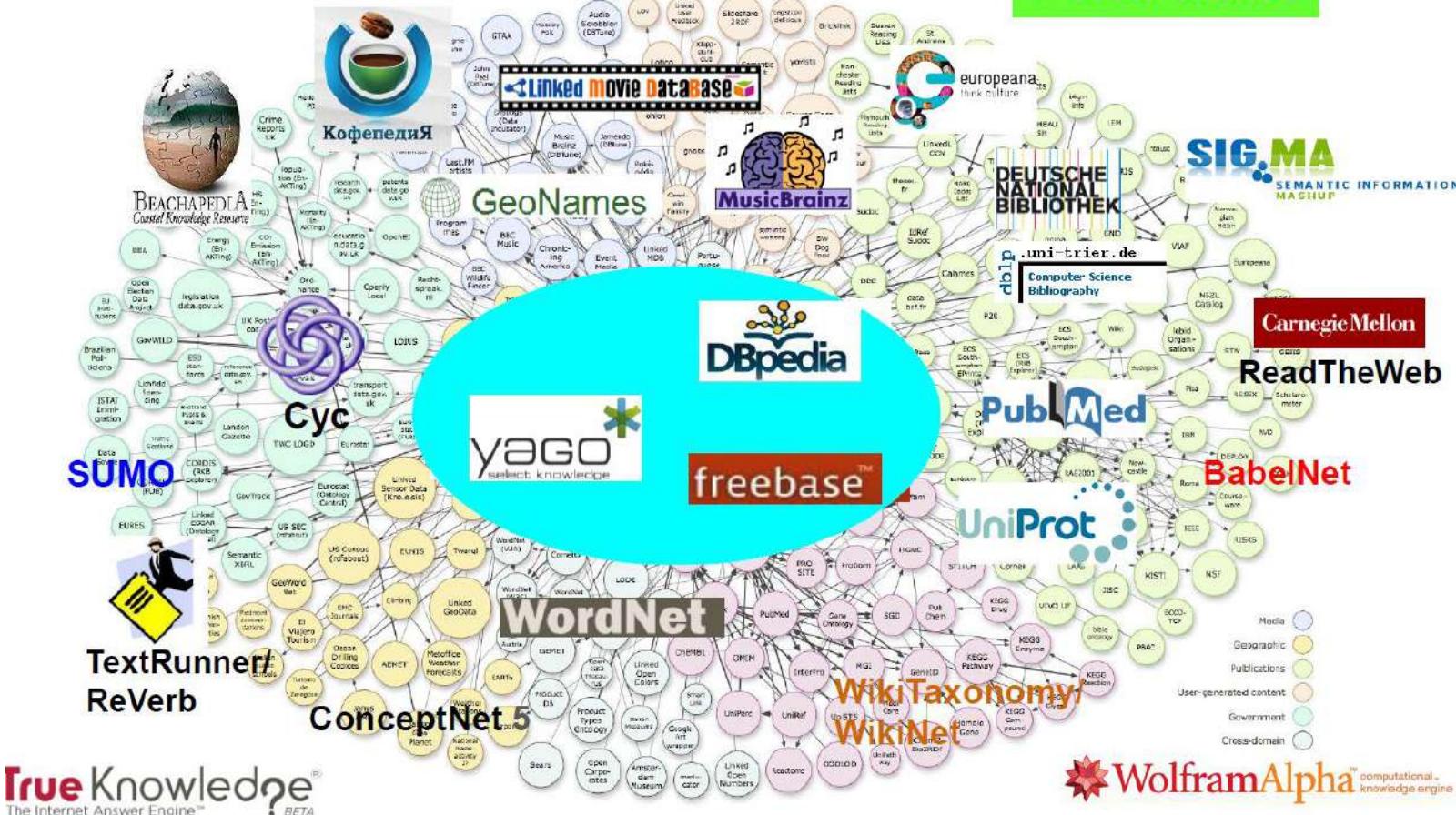
Take some input text and output some text where the task type is embedded as descriptors in the input.

Data + Knowledge

Web of Data & Knowledge

62 Bio. SPO triples (RDF) from 870 sources, and growing

+ Web tables



(Gerhard Weikum, 2013)

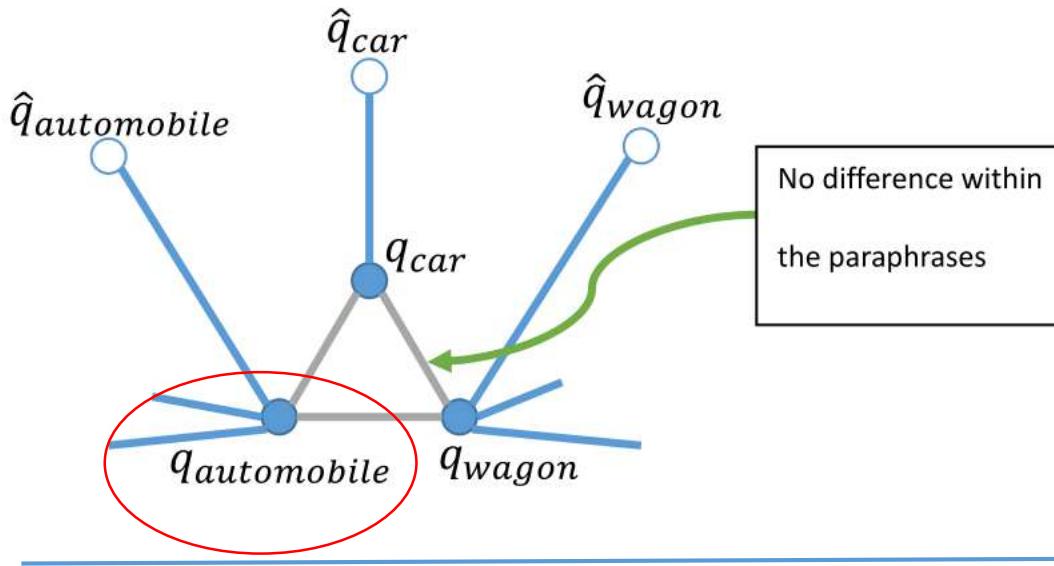
Integration of Data and Knowledge

- Why do we not use the knowledge bases accumulated in very long time?
- How to integrate data and knowledge?
- Start from lexical semantics to solve multiple-senses one-embedding problem

On Utilization of Ontology and Retrofitting Techniques for Better Distributed Representations of Words and Senses

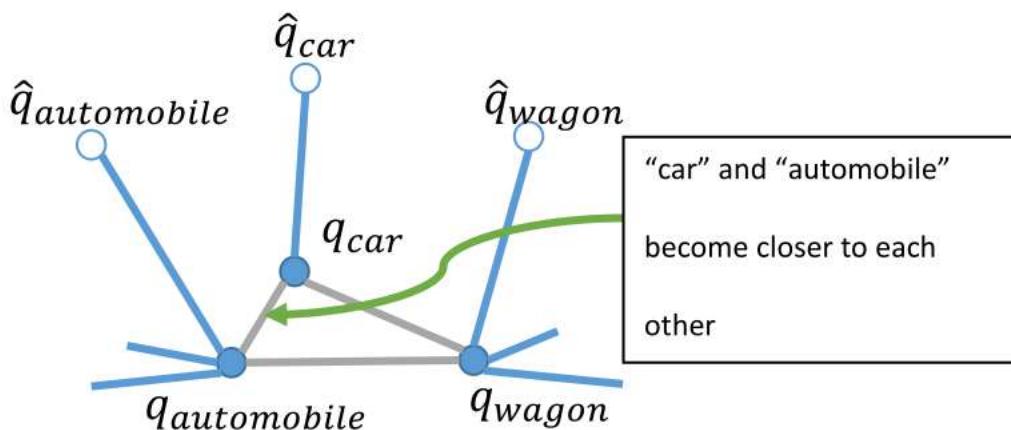
Yang-Yin Lee, Hao Ke, Ting-Yu Yen,
Hen-Hsen Huang, Hsin-Hsi Chen

Structural-fitting of Word Embedding



Retrofitting (Faruqui et al., 2015)

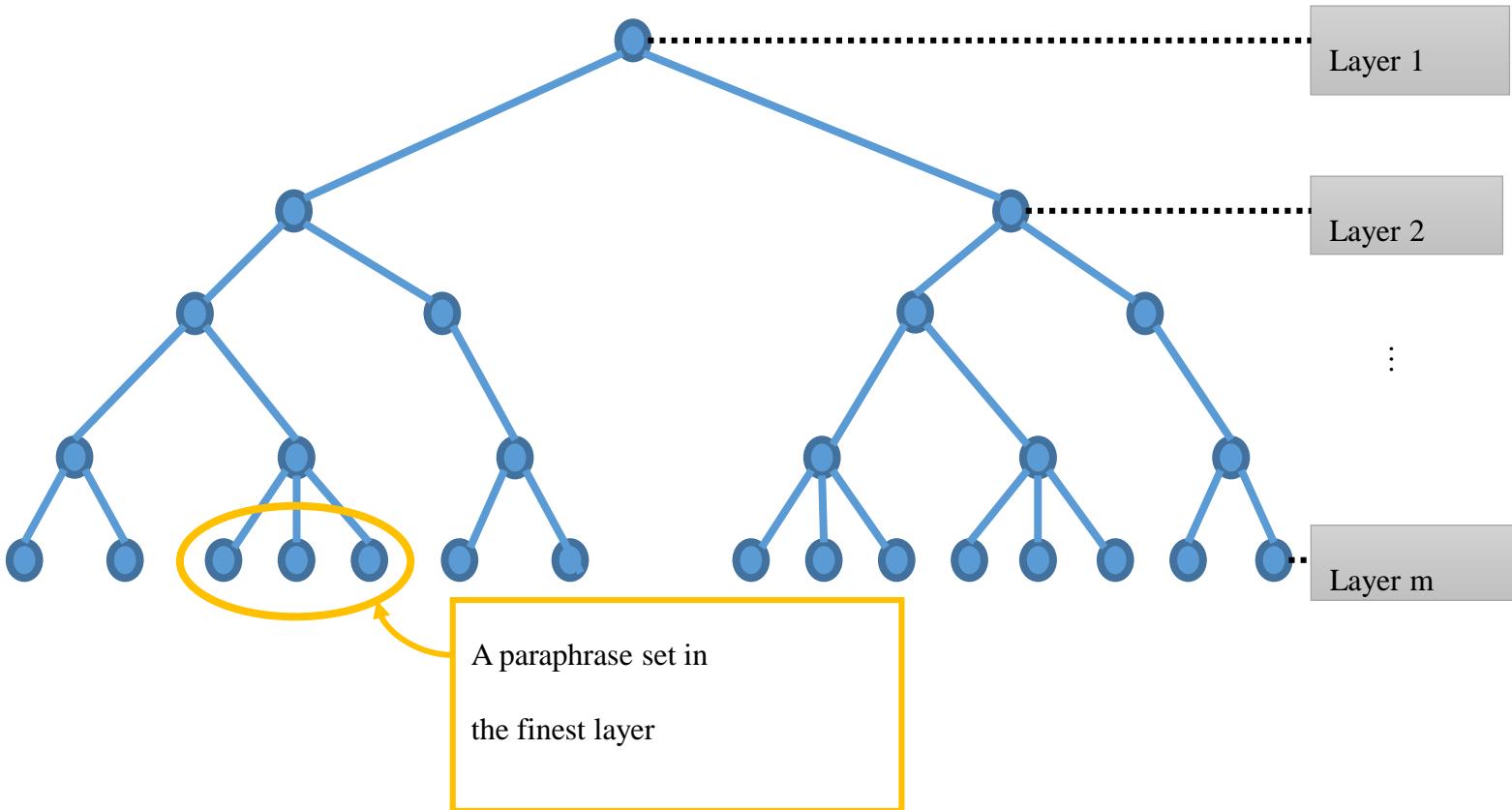
$\hat{q}_i \in \hat{Q}$: pre-trained word vector
(from GloVe, Word2Vec,...)
 $q_i \in Q$: retrofitted word vector



Structural-fitting (our proposed)

- Fine2coarse
 - $(\text{automobile}, \text{car})$
 - $(\text{automobile}, \text{car}, \text{wagon})$
- Coarse2fine
 - $(\text{automobile}, \text{car}, \text{wagon})$
 - $(\text{automobile}, \text{car})$

- Tongyici Cilin 同義詞詞林



Results (English)

	50d						300d
	RG65	WS353	MEN	SL999	RW	MTurk	WS353
GloVe	59.5	49.6	65.2	26.5	34.0	61.9	60.1
r1-ppdb (xl)	68.9	51.5	68.6	39.9	35.7	65.1	63.2
r1-ppdb (m)	61.4	51.1	66.0	29.8	35.6	63.0	61.0
r1-ppdb (s)	61.8	51.0	66.2	29.2	34.9	62.9	61.0
fine2coarse (m-xl)	69.2	52.8	68.9	39.4	36.6	65.9	63.4
fine2coarse (s-xl)	69.3	53.0	69.0	39.2	36.2	65.8	63.8
coarse2fine (xl-m)	70.0	52.5	68.8	41.1	36.1	65.7	63.4
coarse2fine (xl-s)	70.3	52.5	68.9	41.0	35.8	65.8	63.8
fine2coarse (s-m-xl)	68.4	53.3	68.7	37.8	36.5	65.6	64.0
coarse2fine (xl-m-s)	69.9	52.7	68.8	41.3	36.1	65.8	63.4

Spearman ($\rho \times 100$) correlation on six English semantic relatedness datasets

GloVe: Pennington, Socher, & Manning, 2014
r1: Faruqui et al., 2015

Results (Chinese)

	50	100	200	300
GloVe	48.8	50.5	49.9	50.5
r1-TongyiciCilin	54.3	54.4	53.1	53.0
fine2coarse 2 layers	55.5	55.2	54.2	53.4
fine2coarse 3 layers	55.3	55.7	55.5	54.9
coarse2fine 2 layers	55.0	54.3	52.7	52.9
coarse2fine 3 layers	55.2	54.4	53.6	53.4

($\rho \times 100$) of CWS297 of Chinese structural-fitting

討論之一

- 領域知識對於自然語言處理很重要
- 領域的轉移對於系統效能有很大的影響
- 資源在以機器學習/深度學習為基礎的自然語言處理扮演重要角色
- 稀少資源或有限資源機器學習/深度學習的研究，資料和知識的整合是重要趨勢

討論之二

- 語言陳述可能含主觀、誇大、不實、諷刺、隱喻、雙關語...等資訊
 - 可倍速啟動不老基因，延長皮膚細胞的青春生命力
 - 點餐都要等半小時，服務還真是好阿
 - 到油麻地地鐵站只要兩分鐘
 - 冬天：能穿多(匁ㄨㄛ)少穿多少；夏天：能穿多 (匁ㄨㄛ')少穿多(匁ㄨㄛ')少。
 - 太陽生下小太陽，要如何祝賀？答：生日快樂
- 語言理解不僅和語言相關，還可能需要用到世界知識、常識、...等外部知識
- 建立人類常識、世界知識、因果關係、動作過程，或是整合文字、視覺、聽覺等資訊的多模態知識庫，呈現知識表達和推理

World-Wide Web Resources

- The Association for Computational Linguistics site (the major international organization in the field)
<http://www.aclweb.org> [acl]
- The ACL Anthology (A Digital Archive of Research Papers in Computational Linguistics)
<https://www.aclweb.org/anthology/> [anthology]
- The Linguistic Data Consortium (create, collect and distribute speech and text databases, lexicons, and other resources for research and development purposes)
<http://www.ldc.upenn.edu/>

Major Publications

- COMPUTATIONAL LINGUISTICS
- Transactions of ACL
- COMPUTER SPEECH & LANGUAGE
- MACHINE TRANSLATION
- SPEECH TECHNOLOGY
- JOURNAL OF NATURAL LANGUAGE ENGINEERING
- JOURNAL OF LOGIC, LANGUAGE AND INFORMATION
- ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)
- Artificial Intelligence

Professional Organizations, Associations

- Association for Computational Linguistics (ACL)
- Association for the Advancement of Artificial Intelligence (AAAI)
- The Association for Computational Linguistics and Chinese Language Processing (ROCLING)

Major Conferences

- Annual Meeting of Association for Computational Linguistics (ACL)
- International Conference on Computational Linguistics (COLING)
- EACL, NAACL, ACL, IJCNLP, CoNLL, IWPT, TMI, ICSLP, Interspeech, AAAI, IJCAI, SIGIR, ROCLING, ...

Evaluation Competitions

- Message Understanding Conference ([**MUC**](#))
 - named entity categorization
 - word sense disambiguation
 - mini-MUC (contents scanning, template filling)
 - co-reference identification
 - predicate-argument identification
- Document Understanding Conference ([**DUC**](#))
 - Automatic Summarizing Evaluation
- Text Retrieval Conference ([**TREC**](#))
 - Information retrieval using NLP/statistical techniques
- SemEval ([**EVAL**](#))
 - Evaluating Word Sense Disambiguation Systems

NLP Toolkits

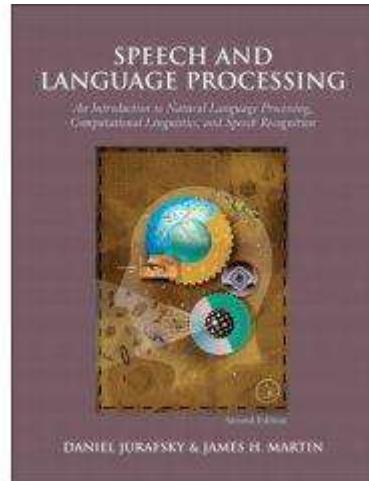
- NLTK (<http://www.nltk.org/>)
- The Stanford CoreNLP Natural Language Processing Toolkit (<http://nlp.stanford.edu/software/>)
- OpenNLP (<http://opennlp.apache.org/>)
- DeepNL (Deep Learning for Natural Language Processing) (<https://github.com/attardi/deepnl>)

Natural Language Toolkit

- Open source Python modules, linguistic data and documentation for research and development in natural language processing
- Code - functionality provided by NLTK in over 100,000 lines of Python code, distributed under the Apache License
- Data - ~60 corpora, grammar collections, and trained models that come with NLTK

Text/Reference Books

- Daniel Jurafsky and James H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, Second Edition. Prentice Hall, 2008



<http://www.cs.colorado.edu/~martin/slp2.html>

Text/Reference Books

- Speech and Language Processing (3rd ed. draft)
- <https://web.stanford.edu/~jurafsky/slp3/>
- Draft chapters in progress, October 16, 2019
- https://web.stanford.edu/~jurafsky/slp3/edbook_oct162019.pdf

- 2: [Regular Expressions, Text Normalization, and Edit Distance](#)
- 3: [Language Modeling with N-Grams](#)
- 4: [Naive Bayes Classification and Sentiment](#)
- 5: [Logistic Regression](#)
- 6: [Vector Semantics and Embeddings](#)
- 7: [Neural Nets and Neural Language Models](#)
- 8: [Part-of-Speech Tagging](#)
- 9: [Sequence Processing with Recurrent Networks](#)
- 10: [Encoder-Decoder Models, Attention, and Contextual Embeddings](#)
- 11: Machine Translation

Text [pptx] [pdf]	Edit Distance [pptx] [pdf]	[Ch. 2 and parts of Ch. 3 in 2nd ed.]
LM [pptx] [pdf]		[Ch. 4 in 2nd ed.]
NB [pptx] [pdf]		[new in this edition]
Sentiment [pptx] [pdf]		
Vector1 [pptx] [pdf]	Vector2 [pptx] [pdf]	
		[new in this edition]
		[Ch. 5 in 2nd ed.]
		[new in this edition]
		[new in this edition]

- 12: [Constituency Grammars](#)
- 13: [Constituency Parsing](#)
- 14: [Statistical Constituency Parsing](#)
- 15: [Dependency Parsing](#)

[Ch. 12 in 2nd ed.]
[Ch. 13 in 2nd ed.]
[Ch. 14 in 2nd ed.]
[new in this edition]

- 16: [Logical Representations of Sentence Meaning](#)
- 17: Computational Semantics and Semantic Parsing
- 18: [Information Extraction](#)
- 19: [Word Senses and WordNet](#)
- 20: [Semantic Role Labeling and Argument Structure](#)
- 21: [Lexicons for Sentiment, Affect, and Connotation](#)

SRL [[pptx](#)] [[pdf](#)]
Select [[pptx](#)] [[pdf](#)]
SentLex [[pptx](#)] [[pdf](#)]

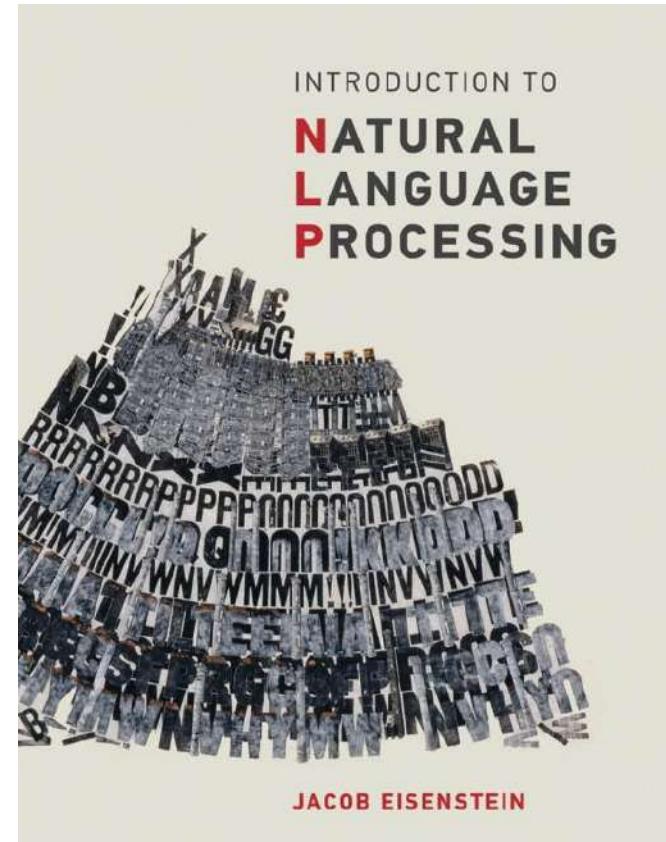
[expanded from parts of Ch. 19, 20 in 2nd ed]
[new in this edition]

- 22: [Coreference Resolution](#)
- 23: [Discourse Coherence](#)

[expanded from parts of Ch 21 in 2nd ed]
[expanded from parts of Ch 21 in 2nd ed]

Text/Reference Books

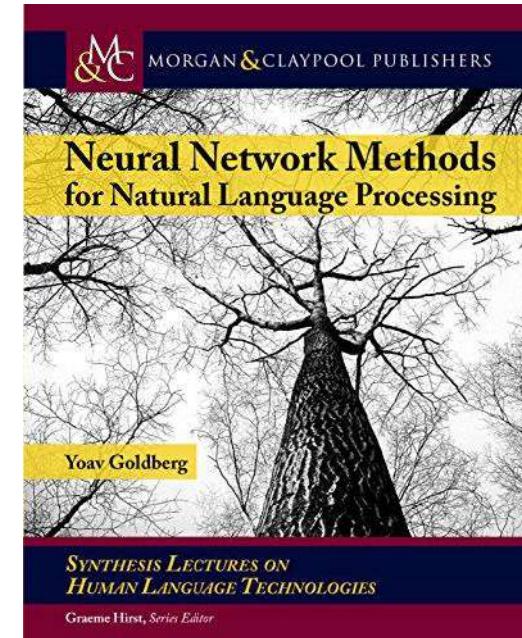
- Jacob Eisenstein, Introduction to Natural Language Processing, The MIT Press, 2019.



6	Language Models 119	
6.1	<i>N</i> -Gram Language Models 120	
6.2	Smoothing and Discounting 122	
6.3	Recurrent Neural Network Language Models 127	
6.4	Evaluating Language Models 132	
6.5	Out-of-Vocabulary Words 134	
7	Sequence Labeling 137	
7.1	Sequence Labeling as Classification 137	
7.2	Sequence Labeling as Structure Prediction 139	
7.3	The Viterbi Algorithm 140	
7.4	Hidden Markov Models 145	
7.5	Discriminative Sequence Labeling with Features 149	
7.6	Neural Sequence Labeling 158	
7.7	*Unsupervised Sequence Labeling 161	
8	Applications of Sequence Labeling 167	
8.1	Part-of-Speech Tagging 167	
8.2	Morphosyntactic Attributes 173	
8.3	Named Entity Recognition 175	
8.4	Tokenization 176	
8.5	Code Switching 177	
8.6	Dialogue Acts 178	
9	Formal Language Theory 183	
9.1	Regular Languages 184	
9.2	Context-Free Languages 198	
9.3	*Mildly Context-Sensitive Languages 209	
10	Context-Free Parsing 215	
10.1	Deterministic Bottom-Up Parsing 216	
10.2	Ambiguity 219	
10.3	Weighted Context-Free Grammars 222	
10.4	Learning Weighted Context-Free Grammars 227	
10.5	Grammar Refinement 231	
10.6	Beyond Context-Free Parsing 238	
11	Dependency Parsing 243	
11.1	Dependency Grammar 243	
11.2	Graph-Based Dependency Parsing 248	
11.3	Transition-Based Dependency Parsing 253	
12	Logical Semantics 269	
12.1	Meaning and Denotation 270	
12.2	Logical Representations of Meaning 270	
12.3	Semantic Parsing and the Lambda Calculus 274	
12.4	Learning Semantic Parsers 280	
13	Predicate-Argument Semantics 289	
13.1	Semantic Roles 291	
13.2	Semantic Role Labeling 295	
13.3	Abstract Meaning Representation 302	
14	Distributional and Distributed Semantics 309	
14.1	The Distributional Hypothesis 309	
14.2	Design Decisions for Word Representations 311	
14.3	Latent Semantic Analysis 313	
14.4	Brown Clusters 315	
14.5	Neural Word Embeddings 317	
14.6	Evaluating Word Embeddings 322	
14.7	Distributed Representations beyond Distributional Statistics 324	
14.8	Distributed Representations of Multiword Units 327	
15	Reference Resolution 333	
15.1	Forms of Referring Expressions 334	
15.2	Algorithms for Coreference Resolution 339	
15.3	Representations for Coreference Resolution 348	
15.4	Evaluating Coreference Resolution 353	
16	Discourse 357	
16.1	Segments 357	
16.2	Entities and Reference 359	
16.3	Relations 362	
IV	APPLICATIONS 377	
17	Information Extraction 379	
17.1	Entities 381	
17.2	Relations 387	
17.3	Events 395	
17.4	Hedges, Denials, and Hypotheticals 397	
17.5	Question Answering and Machine Reading 399	
18	Machine Translation 405	
18.1	Machine Translation as a Task 405	
18.2	Statistical Machine Translation 410	
18.3	Neural Machine Translation 415	
18.4	Decoding 423	
18.5	Training toward the Evaluation Metric 424	
19	Text Generation 431	
19.1	Data-to-Text Generation 431	
19.2	Text-to-Text Generation 437	
19.3	Dialogue 440	

Text/Reference Books

- Neural Network Methods in Natural Language Processing (Synthesis Lectures on Human Language Technologies)



Yoav Goldberg, A Primer on Neural Network Models for
Natural Language Processing, Journal of Artificial Intelligence
Research 57 (2016) 345–420

Major Topics

- Word
 - Sequence
 - Syntax 老師
 - Meaning
 - Discourse
 - Applications 學生
-

Course Outlines

- Collocation, Word, Multi-Word Expression
- N-Gram Language Models, Smoothing and Discounting
- Neural Network Language Models
- Hidden Markov Models and Part of Speech Tagging
- Syntax, Grammars and Parsing
- Statistical Parsing
- Deep Neural Network for Parsing
- Sequence-to-Sequence Model
- Dependency Parsing

Course Outlines

- Lexical Semantics
- Semantic Role Labelling and Argument Structure
- Representation of Sentence Meanings
- Computational Discourse
- Chinese Discourse Parsing

Important Information

- TAs
 - 魏聖倫、黃國祐
 - 33664888 ext 301
- Grading
 - Midterm Examination
 - Term Examination
 - 1 Term Project
 - 1 Oral Presentation
- NLP web site
 - https://ceiba.ntu.edu.tw/1082CSIE5042_nlp

Note

- 1. 中間下課抽加籤單
- 2. 學生報告時，提問列入評分標準，且要在 office hour 時找助教討論
- 3. 四月之前會給報告的主題
- 4. term project 不做口頭報告

Lecture 2: Words, Collocations and Multiword Expressions

Outlines

- Granularity
 - Words
 - Collocations
 - Multiword Expressions
 - N-grams (Lecture 3)
- Approaches
 - Counting-Based (Statistics-based) Approach
 - Prediction-Based (Neural Network) Approach

斷詞(分詞)問題

- 詞的定義
 - 語句中具有完整概念且能獨立自由運用的基本單位為詞
- 中文詞彙
 - 中文句子詞與詞之間並沒有明顯的分隔記號。
 - 這名記者會說國語。
 - 這名記者會說國語。
 - 這名記者會說國語。
- 分詞標準
 - 中國【信息處理用現代漢語分詞規範】
 - GB/T 13715-92
 - 1992年10月4日國家技術監督局
 - 台灣【中文資訊處理分詞規範】
 - 經濟部中央標準局印行
 - CNS14366
 - 1999 中文分詞原則正式通過為國家標準

1.1 主题内容

本规范规定了现代汉语的分词原则,以满足信息处理的需要。它对汉语信息处理的规范化,对各种汉语信息处理系统之间的兼容性有重要的作用。

1.2 适用范围

本规范适用于汉语信息处理各领域,其他行业和有关学科可以参考使用。

汉语信息处理各领域可以根据其专门需求,进一步补充和细化本规范的规定。

2 引用标准

GB 12200 汉语信息处理词汇

3 术语

以下术语引自 GB 12200。

3.1 汉语信息处理

用计算机对汉语的音、形、义等信息进行的处理。

3.2 词

最小的能独立运用的语言单位。

3.3 词组

由两个或两个以上的词,按一定的语法规则组成,表达一定意义的语言单位。

3.4 分词单位

汉语信息处理使用的、具有确定的语义或语法功能的基本单位。它包括本规范的规则限定的词和词组。

3.5 汉语分词

从信息处理需要出发,按照特定的规范,对汉语按分词单位进行划分的过程。

4 概述

本规范以信息处理应用为目的,根据现代汉语的特点及规律,规定现代汉语的分词原则。

中文資訊處理分詞規範

1. 適用範圍：本標準規定資訊處理用分詞標準研擬草案之原則、層次劃分及應用實例，並且經後處理之後適用於中文資訊檢索、機器翻譯和文句校正等。
2. 用語釋義
 - (1) 分詞標準：根據語言學的資訊處理為著眼點，規範中文字串基本語意單位切分的遵循標準。
 - (2) 定義：用來說明中文分詞的基本單位。
 - (3) 基本原則：中文分詞標準的一般性原則，從語意、語法兩方面來規範，符合語言學理論。
 - (4) 輔助原則：中文分詞的操作型原則，輔助原則可因需要而有變異性。
 - (5) 層次劃分：依自動化處理的難易，規定不同階段應達成的分詞目標。
 - (6) 信級：在本規範的前提下，所定的操作原則，依標準詞典將詞切分，用來作基本資料交換。
 - (7) 達級：在本規範的前提下，所定的操作原則，是簡單構詞規律所能組合成詞的層次，用於大部分自然語言處理。

- (8) 雅級：在本規範的前提下，所定的操作原則，是符合語言學理論的理想階層，可用於剖析。
- (9) 操作型定義：在本規範的前提下，依詞類、結構等，列出各類型詞語，供使用者實際操作時的依據。
- (10) 標準辭典：是理想中的辭典，辭典收納的詞能符合分詞標準，並能與時俱增，隨時更新編入語言演變產生的新詞。
- (11) 附著語素：有獨立意義卻無法獨立扮演一個語法功能的語言成分。
- (12) 衍生詞綴：具有衍生性的附著語素，可由構詞律組成複合詞，對資訊處理的困難度不大，在達級即可達成組合詞的目標。
- (13) 語法詞綴：具有固定獨立的語法功能，且不影響緊鄰成分語法類別的詞綴，包括「了、著、過、看、看看、們、者」等。
- (14) 接頭詞：附加於別的成分之前構成複合詞的詞，在雅級處理其組合。如「準一、多一、非一」。
- (15) 接尾詞：附加於別的成分之後構成複合詞的詞，在雅級處理其組合。如「一盃、一盒、一觀」。

SIGHAN Chinese Language Processing Bakeoff

Source	Encodings	Training (Wds/Types)	Test (Wds/Types)
CITYU	BIG5HKSCS/Unicode	1.6M/76K	220K/23K
CKIP	BIG5/Unicode	5.5M/146K	91K/15K
LDC	Unicode	632K (est. wds)	61K (est. wds)
MSRA	GB18030/Unicode	1.3M/63K	100K/13K
UPUC	GB/Unicode	509K/37K	155K/17K

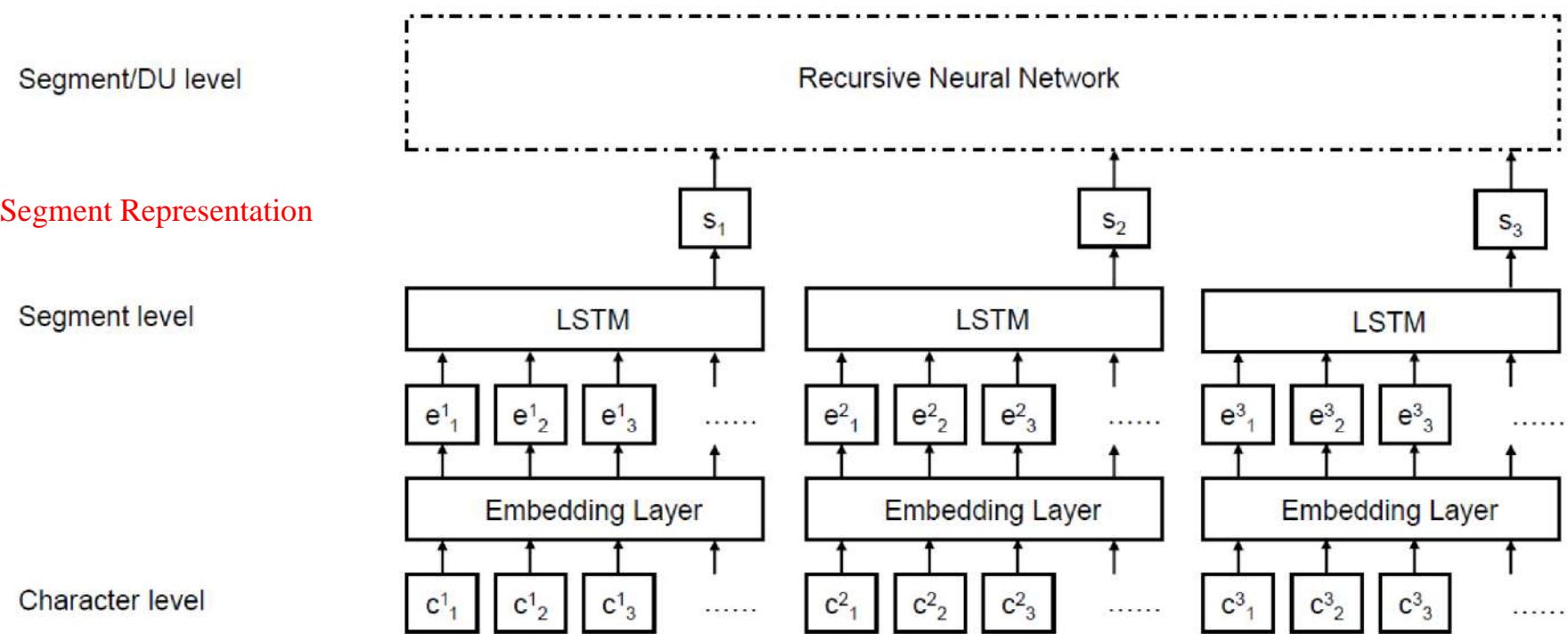
SIGHAN Word Segmentation Bakeoff 6 Dataset

Neural Models	PKU	MSR	Non-neural Model	PKU	MSR
Zheng et al. [22]	92.8	93.9	Zhang and Clark [21]	95.1	97.2
Pei et al. [13]	95.2	97.2	Sun et al. [15]	95.4	97.4
Ma and Hinrichs [10]	95.1	96.6	Zhang et al. [19]	96.1	97.4
Cai and Zhao [3]	95.5	96.5			
Zhang et al. [20]	95.7	97.7			
Liu et al. [9]	95.7	97.6			
<i>Our models</i>					
DCN	95.9	97.7	MT-DCN	96.4	97.8

斷詞系統

- Jieba 結巴
 - <https://github.com/fxsjy/jieba>
- CkipTagger
 - <https://github.com/ckiplab/ckiptagger>
- 斷與不斷

Architecture of NTU Discourse Parser



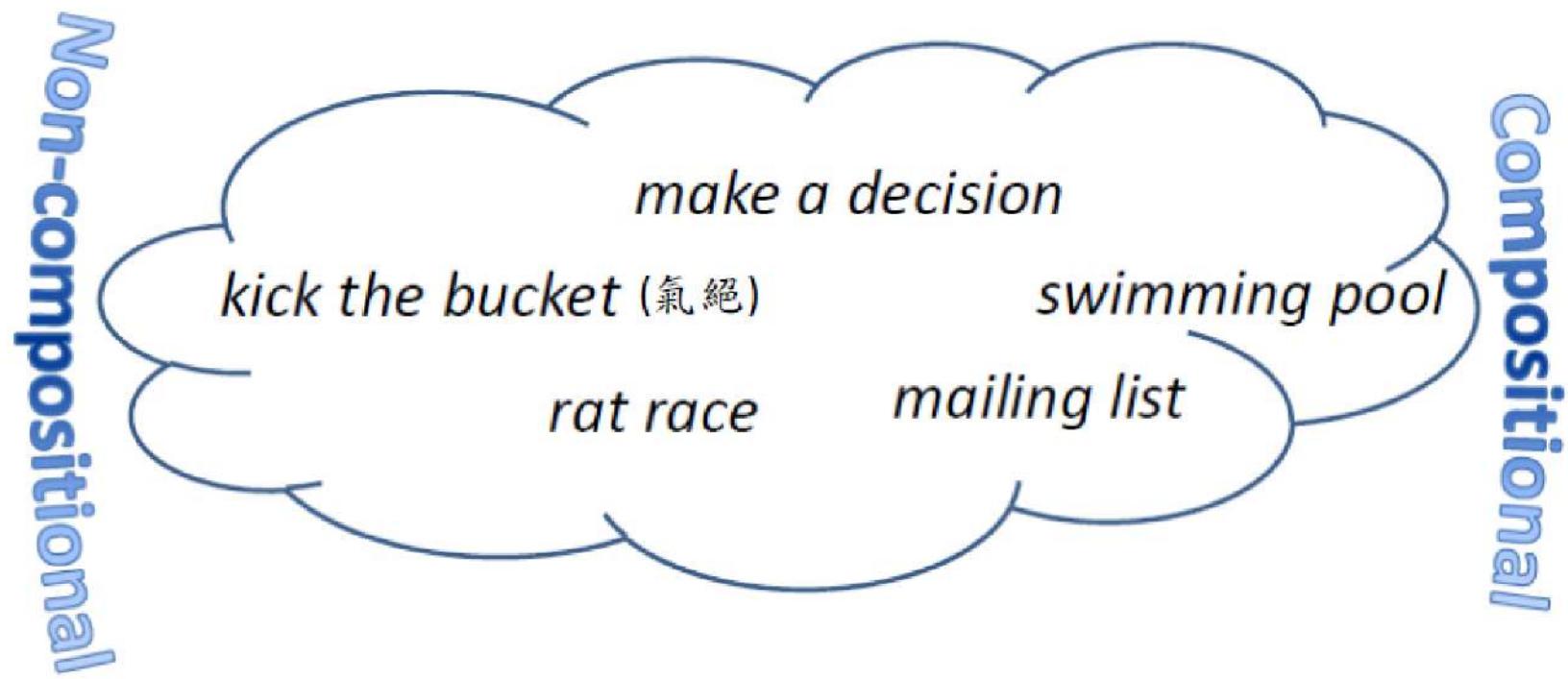
(Li, Huang, and Chen, COLING2018)

Collocation (搭配詞)

- Collocations consist of two or more words with limited *compositionality*.
- Term extraction
 - *collocations* and *terms*, *technical term* and *terminological phrase*.
- *strong* cigarettes, tea, coffee versus *powerful* drug
- Applications
 - natural language generation
 - computational lexicography
 - Parsing, ...

Compositionality

- Whether the meaning of a MWE can be determined from its components



Definition

- collocation (搭配詞)
 - a sequence of two or more consecutive words
 - cannot be derived directly from the meaning or connotation of its components
- *strong tea*
 - basic sense: having great physical strength
 - meaning in *strong tea*: rich in some active agent

Other Definitions/Notions

- Collocations are **not necessarily adjacent**
- Typical criteria for collocations
 - non-compositionality
 - non-substitutability
 - non-modifiability
- Collocations cannot be translated into other languages. green house → 溫室 (O) 綠色的房子 (X)
- strong association of words, but not necessarily fixed occurrence

Linguistic Subclasses of Collocations

- Light verbs (輕動詞)
 - verbs with little semantic content
 - *The man took a walk.* vs. *The man took a radio*
- Verb particle (語助詞) constructions/
Phrasal Verbs (詞組動詞, 短語動詞)
 - phrasal verb=verb + preposition
 - Create a meaning different from the original verb
 - take in → deceive
- Proper Nouns/Names
- Terminological Expressions

Multiword Expression (MWE)

- MWE
 - Expressions with at least 2 words
 - syntactically and/or semantically idiosyncratic (有特性) in nature
- A single unit at some level of linguistic analysis
 - idioms: kick the bucket (氣絕)
 - compound nouns: telephone box and post office
 - verb-particle constructions: look sth. up
 - proper names: San Francisco

Classification of MWEs

- Fixed expressions
 - in short, by and large, every which way (O)
 - in shorter or in very short (X)
- Semi-fixed expressions
 - non-decomposable idioms
 - kick the bucket (O)
 - he kicks the bucket (O)
 - the bucket was kicked (X)
 - compound nominals
 - car park, car parks
 - Proper names

Classification of MWEs

- Syntactically-Flexible Expressions
 - decomposable idioms
 - let the cat out of the bag (將祕密公開，洩漏秘密)
 - verb-particle constructions
 - light verbs
- Institutionalized Phrases (習慣用法)
 - salt and pepper
 - traffic light
 - kindle excitement (點燃激情)

Workshops on Multiword Expressions

- 2020
 - COLING Joint Workshop on Multiword Expressions and Electronic Lexicons (MWE-LEX 2020) – September 14, 2020 – Barcelona, Spain.
- 2019
 - ACL Joint Workshop on Multiword Expressions and WordNet – 1st or 2nd August 2019 – Florence, Italy.
- 2018
 - COLING Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions – August 25-26, 2018 – Santa Fe, USA.
- 2017
 - EACL 13th Workshop on Multiword Expressions – April 3 or 4, 2017 – Valencia, Spain.
- 2016
 - ACL 12th Workshop on Multiword Expressions – August 11, 2016 – Berlin, Germany.
- ...
- 2003

Counting-Based Collocation Detecting Techniques

- *Frequency*
- *Mean and Variance* of the distance between focal word (焦點詞) and collocating word (搭配詞)
- *Hypothesis Testing*
- *Mutual Information*

Reference Corpus

- New York Times
 - 115MB of text, 14M words
- Examples drawn from bigrams

Frequency

1. Selecting the most frequently occurring bigrams (2-23) ← quantitative technique
2. Passing the results through a part-of-speech filter (2-24) ← linguistic knowledge
3. Simple method that works very well.

pairs of
function
words
(功能詞)

$C(w^1 w^2)$	w^1	w^2
80871	of	the
58841	in	the
26430	to	the
21842	on	the
21839	for	the
18568	and	the
16121	that	the
15630	at	the
15494	to	be
13899	in	a
13689	of	a
13361	by	the
13183	with	the
12622	from	the
11428	New	York
10007	he	said
9775	as	a
9231	is	a
8753	has	been
8573	for	a

Selecting the most frequently occurring bigrams is not always significant.

Table 5.1 Finding Collocations: Raw Frequency. $C(\cdot)$ is the frequency of something in the corpus.

Keep those patterns that are likely to be “phrases”.

Tag Pattern	Example
A N	<i>linear function</i>
N N	<i>regression coefficients</i>
A A N	<i>Gaussian random variable</i>
A N N	<i>cumulative distribution function</i>
N A N	<i>mean squared error</i>
N N N	<i>class probability function</i>
N P N	<i>degrees of freedom</i>

Table 5.2 Part of speech tag patterns for collocation filtering. These patterns were used by Justeson and Katz to identify likely collocations among frequently occurring word sequences.

A: adjective, P: preposition, N: noun

stop list to exclude words whose frequent tag is not a verb, noun, or adjective

	$C(w^1 w^2)$	w^1	w^2	Tag Pattern
not regarded as non-compositional phrases	11487	New	York	A N
	7261	United	States	A N
	5412	Los	Angeles	N N
	3301	last	year	A N
	3191	Saudi	Arabia	N N
	2699	last	week	A N
	2514	vice	president	A N
	2378	Persian	Gulf	A N
	2161	San	Francisco	N N
	2106	President	Bush	N N
	2001	Middle	East	A N
	1942	Saddam	Hussein	N N
	1867	Soviet	Union	A N
	1850	White	House	A N
	1633	United	Nations	A N
	1337	York	City	N N
	1328	oil	prices	N N
	1210	next	year	A N
	1074	chief	executive	A N
	1073	real	estate	A N

Most of them are non-compositional phrases.

Can be filtered out by the longest sequence

Issue: Predicting the Compositionalty of Multiword Expressions

Table 5.3 Finding Collocations: Justeson and Katz' part-of-speech filter

w	$C(\text{strong}, w)$	w	$C(\text{powerful}, w)$
support	50	force	13
safety	22	computers	10
sales	21	position	8
opposition	19	men	8
showing	18	computer	8
sense	18	man	7
message	15	symbol	6
defense	14	military	6
gains	13	machines	6
evidence	13	country	6
criticism	13	weapons	5
possibility	11	post	5
feelings	11	people	5
demand	11	nation	5
challenges	11	forces	5
challenge	11	chip	5
case	11	Germany	5
supporter	10	senators	4
signal	9	neighbor	4
man	9	magnet	4
force	4		

Table 5.4 The nouns w occurring most often in the patterns '*strong w*' and '*powerful w*.'

Mean and Variance

- Many collocations consist of two words in more flexible relationships.
 - She **knocked** on his **door**.
 - They **knocked** at the **door**.
 - 100 women **knocked** on Donaldson's **door**.
 - A man **knocked** on the metal front **door**.
- How to extend bigrams to **bigrams at a distance**?

- (1) Define a collocational window (e.g., 3-4 words on each side)
- (2) Enter every word pair a collocational bigram

Sentence: *Stocks crash as rescue plan teeters*

Bigrams: *stocks crash stocks as stocks rescue*
crash as crash rescue crash plan
as rescue as plan as teeters
rescue plan rescue teeters
plan teeters

Figure 5.1 Using a three word collocational window to capture bigrams at a distance.

Mean and Variance

- Compute the mean and variance of the offset (偏移 : **signed distance**) between the two words in the corpus.
 - She **knocked** on his **door**. (Donaldson ‘ s)
 - They **knocked** at the **door**.
 - 100 women **knocked** on Donaldson’s **door**.
 - A man **knocked** on the metal front **door**.

$$\frac{1}{4}(3 + 3 + 5 + 5) = 4.0$$

-3 for *the door that she knocked on*

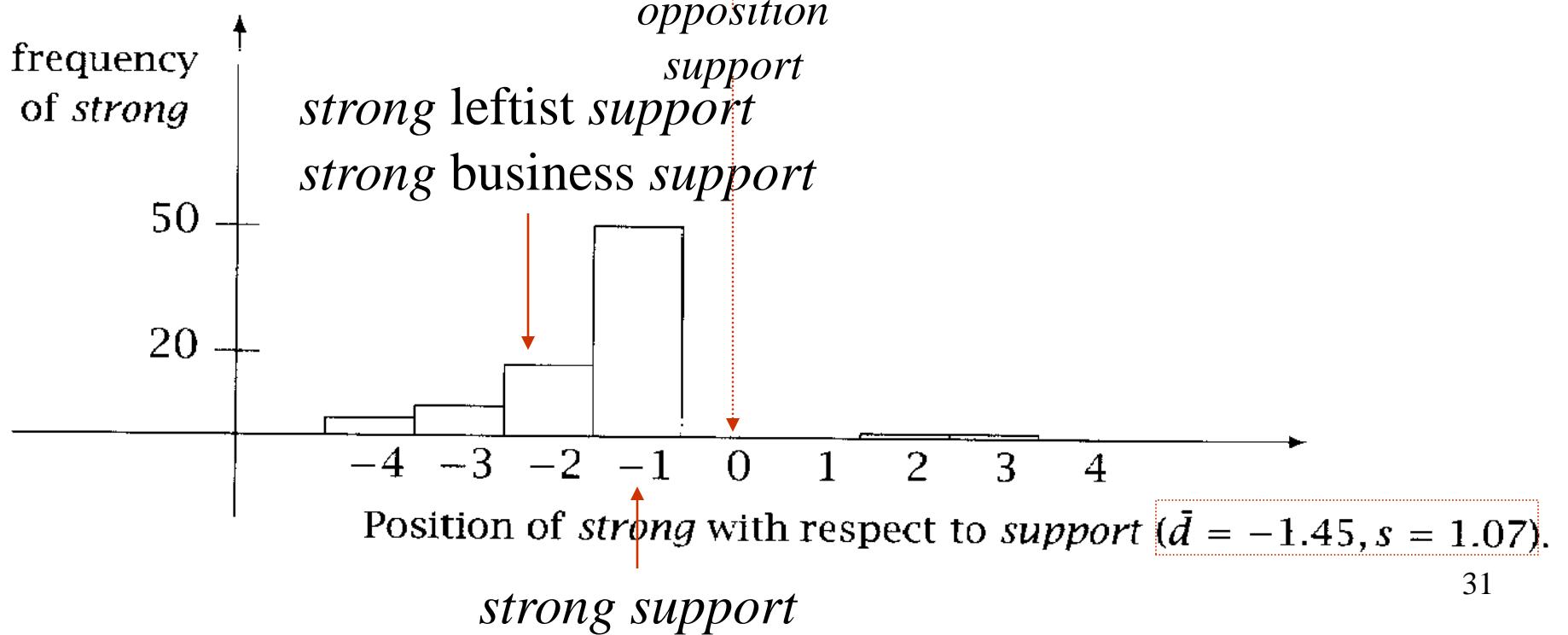
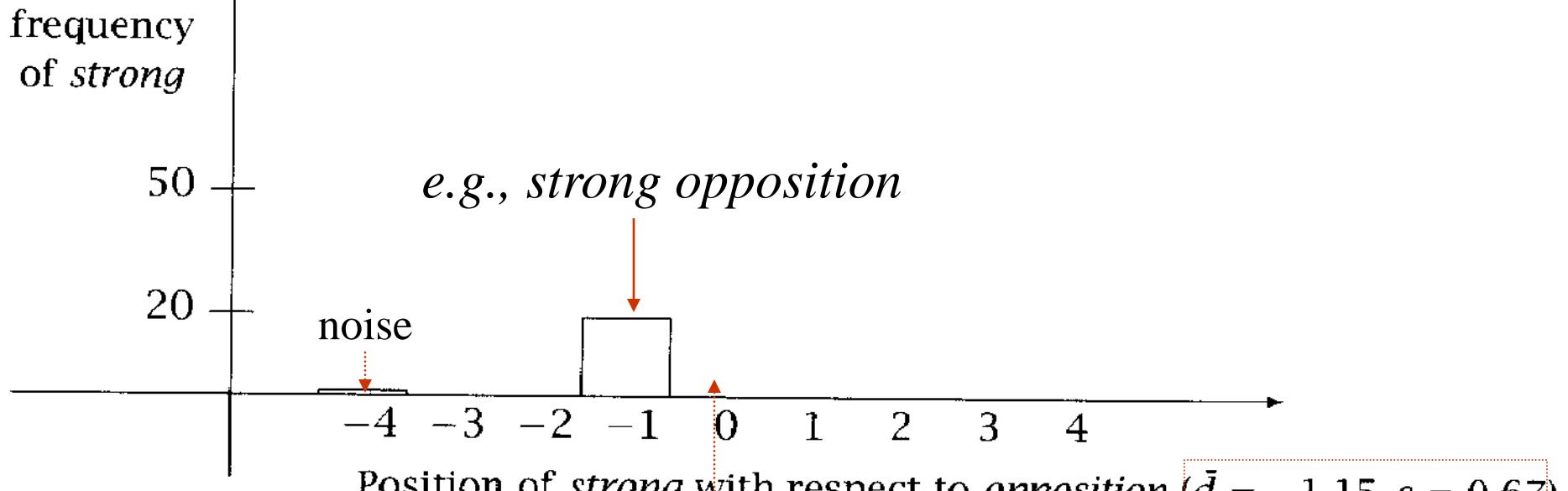
Mean and Variance

- Variance
 - how much the individual offsets deviate from the mean

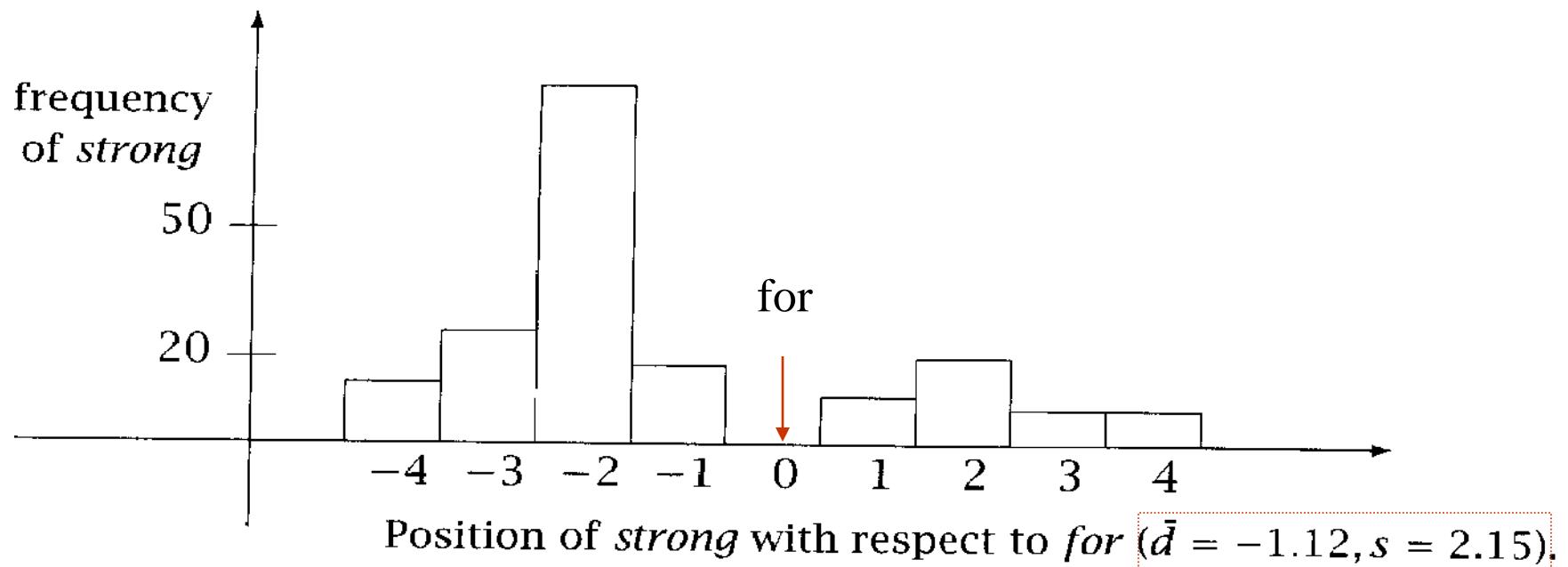
$$s^2 = \frac{\sum_{i=1}^n (d_i - \bar{d})^2}{n - 1}$$

$$s = \sqrt{\frac{1}{3} ((3 - 4.0)^2 + (3 - 4.0)^2 + (5 - 4.0)^2 + (5 - 4.0)^2)} \approx 1.15$$

- Interpretation of standard deviation
 - 0: the two words always occur at **exactly** the same distance.
 - **Low**: the two words usually occur at **about the same** distance.
 - High: the two words occur by chance.
- Find the **pairs** with **low** deviation



If the offsets are randomly distributed (i.e., no collocation), then the variance/sample deviation will be high.



for and *strong* don't form interesting collocations

	s	\bar{d}	Count	Word 1	Word 2	
low	0.43	0.97	11657	New	York	distance: 1
	0.48	1.83	24	previous 10/15	games	distance: 2
	0.15	2.98	46	minus 2 percentage	points	distance: 3
	0.49	3.87	131	hundreds	dollars	distance: 4
no interesting relationship	4.03	0.44	36	editorial	Atlanta	
	4.03	0.00	78	ring	New	
	3.96	0.19	119	point	hundredth	of millions of
	3.96	0.29	≈ 0	subscribers	by	
medium	1.07	1.45	80	strong	support	distance: 1,2
	1.13	2.57	7	powerful	organizations	distance: 3,4
	1.01	2.00	112	Richard	Nixon	distance: 2,3
	1.05	0.00	10	Garrison	said	distance: -1,1

normal distribution

Table 5.5 Finding collocations based on mean and variance. Sample deviation s and sample mean \bar{d} of the distances between 12 word pairs.

strong {business} support: strong support, strong business support

powerful {lobbying} organizations

Richard {M.} Nixon: Richard M Nixon, Richard M. Nixon

Garrison said/said Garrison

distance-1: how many words are inserted in between

Overview of Hypothesis Testing

- High frequency and low variance can be accidental.
 - whether the co-occurrence is random, or
 - whether it occurs more often than chance
 - This is a classical problem in Statistics called ***Hypothesis Testing***. 假設檢定
 - Formulate a ***null hypothesis*** H_0 (no association beyond chance occurrences) 虛無假設
 - Calculate the probability that a collocation would occur if H_0 were true, and then ***reject*** H_0 if p is too low and retain H_0 as possible, otherwise.
 - Two issues
 - Look for particular patterns in the data
 - How much data we have seen
- ↑
significance level of p <
0.05, 0.01, 0.005, or 0.001

Overview of Hypothesis Testing

- Null hypothesis
 - Each of the words w^1 and w^2 is generated completely **independently of the other**.
 - They are no association.
 - They are randomly distributed (normal distribution is adopted).
 - The chance of coming together is

$$P(w^1 w^2) = P(w^1)P(w^2)$$

The t test

- Look at the mean and variance of a sample of measurements, where **the null hypothesis is that the sample is drawn from a distribution with mean μ .** μ . ← expected mean
Look at **the difference between the observed and expected means, scaled by the variance of the data.**
 - Idea: how likely one is to get a sample of that mean and variance assuming that the sample is drawn from a normal distribution with mean μ .
- Regard the text corpus as a long sequence of N bigrams.

觀測值
期望值

The t test

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}}$$

- \bar{x} is the sample mean
- s^2 is the sample variance
- N is the sample size
- μ is the mean of distribution

If the t statistic is large enough, then we can reject the null hypothesis (null means no difference).

Example

- a text corpus has 14,307,668 tokens
- *new* occurs 15,828 times
- *companies* occurs 4,675 times
- *new companies* occur 8 times
- Does “new companies” form a collocation?
- What is the sample that we are measuring the mean and variance?

- *new* occurs 15,828 times, *companies* 4,675 times, and 14,307,668 tokens

$$P(\text{new}) = \frac{15828}{14307668}$$

$$P(\text{companies}) = \frac{4675}{14307668}$$

maximum likelihood estimate

- Null hypothesis: the occurrences of *new* and *companies* are independent, i.e., they are in no relationship

$$H_0 : P(\text{new}\text{companies}) = P(\text{new})P(\text{companies}) =$$

$$\frac{15828}{14307668} \times \frac{4675}{14307668} \approx 3.615 \times 10^{-7} \text{ expected probability}$$

$$\mu = 3.615 \times 10^{-7}$$

Bernoulli trial

$$\sigma^2 = p(1-p) \approx p \text{ when } p \text{ is small}$$

- 8 occurrences of *new companies* among the 14,307,668 bigrams (observed probability)

$$\bar{x} = \frac{8}{14307668} \approx 5.591 \times 10^{-7} \quad \sigma^2 \approx p = 5.591 \times 10^{-7}$$

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}} = \frac{5.591 \times 10^{-7} - 3.615 \times 10^{-7}}{\sqrt{\frac{5.591 \times 10^{-7}}{14307668}}} \approx 0.999932$$

- Because $0.999932 < 2.576$, we cannot reject the null hypothesis. I.e., *new* and *companies* occur independently, and do not form a collocation.
- “new companies” is completely compositional, and there is no element of added meaning.

Reject the null hypothesis, so that they are good candidates for Significance.

A high portion of the occurrence of both words, or at least a very high portion of the occurrence of one of the words occurs in the bigram → t is high

t	$C(w^1)$	$C(w^2)$	$C(w^1 w^2)$	前伊朗領袖何梅尼 (Ayatollah Ruhollah Khomeini)	w^1	w^2
4.4721	$\min(42, 20) = 20$		$\leftrightarrow 20$	Ayatollah		Ruhollah
4.4721	41	27	$= 27 \leftrightarrow 20$	Bette		Midller
→ 4.4720	30	117	$= 30 \leftrightarrow 20$	Agatha		Christie
4.4720	77	59	$= 59 \leftrightarrow 20$	videocassette		recorder
4.4720	24	320	$= 24 \leftrightarrow 20$	unsalted		butter
2.3714	14907	9017	$= 9017 \leftrightarrow 20$	first		made
2.2446	13484	10570	$= 10570 \leftrightarrow 20$	over		many
→ 1.3685	14734	13478	$= 13478 \leftrightarrow 20$	into		them
1.2176	14093	14776	$= 14093 \leftrightarrow 20$	like		people
0.8036	15019	15629	$= 15019 \leftrightarrow 20$	time		last

Table 5.6 Finding collocations: The t test applied to 10 bigrams that occur with frequency 20.

Fail the test for significance, so that they are not good candidates for collocations

the same frequency
(frequency-based method fails)

Hypothesis testing of differences

- Find words whose co-occurrence patterns best distinguish between two words.
 - Differentiate the meanings of *strong* and *powerful*
- Application for lexicography
 - *strong* computer vs. *powerful* computer for a lexicographer

	t	$C(w)$	$C(\text{strong } w)$	$C(\text{powerful } w)$	Word
p o W e r f u l	3.1622	933	0	10	computers
	2.8284	2337	0	8	computer
	2.4494	289	0	6	symbol
	2.4494	588	0	6	machines
	2.2360	2266	0	5	Germany
	2.2360	3745	0	5	nation
	2.2360	395	0	5	chip
	2.1828	3418	4	13	force
	2.0000	1403	0	4	friends
	2.0000	267	0	4	neighbor
s t r o n g	7.0710	3685	50	0	support
	6.3257	3616	58	7	enough
	4.6904	986	22	0	safety
	4.5825	3741	21	0	sales
	4.0249	1093	19	1	opposition
	3.9000	802	18	1	showing
	3.9000	1641	18	1	sense
	3.7416	2501	14	0	defense
	3.6055	851	13	0	gains
	3.6055	832	13	0	criticism

Table 5.7 Words that occur significantly more often with *powerful* (the first ten words) and *strong* (the last ten words).

- The comparison of the means of two normal populations.
- The null hypothesis is that the average difference is 0 ($\mu=0$).

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}} \quad \therefore \quad \left\{ \begin{array}{l} \bar{x} - \mu = \bar{x} = \frac{1}{N} \sum (x_{1i} - x_{2i}) = \bar{x}_1 - \bar{x}_2 \\ \sigma^2(x_1, x_2) = \sigma^2(x_1) + \sigma^2(x_2) \end{array} \right.$$

$$\therefore t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$$

Assume a Bernoulli distribution.

$$\mu = p \quad \sigma^2 = p(1-p) \approx p \quad \text{when } p \text{ is small}$$

$$\bar{x}_1 = s_1^2 = P(v^1 w), \bar{x}_2 = s_2^2 = P(v^2 w)$$

where v^1 and v^2 are the words we are comparing

(e.g., *powerful* and *strong*), and

w is the collocate of interest (e.g., *computers*)

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \approx \frac{P(v^1 w) - P(v^2 w)}{\sqrt{\frac{P(v^1 w) + P(v^2 w)}{N}}} \approx \frac{\frac{C(v^1 w) - C(v^2 w)}{N}}{\sqrt{\frac{C(v^1 w) + C(v^2 w)}{N^2}}}$$

$$= \frac{C(v^1 w) - C(v^2 w)}{\sqrt{C(v^1 w) + C(v^2 w)}}$$

where $C(x)$ is the number of times x occurs in the corpus

	t	$C(w)$	$C(\text{strong } w)$	$C(\text{powerful } w)$	Word
p o W e r f u l	3.1622	933	0	10	computers
	2.8284	2337	0	8	computer
	2.4494	289	0	6	symbol
	2.4494	588	0	6	machines
	2.2360	2266	0	5	Germany
	2.2360	3745	0	5	nation
	2.2360	395	0	5	chip
	2.1828	3418	4	13	force
	2.0000	1403	0	4	friends
	2.0000	267	0	4	neighbor
s t r o n g	7.0710	3685	50	0	support
	6.3257	3616	58	7	enough
	4.6904	986	22	0	safety
	4.5825	3741	21	0	sales
	4.0249	1093	19	1	opposition
	3.9000	802	18	1	showing
	3.9000	1641	18	1	sense
	3.7416	2501	14	0	defense
	3.6055	851	13	0	gains
	3.6055	832	13	0	criticism

Table 5.7 Words that occur significantly more often with *powerful* (the first ten words) and *strong* (the last ten words).

Mutual Information

- An Information-Theoretic measure for discovering collocations
- Pointwise Mutual Information

$$I(x', y') = \log_2 \frac{P(x' y')}{P(x') P(y')}$$

$$= \log_2 \frac{P(x' | y')}{P(x')}$$

$$= \log_2 \frac{P(y' | x')}{P(y')}$$

random variables
vs.
values of random variables

t	$C(w^1)$	$C(w^2)$	$C(w^1 w^2)$	w^1	w^2
4.4721	42	20	20	Ayatollah	Ruhollah
4.4721	41	27	20	Bette	Midler
4.4720	30	117	20	Agatha	Christie
4.4720	77	59	20	videocassette	recorder
4.4720	24	320	20	unsalted	butter
2.3714	14907	9017	20	first	made
2.2446	13484	10570	20	over	many
1.3685	14734	13478	20	into	them
1.2176	14093	14776	20	like	people
0.8036	15019	15629	20	time	last

Table 5.6 Finding collocations: The t test applied to 10 bigrams that occur with frequency 20.

$$I(Ayatollah, Ruhollah) = \log_2 \frac{20}{\frac{14307668}{42} \times \frac{20}{14307668}} \approx 18.38$$

Mutual Information

- Pointwise Mutual Information is roughly a measure of how much one word tells us about the other.
- Pointwise mutual information works particularly badly in sparse environments.

Joint Entropy and Conditional Entropy

- The *joint entropy* of a pair of discrete random variables $X, Y \sim p(x,y)$ is the amount of information needed on average to specify both their values.

$$H(X, Y) = - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x, y)$$

- The *conditional entropy* of a discrete random variable Y given another X , for $X, Y \sim p(x,y)$, expresses how much extra information to supply on average to communicate Y given that the other party knows X .

$$\begin{aligned} H(Y | X) &= \sum_{x \in X} p(x) H(Y | X = x) \\ &= \sum_{x \in X} p(x) \left(- \sum_{y \in Y} p(y | x) \log p(y | x) \right) \\ &= - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(y | x) \end{aligned}$$

Mutual Information

- By the chain rule for entropy, we have
$$H(X, Y) = H(X) + H(Y/X) = H(Y) + H(X/Y)$$
- Therefore, $H(X) - H(X/Y) = H(Y) - H(Y/X)$
- This difference is called the *mutual information between X and Y.*
- Interpretations
 - Reduction in uncertainty of one random variable due to knowing about another
 - The amount of information one random variable contains about another

Sparseness is a particularly difficult problem for mutual information.

(1) Perfect dependence

$$I(x, y) = \log \frac{P(xy)}{P(x)P(y)} = \log \frac{P(x)}{P(x)P(y)} = \log \frac{1}{P(y)}$$

As the perfectly dependent bigrams *get rarer*, their mutual information *increases*. → bad measure of dependence

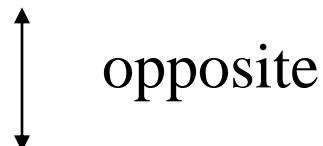
I.e., the score depends on the frequency of the individual words.

(2) Perfect independence

$$I(x, y) = \log \frac{P(xy)}{P(x)P(y)} = \log \frac{P(x)P(y)}{P(x)P(y)} = \log 1 = 0$$

→ good measure of independence

With MI, bigrams composed of low-frequency words will receive a higher score than bigrams composed of high-frequency words.



Higher frequency means more evidence and a higher rank for bigrams is preferred when we have more evidence

$$C(w^1 w^2) I(w^1 w^2)$$

Symbol	Definition	Current use	Fano
$I(x, y)$	$\log \frac{p(x, y)}{p(x)p(y)}$	pointwise mutual information	mutual information
$I(X; Y)$	$E \log \frac{p(X, Y)}{p(X)p(Y)}$	mutual information	average MI/expectation of MI

Table 5.17 Different definitions of *mutual information* in (Cover and Thomas 1991) and (Fano 1961).

Emotion Analysis

Changhua Yang, Kevin Hsin-Yih Lin and Hsin-Hsi Chen (2007). “Building Emotion Lexicon from Weblog Corpora.” *Proceedings of 45th Annual Meeting of Association for Computational Linguistics.*

Non-verbal Emotional Expressions

- Yahoo! Kimo Emotion Icon Set ([Emoticon](#))

符號	文字	意涵	符號	文字	意涵	符號	文字	意涵	符號	文字	意涵
	:)	微笑		:O	驚訝		0:)	天使		(:	呵欠
	:(難過		X-(生氣		:-B	戴眼鏡		=P~	流口水
	;)	眨眼		:>	得意		=;)	再見		:-?	考慮
	:D	開懷		B-)	耍帥		I-)	睡著			偷笑
	::)	眨眨眼		:-S	擔心		8-)	環顧		=D>	鼓掌
	:-/	疑惑		>:)	邪惡		:-&	不舒服		[-o<	祈禱
	:x	愛意		:((大哭		:-\$	安靜		:-<	嘆氣
	:">>	害羞		:))	大笑		[-(不說了		>:P	呸
	:p	吐舌頭		:	呆住		:o)	小丑		@};-	花
	:*	親親		/:)	皺眉		@-)	神智不清		:@)	豬頭

Blog Post Sample

Tom so crazy but I'm so happy

今天跟你約吃飯 不知為什麼特別緊張 😱

(I had a meal with you today. Don't know why I was so nervous.)

也許因為一陣子沒見吧

(Maybe it's because we hadn't seen each other for a long time.)

謝謝你請我吃飯 還送我禮物 🎁

(Thank you for treating me and giving me a present.)

雖然一直叫你不要送我東西

(Although I told you not to give me anything.)

但收到的時候還是很開心 😊

(But I was very happy receiving it.)

當打開禮物的時候 整個傻眼

(When I opened the gift, I was astonished.)

居然送我 iPod 太誇張了

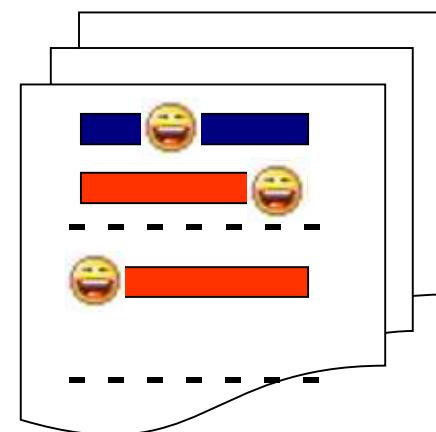
(You gave me an iPod. It's so unbelievable.)

Tom so crazy but I'm so happy 😆



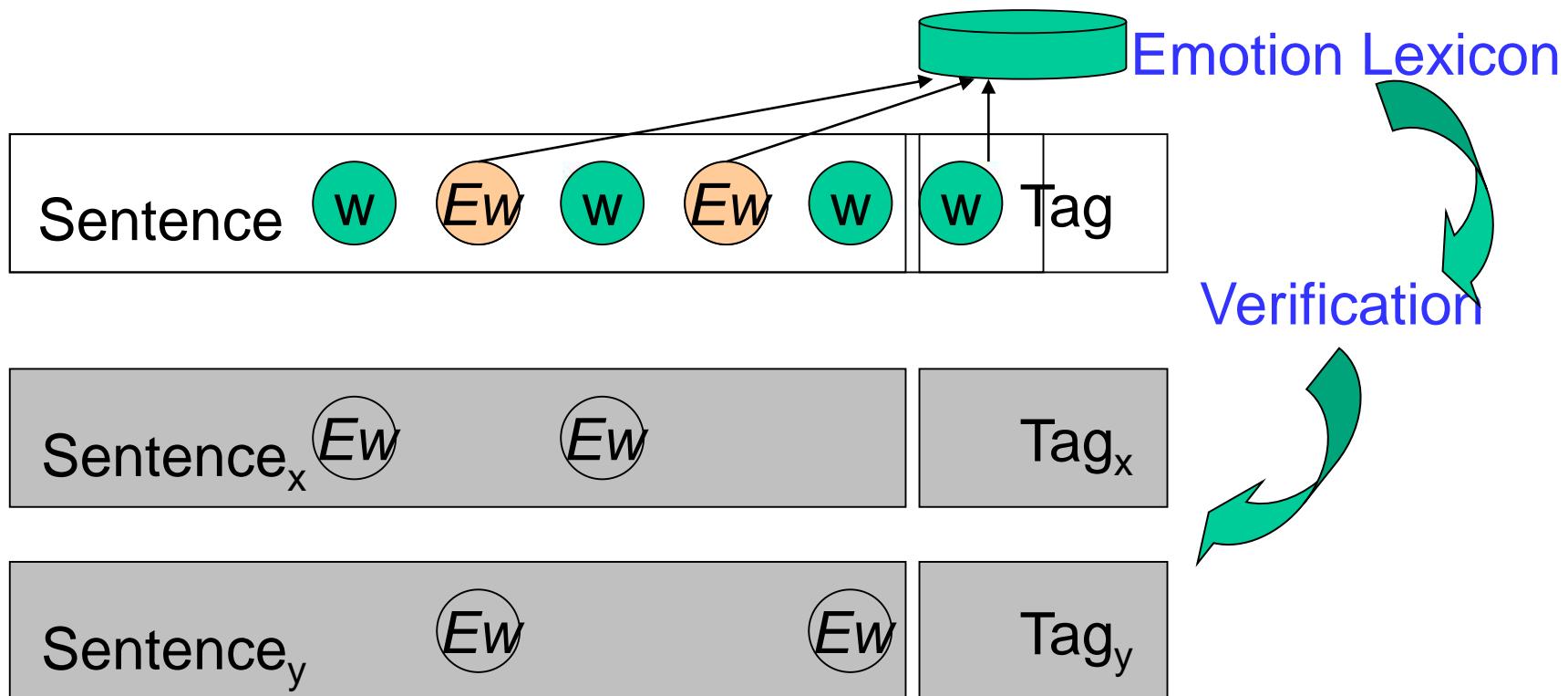
Some Assumptions

- Bloggers use text (raw) and emoticons (tag) to express their emotion
- Their contributions on “tagging” the blog with emotion form a huge collection. (over 10 millions, 10,231,403)
- To extract the knowledge (automatically & computationally)
- First, we need to clarify that
 - The appearance of an emoticon is a good emotion indicator to a
 - Nearby word ?
 - ✓ Sentence
 - Document ?



Basic Ideas

- A sentence is a composition of lexical units

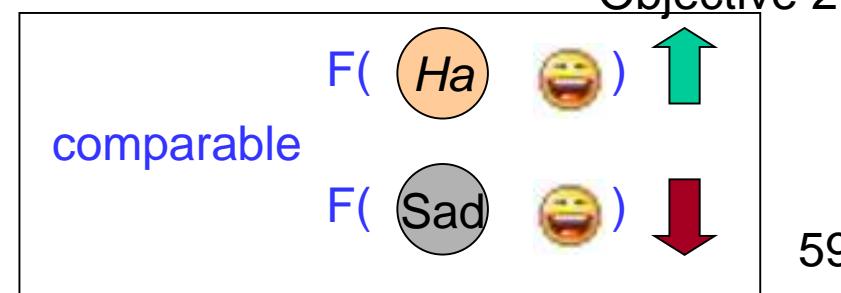
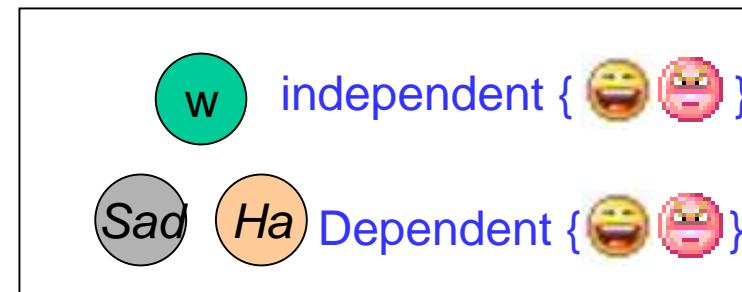


Methodologies for Emotion Lexicon Construction

- Emotion $e = \{e_1, e_2, e_3, \dots, e_m\}$
- Word $w = \{w_1, w_2, w_3, \dots, w_n\}$
- $Es(w_i) = \{es_{i1}, es_{i2}, \dots, es_{io}\} \subset e$ Objective 1

$$Lex = \bigcup_{i=1}^n Es(w_i)$$

Sentence



Methods for Emotion Lexicon Construction

- Collocation test methods
 - Pearson's chi-square
 - Mutual Information
 - Likelihood Ratio

Null hypothesis , $H_0: P(w | e) = p_0 = P(w | \neg e)$

Alternative hypothesis, $H_a: P(w | e) = p_1 \neq p_2 = P(w | \neg e)$

$$p_0 = \frac{C_w}{N}, p_0 = \frac{C_{ew}}{C_e}, p_0 = \frac{C_w - C_{ew}}{N - C_e},$$

$$\log \lambda = \log L(H_1) / \log L(H_2)$$

$$= \log L(C_{ew}, C_e, p) + \log L(C_w - C_{ew}, N - C_e, p)$$

$$- \log L(C_{ew}, C_e, p_1) - \log L(C_w - C_{ew}, N - C_e, p_2)$$

哈哈 (halhal) “hah hah”

Sense 1. 😄 (laughing) – co: 25154.50

e.g., 哈哈... 😄 我應該要出運了~

“hah hah... 😄 I am getting lucky~”

Sense 2. 😊 (big grin) – co: 2667.11

e.g., 今天只背了單母音而已~哈哈 😊

“I only memorized vowels today~ haha 😊”

可惡 (ke3wu4) “darn”

Sense 1. 😡 (angry) – co: 2797.82

e.g., 駭客在搞什麼... 可惡 😡

“What's the hacker doing... darn it 😡”

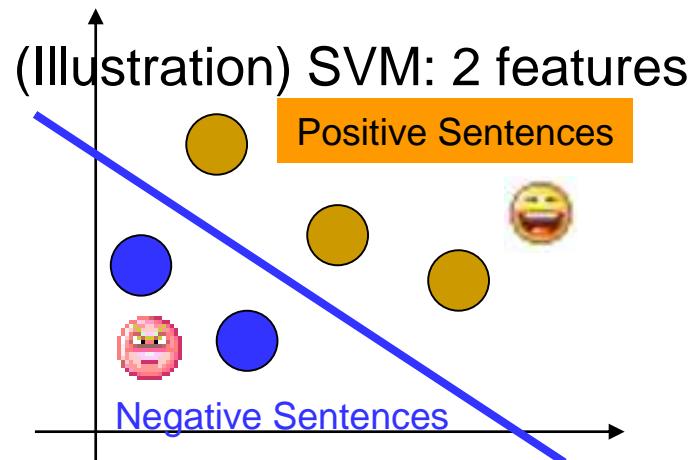
Sense 2. 😠 (phbbbbt) – co: 619.24

e.g., 可惡的外星人... 😠

“Damn those aliens 😠”

Lexicon Method Evaluations

- To evaluate the mining methods by algorithms that use the mined lexicons
- Algorithm based on a sentence classifier
 - Use top 200 lexicon entries as features
 - Tag={Positive, Negative}
 - LIBSVM (Chang & Lin)
 - A test dataset
 - 163,029 sentences



200 features	Accuracy	Application #
Likelihood Ratio Test	78.61%	78,487
Person's Chi-square Test	85.42%	103,336
PMI	84.23%	97,843

Emotional Lexicon

Categorized by Emotions

😂 Laughing								
哈	哈哈哈	哈哈	笑	好笑	開心	高興	呵呵	
😭 Crying								
哭	大哭	感動	難	痛	大家	可以	唉	
😡 Angry								
生气	气死	討厭	吼	脾气	婆婆	老公	居然	
😱 Surprising								
竟然	居然	怎么	哇	發現	結果	醫生	突然	
😴 Sleeping								
睡著	睡覺	累	大睡	瑪咪	夢	呼呼	晚上	
🤤 Drooling								
好吃	口水	蛋糕	流口	美味	雞腿	口味	咬	
🤔 Pondering								
什么	應該	到底	怎么	想想	奇怪	好像	知道	

Emotional Lexicon

Categorized by Words

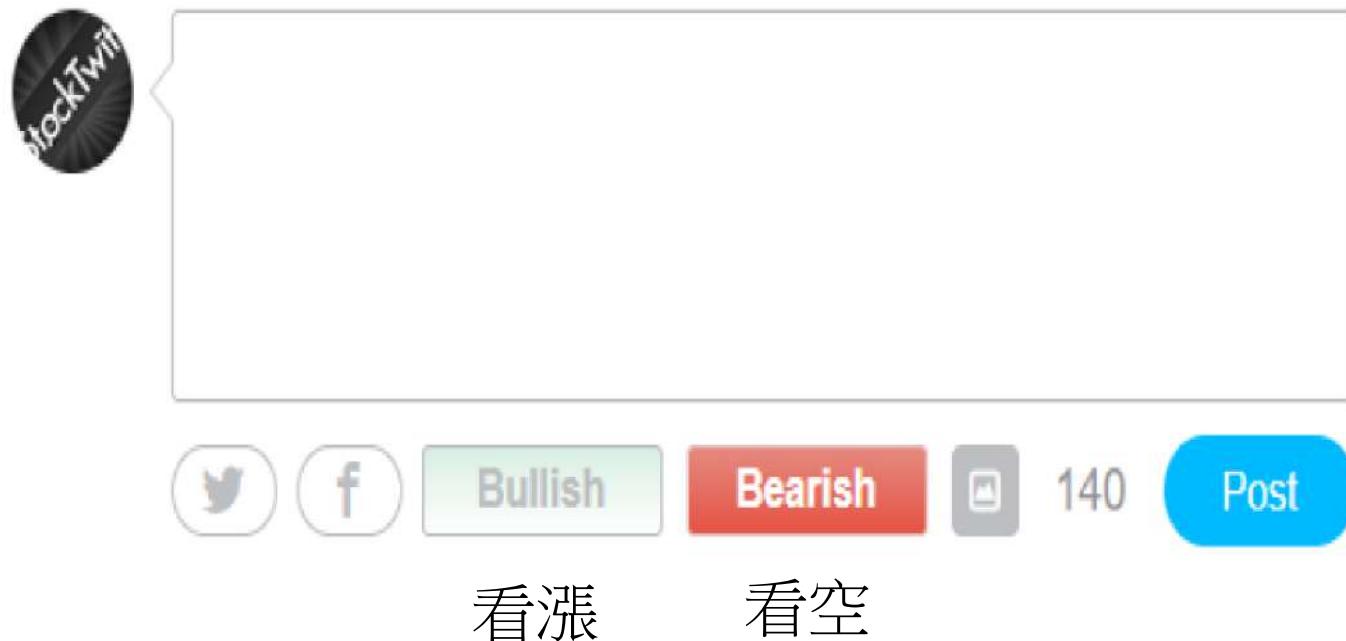
希望		哭		愛	
哈		生气		哈哈哈	
哈哈		幸福		笑	
睡著		什么		鼓勵	
為什麼		怎么		睡覺	
可愛		棒		拜託	
拍拍手		掌聲		好吃	
大哭		寶貝		厲害	
保佑		到底		好笑	
加油		嗚		謝謝	

NTUSD-Fin: A Market Sentiment Dictionary for Financial Social Media Data Applications

Chung-Chi Chen, Hen-Hsen Huang, Hsin-Hsi Chen
Proceedings of The First Financial Narrative
Processing Workshop (FNP 2018)

Fianacial Social Media Data

- StockTwits



Distribution of Dataset

	Bullish	Bearish
Post	289,416	45,382
User	12,452	5,834
Word	69,114	25,956
Hashtag	2,507	715
Emoji	427	174

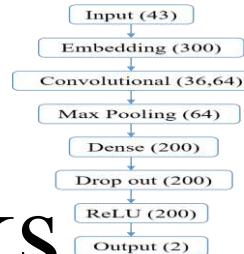
Methods

- Chi-Squared Test
- Collection Frequency-Inverse Document Frequency
- Pointwise Mutual Information
- Convolutional Neural Networks

Statistical Methods

- Chi-squared
 - Determine if there exist the difference between expected and observed frequency.
 - Decide if the token should be remained in our dictionary with the confidence level set to 95%.
- Pointwise Mutual Information
 - Observe how much the token t is correlated to the sentiment s

$$pmi(t, s) = \log \frac{p(t,s)}{p(t)p(s)}$$



Convolutional Neural Networks

- Use the CNN model to train a classifier for market sentiment with our dataset
- Use the output vector of embedding layer as the representation of each token
- Calculate the cosine similarity of each token with the “bullish” and “bearish”
- Subtract the cosine similarity with “bearish” from that with “bullish” to measure the tendency of each token

Top 10 highly frequent tokens for sentiments Bullish and Bearish

Bullish				Bearish							
	Word	Hashtag	Emoji		Word	Hashtag	Emoji				
buy	14,489	stocks	202	☺	927	short	3,653	stocks	68	☺	184
today	13,191	sharkalerts	110	👉	518	going	2,094	noshamenate	40	💩	55
like	11,624	bitcoin	104	🚀	517	stock	2,075	sanofiwasright	34	🐯	35
go	10,959	cesium	102	🌐	332	like	1,989	mnkdsecinvestigation	29	😁	26
get	10,829	trading	92	😊	290	sell	1,897	forex	27	风控	24
going	10,203	bullish	71	😃	282	get	1,678	trading	26	😊	21
back	9,768	stock	71	👀	242	lol	1,663	elliottwave	16	😊	21
day	9,400	brachytherapy	63	🤣	233	today	1,626	earnings	15	😁	20
stock	8,590	tradesmart	60	💰	210	back	1,595	scambags	15	騙子	20
next	8,451	btfd	59	🔜	186	buy	1,592	markets	14	📈	17

Analysis

- “buy” is the most frequent word used in bullish posts
 - a writer expects the price of the mentioned instrument will rise
- “buy” is also in the top-10 tokens in bearish posts
 - mention the buy back price of the instrument
- (😊) emoji gets the first place in both bullish and bearish posts
 - sentiment of an investor may depend on her/his 收益 return from trading

Comparison of NTUSD-Fin with SentiWordNet

	NTUSD-Fin							SentiWordNet	
Word	Market sentiment	Chi squared	Bull freq.	Bull cfidf	Bear freq.	Bear cfidf	Sentiment	Word ID	
buy 買入	0.59	14711.71	14489	61.54	1592	52.32	0.00	buy#1	
sell 賣出	-0.98	3581.53	5800	71.02	1897	51.60	0.00	sell#4	
call 買權	0.44	2211.63	2259	78.16	277	59.76	0.00	call#13	
put 賣權	-0.49	973.82	1326	80.52	310	59.59	0.00	put#1	
overvalued 高估	-3.42	1625.99	54	71.49	172	59.75	0.25	overvalue#1	
undervalued 低估	1.21	1095.06	844	81.71	9	40.81	-0.38	undervalue#1	

<http://nlg.csie.ntu.edu.tw/nlpresource/NTUSD-Fin/>

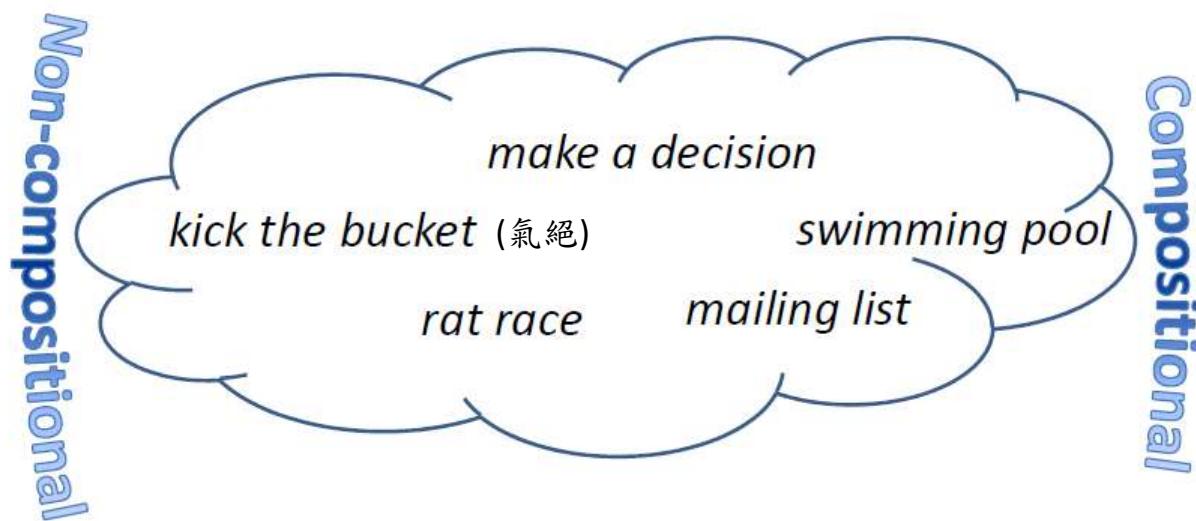
A Word Embedding Approach to Predicting the Compositionality of Multiword Expressions

Bahar Salehi, Paul Cook and Timothy Baldwin

Human Language Technologies: The 2015 Annual Conference of the
North American Chapter of the ACL, pages 977–983

Compositionality

- Whether the meaning of a MWE can be determined from its components
- There is a continuum of compositionality



Research Questions

- Are word embeddings superior to conventional count-based models of distributional similarity?
- How sensitive to parameter optimization are different word embedding approaches?

Methodology

- Estimate the compositionality of an MWE based on the similarity between the expression and its component words in vector space.
 - A simple count-based model of distributional similarity
 - Word embeddings based on WORD2VEC
 - A multi-sense skip-gram model (learn multiple embeddings per target word (or MWE))
- For all three models, first greedily pre-tokenise the corpus to represent each MWE as a single token
- MWE identification takes the form of simple string match for concatenated instances of the component words

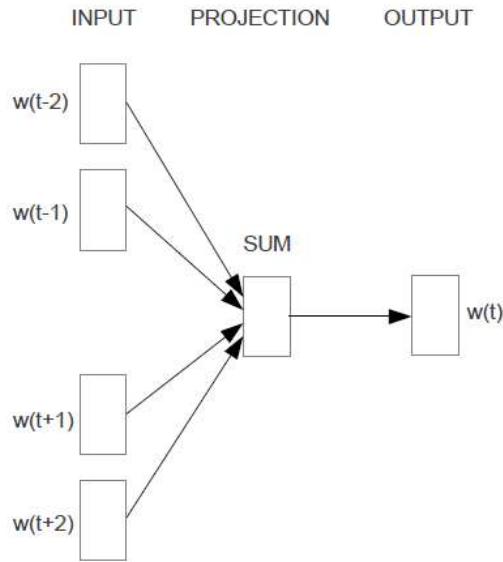
Word Representation

- One-hot representation/One-of-N coding
 - In vector space terms, this is a vector with one 1 and a lot of zeroes
 - motel: [0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 - hotel: [0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 - [0 0 0 0 0 0 0 0 0 1 0 0 0 0] **and** [0 0 0 0 0 0 0 1 0 0 0 0 0 0]=0
- **Count**-based representation
 - Explicit representation: construct a high dimensional sparse matrix M, where each row represents a word w in the vocabulary and each column a potential context.
 - Context: sentence, paragraph, document
 - Singular Value Decomposition (SVD) of term-context matrix
 - Term-document matrix when context is document
- **Prediction**-based (neural) embeddings
 - Continuous Bag-of-Words (CBOW)
 - Skip-Grams
 - Global Vectors (GloVe)

Word Vector Representation Methods

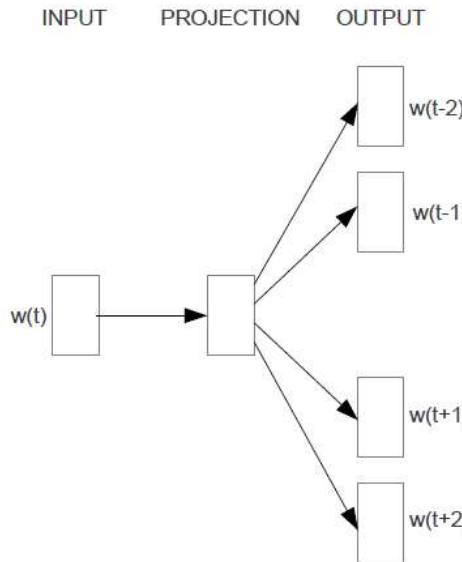
- Dimension reduction on word co-occurrence matrix
 - Latent Semantic Analysis (LSA)
 - Principal Component Analysis (PCA)
 - Latent Dirichlet Allocation (LDA)
- Neural networks (neural embeddings, embeddings)
 - CBOW (Continuous Bag-of-Word)
 - Skip-Gram
 - GloVe (Global Vectors for Word Representation)
 - CWINDOW
 - structured skip-gram
 - word to vector → phrase to vector → document to vector
- Explicit Vector Space Representation (bag of contexts)

Word Representations in Vector Space



CBOW

predict the current word based on the context



Skip-gram

predict surrounding words given the current word

Efficient estimation of word representations in vector space (Mikolov et al, 2013)

word2vec

- word2vec is **not** a single algorithm
- It is a **software package** for representing words as vectors
 - Two distinct models
 - CBOW
 - Skip-Gram
 - Various training methods
 - Negative Sampling
 - Hierarchical Softmax
 - A rich preprocessing pipeline
 - Dynamic Context Windows
 - Subsampling
 - Deleting Rare Words

Count-Based Distributional Similarity

- The dimensions for the vectors
 - The top 50 most-frequent words in the training corpus are considered to be stopwords and discarded
 - Words with frequency rank 51– 1051 are considered to be the content-bearing words
- Similarity of the MWE vector and the component word vectors

$$\begin{aligned} \text{comp}_1(\mathbf{MWE}) &= \alpha \text{sim}(\mathbf{MWE}, \mathbf{C}_1) \\ &\quad + (1 - \alpha) \text{sim}(\mathbf{MWE}, \mathbf{C}_2) \end{aligned}$$

$$\text{comp}_2(\mathbf{MWE}) = \text{sim}(\mathbf{MWE}, \mathbf{C}_1 + \mathbf{C}_2)$$

where **MWE** is the vector associated with the MWE, \mathbf{C}_i is the vector associated with the i th component word of the MWE, sim is a vector similarity function, and $\alpha \in [0, 1]$ is a weight parameter.

WORD2VEC

- Recurrent neural network language model (RNNLM) approach to learning word embeddings
- Continuous bagof-words (CBOW)
- Continuous skip-gram model (C-SKIP)
- WORD2VEC generates a vector of fixed dimensionality d for each pre-tokenised word/MWE type with frequency above a certain threshold in the training corpus

$$\begin{aligned}comp_1(\text{MWE}) &= \alpha sim(\text{MWE}, \mathbf{C}_1) \\&\quad + (1 - \alpha) sim(\text{MWE}, \mathbf{C}_2)\end{aligned}$$

$$comp_2(\text{MWE}) = sim(\text{MWE}, \mathbf{C}_1 + \mathbf{C}_2)$$

Multi-Sense Skip-gram Model

- WORD2VEC: a single word embedding for each word, irrespective of the relative polysemy of the word
- Multi-sense skip-gram (MSSG) model: learns multiple embeddings per word/MWE
- Modify the formulation slightly to handle the variable number of vectors for each word/MWE, by searching over the cross-product of vectors in each sim calculation and taking the maximum in each case

$$comp_1(\text{MWE}) = \alpha sim(\text{MWE}, \mathbf{C}_1)$$

$$+ (1 - \alpha) sim(\text{MWE}, \mathbf{C}_2)$$

$$comp_2(\text{MWE}) = sim(\text{MWE}, \mathbf{C}_1 + \mathbf{C}_2)$$

Experimental Setup

- Two MWE construction types (noun compounds and verb particle constructions) and two languages (English and German)
 - spelling bee (拼字比賽) and swimming pool
 - stand up and give away (免費, 贈送)
- For all experiments, we train our models over raw text Wikipedia corpora for either English or German, depending on the language of the dataset

Datasets

- English noun compounds (“ENCs”, e.g. spelling bee and swimming pool)
 - 90 binary English noun compounds
 - annotated on a continuous [0, 5] scale for both overall compositionality and the component-wise compositionality of each of the modifier and head noun
- English verb particle constructions (“EVPCs”, e.g. stand up and give away)
 - 160 English verb particle constructions
 - manually annotated for compositionality on a binary scale for each of the head verb and particle
- German noun compounds (“GNCs”, e.g. ahornblatt “maple leaf” and eidechse “lizard”)
 - 246 German noun compounds
 - annotated on a continuous [1, 7] scale

Experimental Results

Dataset	Method	<i>comp</i> ₁	<i>comp</i> ₁ + SS	<i>comp</i> ₂	<i>comp</i> ₂ + SS				
WORD2VEC	(<i>d</i> = 500, C-SKIP)	.628	.761	.632	.761	WORD2VEC	(<i>d</i> = 500, C-SKIP)	.393	.442
	(<i>d</i> = 500, CBOW)	.696	.786	.710	.791		(<i>d</i> = 500, CBOW)	.400	.439
	(<i>d</i> = 1000, C-SKIP)	.636	.764	.648	.767		(<i>d</i> = 1000, C-SKIP)	.341	.411
	(<i>d</i> = 1000, CBOW)	.717	.789	.736	.796		(<i>d</i> = 1000, CBOW)	.371	.414
ENC	(<i>d</i> = 300, <i>w</i> = 5)	.640	.764	.624	.759	GNC	(<i>d</i> = 300, <i>w</i> = 5)	.181	.320
	MSSG (<i>d</i> = 600, <i>w</i> = 5)	.615	.758	.594	.758		MSSG (<i>d</i> = 600, <i>w</i> = 5)	.202	.335
	MSSG (<i>d</i> = 600, <i>w</i> = 10)	.614	.749	.631	.756		MSSG (<i>d</i> = 600, <i>w</i> = 10)	.155	.310
	Distributional similarity		.714				Distributional Similarity		.140
EVPC	String similarity		.644			EVPC	String Similarity		.372
	State-of-the-art		.744				State-of-the-art		.450
	(<i>d</i> = 500, C-SKIP)	.289	.496	—	—		green house → 溫室		
	(<i>d</i> = 500, CBOW)	.293	.486	—	—		綠色的房子 (by translation)		
EVPC	(<i>d</i> = 1000, C-SKIP)	.289	.504	—	—	EVPC	string Similarity (溫室,綠色的房子)		
	(<i>d</i> = 1000, CBOW)	.289	.489	—	—				
	(<i>d</i> = 300, <i>w</i> = 5)	.309	.506	—	—		make a decision → 做決定		
	MSSG (<i>d</i> = 600, <i>w</i> = 5)	.294	.498	—	—				
EVPC	MSSG (<i>d</i> = 600, <i>w</i> = 10)	.273	.494	—	—				
	Distributional similarity		.165			EVPC	Word-for-word translations indicate compositionality		
	String similarity		.385				Multiple languages will provide useful additional information		
	State-of-the-art		.417						

MSSG (vector dimensionality *d* and window size *w*)

Summary

- Word embeddings are superior to count-based models of distributional similarity
- The results are relatively stable under parameter optimization for a given word embedding learning approach
- Based on two simple approaches to composition, single word embeddings are empirically slightly superior to multi-prototype word embeddings overall

Predicting the Compositionality of Nominal Compounds: Giving Word Embeddings a Hard Time

Proceedings of the 54th Annual Meeting of the Association for
Computational Linguistics, pages 1986–1997, Berlin, Germany, August
7–12, 2016

Models

- Build three types of DSMs
 - Models based on sparse PPMI co-occurrence vectors
 - Models constructed with word2vec and GloVe
- PPMI
 - For each target word or compound, we extract from the corpus its neighboring nouns and verbs in a symmetric sliding window of w words to the left/right⁵, using a linear decay weighting scheme with respect to its distance d to the target
 - PPMI-thresh: Select the top k most relevant contexts (highest PPMI) for each target. No further dimensionality reduction is applied.
 - PPMI-TopK: Use a fixed global list of 1000 contexts, built by looking at the most frequent words in the corpus: the top 50 are skipped, and the next 1000 are taken. No further dimensionality reduction is applied.
 - PPMI-SVD: build a PPMI matrix and reduce its dimensionality using singular value decomposition (SVD) to factorize the matrix

Models

- w2v: Use the word2vec toolkit based on neural networks to predict target/context co-occurrence
 - CBOW (w2v-cbow)
 - skipgram (w2v-sg)
- Glove: Use the count-based DSM of Pennington et al. (2014), which implements a factorization of the co-occurrence count matrix.

Compositionality Prediction

- To predict the compositionality of a nominal compound w_1w_2 , we use as a measure the cosine similarity between the compound vector representation $v(w_1w_2)$ and the sum of the vector representations of the component words:

$$\cos(v(w_1w_2), v(w_1 + w_2))$$

- where for $v(w_1+w_2)$ we use the normalized sum

$$v(w_1 + w_2) = \frac{v(w_1)}{\|v(w_1)\|} + \frac{v(w_2)}{\|v(w_2)\|}$$

- A compound is compositional if the compound representation is close to the sum of its components representations (cosine is close to 1), and it is idiomatic otherwise.

Still a Pain in the Neck: Evaluating Text Representations on Lexical Composition

Vered Shwartz and Ido Dagan

Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel

arXiv:1902.10618v2 [cs.CL] 19 May 2019

Composition Tasks

- Recognizing Verb-Particle Constructions
 - Given a sentence s that includes a verb V followed by a preposition P , the goal is to determine whether it is a VPC or not.
- Recognizing Light Verb Constructors
 - Given a sentence s that includes a potential light verb construction (“make an easy decision”), the goal is to determine whether it is an LVC or not.
- Noun Compound Literality
 - Given a noun compound $NC = w_1 w_2$ in a sentence s , and a target word $w \in \{w_1, w_2\}$, the goal is to determine whether the meaning of w in NC is literal.
 - *market* in *flea market* (yes), *flea* in *flea market* (no)
- Noun Compound Relations
 - Given a noun compound $NC = w_1 w_2$ in a sentential context s , and a paraphrase p , the goal is to determine whether p describes the semantic relation between w_1 and w_2 or not.
 - “part that makes up body” is a valid paraphrase for *body part*

Composition Tasks

- Adjective-Noun Attributes
 - Given an adjective-noun composition AN in a sentence s, and an attribute AT, the goal is to determine whether AT is implicitly conveyed in AN.
 - the attribute *temperature* is conveyed in *hot water*, but not in *hot argument*
- Identifying Phrase Type
 - Given a sentence, each word is classified to whether it is part of a phrase, and the specific type of the phrase.

Task	Data Source	Train/val/test Size	Input	Output	Context
VPC Classification	Tu and Roth (2012)	919/209/220	sentence s VP = $w_1 \dots w_k$	is VP a VPC?	O
LVC Classification	Tu and Roth (2011)	1521/258/383	sentence s span = $w_1 \dots w_k$	is the span an LVC?	O
NC Literality	Reddy et al. (2011) Tratz (2011)	2529/323/138	sentence s NC = $w_1 \dots w_k$ target w $\in \{w_1, w_2\}$	is w literal in NC?	A
NC Relations	SemEval 2013 Task 4 (Hendrickx et al., 2013)	1274/162/130	sentence s NC = $w_1 \dots w_k$ paraphrase p	does p explicate NC?	A
AN Attributes	HeiPLAS (Hartung, 2015)	837/108/106	sentence s AN = $w_1 \dots w_k$ paraphrase p	does p describe the attribute in AN?	A
Phrase Type	STREUSLE (Schneider and Smith, 2015)	3017/372/376	sentence s	label per token	O

Representations

training objective	corpus (#words)	output dimension	basic unit	
<i>word embeddings</i>				
WORD2VEC	Predicting surrounding words	Google News (100B)	300	word
GLOVE	Predicting co-occurrence probability	Wikipedia + Gigaword 5 (6B)	300	word
FASTTEXT	Predicting surrounding words	Wikipedia + UMBC + statmt.org (16B)	300	subword
<i>contextualized word embeddings</i>				
ELMo	Language model	1B Word Benchmark (1B)	1024	character
OPENAI GPT	Language model	BooksCorpus (800M)	768	subword
BERT	Masked language model (Cloze)	BooksCorpus + Wikipedia (3.3B)	768	subword

Classification Models

- Embed-Encode-Predict models
- Embed

$$\vec{v}_1, \dots, \vec{v}_n = \text{Embed}(s)$$

- Encode
 - biLM

$$\vec{u}_1, \dots, \vec{u}_n = \text{biLSTM}(\vec{v}_1, \dots, \vec{v}_n)$$

- Att

$$\vec{u}_i = [\vec{v}_i; \sum_{j=1}^n a_{i,j} \cdot \vec{v}_j] \quad \vec{a}_i = \text{softmax}(\vec{v}_i^T \cdot \vec{v})$$

- None

$$\vec{u}_1, \dots, \vec{u}_n = \vec{v}_1, \dots, \vec{v}_n$$

- Predict

$$\vec{o} = \text{softmax}(W \cdot \text{ReLU}(\text{Dropout}(h(\vec{x}))))$$

Experimental Results

Model Family	VPC	LVC	NC	NC	AN	Phrase
	Classification	Classification	Literality	Relations	Attributes	Type
Majority Baselines	Acc 23.6	Acc 43.7	Acc 72.5	Acc 50.0	Acc 50.0	F_1 26.6
Word Embeddings	60.5	74.6	80.4	51.2	53.8	44.0
Contextualized	90.0	82.5	91.3	54.3	65.1	64.8
Human	93.8	83.8	91.0	77.8	86.4	-

Model	VPC		LVC		NC		NC		AN		Phrase
	Classification	Layer	Classification	Layer	Literality	Layer	Relations	Encoding	Layer	Encoding	Type
	Layer	Encoding	Layer	Encoding	Layer	Encoding	Layer	Encoding	Layer	Encoding	
ELMo	All	Att	All	bilM	All	Att/None	Top	bilM	All	None	All bilM
OpenAI GPT	All	None	Top	Att/None	Top	None	All	bilM	Top	None	All bilM
BERT	All	Att	All	bilM	All	Att	All	None	All	None	All bilM

Model	VPC		LVC		NC		NC		AN		Phrase
	Classification	Layer	Classification	Layer	Literality	Layer	Relations	Encoding	Layer	Encoding	Type
	Layer	Encoding	Layer	Encoding	Layer	Encoding	Layer	Encoding	Layer	Encoding	
word2vec	bilM		Att		bilM/Att		None		-		None/bilM
GloVe	bilM		Att		Att		bilM		-		bilM
fastText	Att		bilM		bilM		bilM		Att		bilM

Lecture 3. N-Gram Language Models, Smoothing and Discounting

Outlines

- Language Models
 - Counting-Based Language Models (Lecture 3)
 - Prediction-Based Language Models (Lecture 4)
- Counting-Based Language Models
 - N-gram Models (Lecture 3.1)
 - Smoothing (Lecture 3.2)
- Prediction-Based Language Models
 - Neural Probabilistic Language Model and Word Embedding (Lecture 4.1)
 - An Application of Language Model: Chinese Preposition Selection for Grammatical Error Diagnosis (Lecture 4.2)

Lecture 3-1:

N-Gram Language Models

Word Prediction

- Formalize word prediction using N -gram models
 - N -grams: sequences of length N characters/words
- 2-grams (bigram) for the sentence “So I notice three guys standing on the ...”
 - (So I), (I notice), (notice three), (three guys), (guys standing), (standing on), (on the)
- Guess likely next words in a sequence.

N-Gram Models

- Two tasks
 - Use knowledge of the counts of N -grams to assess the conditional probability of candidate words as the next word in a sequence.
一個詞彙序列 w_1, w_2, \dots, w_{n-1} 後面接詞彙 w 的機率值是多少？也就是計算 $P(w|w_1, w_2, \dots, w_{n-1})$
 - Use them to assess the probability of an entire sequence of words.
一個詞彙序列 w_1, w_2, \dots, w_n 的機率值是多少？也就是計算 $P(w_1, w_2, \dots, w_n)$
 - A model that computes either of these is called a **language model**.

Applications

- Assign a probability to a sentence
 - Machine Translation
 - $P(\text{high winds tonite}) > P(\text{large winds tonite})$
 - Spell Correction
 - The office is about fifteen **minuets** from my house
 - $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
 - Speech Recognition
 - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$
 - More applications
 - Summarization, question-answering, etc.

Chinese Spelling Check

■ Example 1

- Input: 我真的洗碗我可以去看你。
- Output: 4, 希, 5, 望

■ Example 2

- Input: 在日本，大學生打工的情況是相當普偏的。
- Output: 17, 遍

■ Example 3

- Input: 我也是你的朋友，我會永遠在你身邊。
- Output: 0

Chinese Grammatical Error Diagnosis

■ Example 1

- Input: 他是我的以前的室友
- output: 4, 4, Redundant ("的")

■ Example 2

- Input: 那電影是機器人的故事
- Output: 2, 2, Missing (between "那" and "電影")

■ Example 3

- Input: 那部電影是機器人的故事
- Output: correct

Chinese Grammatical Error Diagnosis

- Chinese Preposition Selection
 - 在 公眾 利益 方面 來看
 - 從 公眾 利益 方面 來看

$$\hat{p} = \operatorname*{argmax}_{p \in PS} P(x_1, x_2, \dots, x_{i-1}, p, x_{i+1}, \dots, x_n)$$

Author Identification

- Given a document, who wrote it?

法國圖書館發現莎士比亞作品集首版 早期讀者或與天主教圈子有關

■ 全部文章, 歷史知識考掘學 ① February 3, 2015 作者 徐力恆 0 Comments

根據《紐約時報》等媒體報導，法國北部小鎮聖奧梅爾（Saint-Omer）一座公共圖書館於2014年9月發現極其罕見的首版莎士比亞作品集《第一對開本》（First Folio）。當時，為了準備一場英語文學展覽，該館的圖書管理員清點藏書，負責中世紀和近代藏書的主任發現了一本未經保護、落滿灰塵且扉頁和其他介紹性內容都被撕掉的書。他立即諮詢了美國內華達州立大學里諾分校的莎士比亞專家拉斯穆森（Eric Rasmussen）教授。拉斯穆森很快便斷定這是一部《第一對開本》。



Counting: Corpora

- Simple counting: the core of any probabilistic approach
- Counting: Types and Tokens
- Look at large bodies of text instead of single utterances
- Google Web Crawl
 - Crawl of 1,024,908,267,229 English tokens in Web text
 - 13,588,391 word form types

- Numbers
- Misspellings
- Names
- Acronyms
- etc

Language Modeling

- Calculating a conditional probability

- $P(\text{the} \mid \text{its water is so transparent that})$

- By definition

$P(\text{its water is so transparent that the})$

$P(\text{its water is so transparent that})$

- Counting

Get each count in a large corpus.

Language Modeling

- For most sequences and for most text collections, we cannot get good estimates
 - To get is 0. Or worse 0/0.
- Other consideration
 - First use the chain rule of probability
 - Then apply an independence assumption

The Chain Rule

- Definition of conditional probabilities

$$P(A | B) = \frac{P(A, B)}{P(B)}$$

- Rewriting

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

- In general

$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)\dots P(x_n|x_1\dots x_{n-1})$$

Problem for Chain Rule

- A lot of possible sequences
- Cannot get enough data to compute the statistics for longer prefixes

Independence Assumption

- Simplifying assumption
 - $P(\text{lizard}|\text{the}, \text{other}, \text{day}, I, \text{was}, \text{walking}, \text{along}, \text{and}, \text{saw}, a)$
 $= P(\text{lizard}|a)$
 - $P(\text{lizard}|\text{the}, \text{other}, \text{day}, I, \text{was}, \text{walking}, \text{along}, \text{and}, \text{saw}, a)$
 $= P(\text{lizard}|\text{saw}, a)$
- The probability in question is to some degree independent of its earlier history.

Markov Assumption

Each component in the product is replaced with the approximation (assuming a prefix of $N - 1$)

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

Bigram version

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-1})$$

Estimating Bigram Probabilities

- The Maximum Likelihood Estimate (MLE)

$$P(w_i \mid w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

< s > I am Sam </ s >

< s > Sam I am </ s >

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

$$\begin{array}{lll} P(\text{I} \mid \text{<s>}) = \frac{2}{3} = .67 & P(\text{Sam} \mid \text{<s>}) = \frac{1}{3} = .33 & P(\text{am} \mid \text{I}) = \frac{2}{3} = .67 \\ P(\text{</s>} \mid \text{Sam}) = \frac{1}{2} = 0.5 & P(\text{Sam} \mid \text{am}) = \frac{1}{2} = .5 & P(\text{do} \mid \text{I}) = \frac{1}{3} = .33 \end{array}$$

Maximum Likelihood Estimates

- Maximum likelihood estimate of some parameter of a model M from a training set T
- Suppose the word “*Chinese*” occurs 400 times in a corpus of a million words
- What is the probability that a random word from some other text from the same distribution will be “*Chinese*”
- MLE estimate is $400/1000000 = 0.004$

Berkeley Restaurant Project Sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

Raw Bigram Counts

- Out of 9,222 sentences
 - Eg. “*I want*” occurred 827 times

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Bigram Probabilities

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

- Divide bigram counts by prefix unigram counts to get probabilities.

prefix unigram counts

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Bigram Estimates of Sentence Probabilities

- $P(<\text{s}> I \text{ want english food } </\text{s}>) =$
 $P(I|<\text{s}>)^*$
 $P(\text{want}|I)^*$
 $P(\text{english}|\text{want})^*$
 $P(\text{food}|\text{english})^*$
 $P(</\text{s}>|\text{food})^*$
 $= .000031$

Kinds of Knowledge

- N -gram probabilities capture a range of interesting facts about language.
 - $P(\text{english} | \text{want}) = .0011$
 - $P(\text{chinese} | \text{want}) = .0065$
 - $P(\text{to} | \text{want}) = .66$
 - $P(\text{eat} | \text{to}) = .28$
 - $P(\text{food} | \text{to}) = 0$
 - $P(\text{want} | \text{spend}) = 0$
 - $P(i | \langle s \rangle) = .25$
- World knowledge
- Syntax
- Discourse

Shannon's Method

- Generate random sentences that are like the sentences from which the model was derived
- Sample a random bigram ($< s >$, w) according to its probability
- Sample a random bigram (w, x) according to its probability
- And so on until we randomly choose a (y, $< /s >$)
- An example
- $< s >$ I

I want

want to

to eat

eat Chinese

Chinese food

food $< /s >$

Shakespeare

Unigram	<ul style="list-style-type: none">• To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have• Every enter now severally so, let• Hill he late speaks; or! a more to leg less first you enter• Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like
Bigram	<ul style="list-style-type: none">• What means, sir. I confess she? then all sorts, he is trim, captain.• Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.• What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?• Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
Trigram	<ul style="list-style-type: none">• Sweet prince, Falstaff shall die. Harry of Monmouth's grave.• This shall forbid it should be branded, if renown made it empty.• Indeed the duke; and had a very good friend.• Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
Quadrigram	<ul style="list-style-type: none">• King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;• Will you not tell me who I am?• It cannot be but so.• Indeed the short and the long. Marry, 'tis a noble Lepidus.

Shakespeare as a Corpus

- $N=884,647$ tokens, $V=29,066$
- Shakespeare produced 300,000 bigram types out of $V^2= 844$ million possible bigrams
 - 99.96% of the possible bigrams never appear
 - The biggest problem in language modeling
- Quadrigrams are worse



The Wall Street Journal is Not Shakespeare

unigram: Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives

bigram: Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

trigram: They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Practical Issues

- We do everything in log space
 - Avoid underflow
 - Adding is faster than multiplying

$$\log(p_1 \cdot p_2 \cdot p_3 \cdot p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Google N-Gram Release

All Our N-gram are Belong to You

By Peter Norvig - 8/03/2006 11:26:00 AM

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word [n-gram models](#) for a variety of R&D projects, such as [statistical machine translation](#), speech recognition, [spelling correction](#), entity detection, information extraction, and others. While such models have usually been estimated from training

to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensable 40
- serve as the individual 234

Web N-gram Version

- Linguistic Data Consortium
 - Web 1T 5-gram Version (2006)
 - Web 1T 5-gram, 10 European Languages (2009)
 - Japanese Web N-gram Version (2009)
 - Chinese Web 5-gram Version (2010)

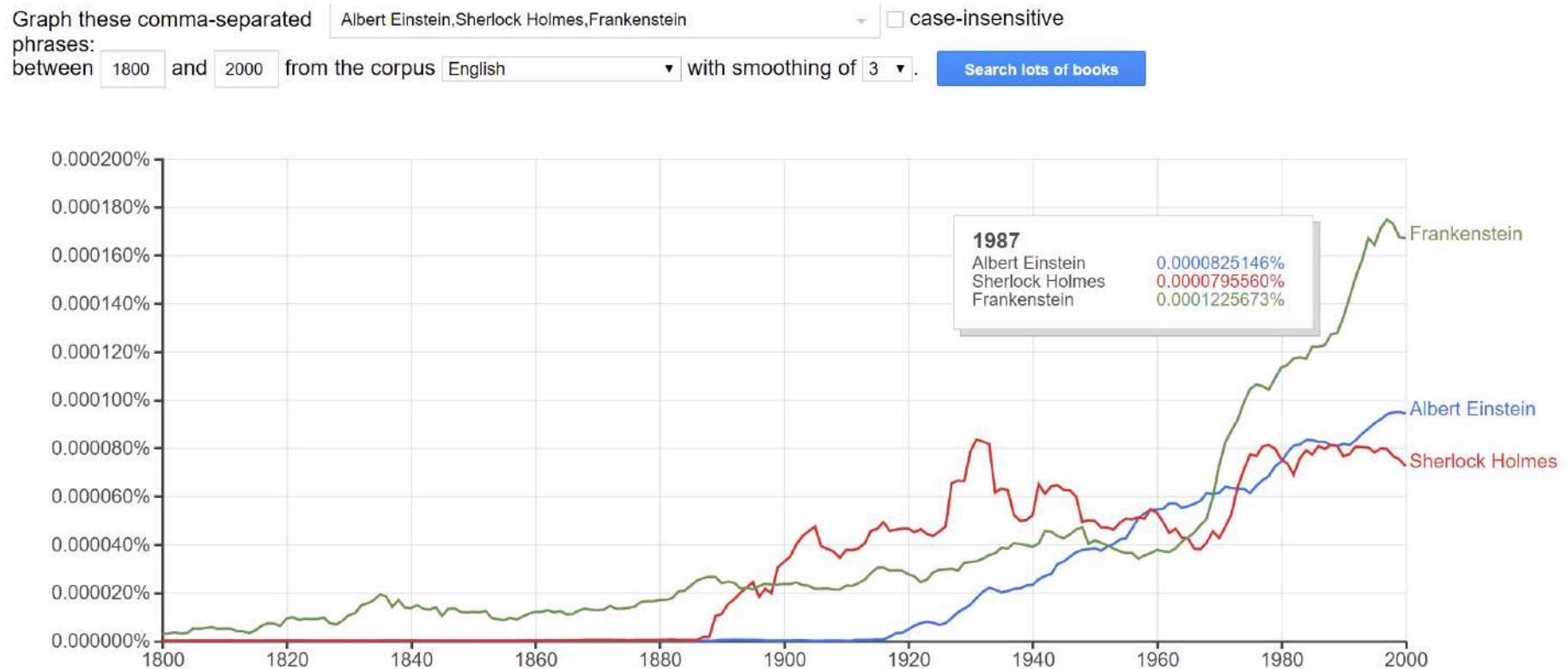
Chinese Web 5-gram Version 1

- Created by researchers at Google Inc.
- Chinese word n-grams and their observed frequency counts generated from over 800 million tokens of text
- the length of the n-grams ranges from unigrams (single words) to 5-grams
- n-grams that appeared at least 40 times in the processed sentences
- for statistical language modeling (e.g., segmentation, machine translation) as well as for other uses

惯例	为 电影 创作	52
惯例	为 的 是	95
惯例	为 目标 职位	49
惯例	为 确保 合作	69
惯例	为 确保 重组	213
惯例	为 科研 和	55
惯例	为 统称 </s>	183
惯例	为 维 和	50
惯例	为 自己 的	43
惯例	为 艺术类 学院	44
惯例	为 避免 侵权	148
惯例	为 那些 有	45
惯例	为 配合 奥运	41
惯例	为 防止 妇人	58
惯例	为主 因为 国际	40
惯例	为了 保护 知识产权	40
惯例	为了 保证 课件	117
惯例	为了 加强 产品	94
惯例	为了 后来 剧情	48
惯例	为了 向 破产	63
惯例	为了 呈现 光	72
惯例	为了 在 这个	144
惯例	为了 控制 收购	73
惯例	为了 获取 更	111
惯例	为了 规避 利益	43
惯例	为了 追求 上半年	40
惯例	为什么 4 位	45
惯例	为什么 在 中国	161
惯例	为什么 在 这	65
惯例	为什么 提前 这么	44
惯例	为什么 要 按	56
惯例	为什么 要 按照	44
惯例	为何 总 与	47
惯例	为何 近期 在	167
惯例	为期 两 天	41
惯例	为由 强行 收取	108
惯例	为由 把 国内	63
惯例	为由 拒 建	167
惯例	为由 拒绝 </s>	354
惯例	为由 拒绝 承付	61
惯例	为由 拒绝 承担	40
惯例	为由 替 自己	337

Google Books Ngram Viewer

- <https://books.google.com/ngrams>



Evaluating N -Gram Models

- *Extrinsic evaluation* for a language model
 - Put model A into an application
 - Evaluate the performance of the application with model A
 - Put model B into the application and evaluate
 - Compare performance of the application with the two models

Intrinsic Evaluation

- Extrinsic evaluation
 - Time-consuming
 - Take days to run an experiment
- Intrinsic evaluation
 - To evaluate N-grams with an approximation called **perplexity**

Perplexity

- Perplexity: the notion of surprise
 - How surprised is the language model when it sees the test set?
 - The more surprised the model is, the lower the probability it assigned to the test set
 - The higher the probability, the less surprised it was

Perplexity

- Perplexity is the probability of a test set (assigned by the language model), as normalized by the number of words:
$$\text{PP}(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$
- Chain rule:
$$\text{PP}(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$
- For bigrams:
$$\text{PP}(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_{i-1})}}$$
- Minimizing perplexity is the same as maximizing probability
 - **The best language model is one that best predicts an unseen test set**

Lower perplexity means a better model

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Lecture 3-2: Smoothing

Generalization and zeros

n-gram Model

- Without assuming anything

$$P(w_1, w_2, w_3, \dots, w_{l-1}, w_l) =$$

$$P(w_1)P(w_2 | w_1)P(w_3 | w_1, w_2) \dots P(w_l | w_1, w_2, \dots, w_{l-2}, w_{l-1})$$

- Map context into a finite set of equivalence classes
- n-gram model assumes equivalence classes are previous n-1 words

$$P_{trigram}(w_1, w_2, w_3, \dots, w_{l-1}, w_l) =$$

$$P(w_1)P(w_2 | w_1)P(w_3 | w_1, w_2)P(w_4 | w_2, w_3) \dots$$

$$P(w_{l-1} | w_{l-3}, w_{l-2})P(w_l | w_{l-2}, w_{l-1})$$

n-gram Model

history
 $P(w_n | w_1, \dots, w_{n-1})$

Group histories that are similar to give reasonable prediction

- **Markov Assumption:** Only the prior $n-1$ local context affects the next entry:
 $(n-1)th$ Markov Model or n-gram
- Size of the n-gram models versus number of parameters
 - Larger n much better context
 - Total number of n-grams scales increases exponentially with n.
- Estimate conditional probabilities as a ratio of joint probabilities

$$P(w_n | w_{n-2}, w_{n-1}) = \frac{P(w_{n-2}, w_{n-1}, w_n)}{P(w_{n-2}, w_{n-1})}$$

Model	Number of Parameters
unigram	60000
bigram	$60000^2 = 3.6 \times 10^9$
trigram	$60000^3 = 2.16 \times 10^{14}$
fourgram	$60000^4 = 1.296 \times 10^{19}$

Statistical Estimators

- **Goal**: To derive a good probability estimate for the target feature based on observed data
- **Running Example**: From n-gram data $P(w_1, \dots, w_n)$'s predict $P(w_n | w_1, \dots, w_{n-1})$

$$P(w_n | w_1, \dots, w_{n-1}) = \frac{P(w_1, \dots, w_n)}{P(w_1, \dots, w_{n-1})}$$

- **Solutions**
 - Maximum Likelihood Estimation
 - Laplace's, Lidstone's and Jeffreys-Perks' Laws
 - Discounting
 - Held Out Estimation
 - Cross-Validation
 - Good-Turing Estimation
 - Kneser-Ney Smoothing

Maximum Likelihood Estimation

$$P_{MLE}(w_1, \dots, w_n) = \frac{C(w_1, \dots, w_n)}{N}$$

$$P_{MLE}(w_n \mid w_1, \dots, w_{n-1}) = \frac{C(w_1, \dots, w_n)}{C(w_1, \dots, w_{n-1})}$$

- Choose parameters to give the highest probability to the training corpus
- The sparseness of the data
 - Use a Discounting or Smoothing technique

Data Sparseness

- Language modeling suffers a data sparsity
 - e.g., trigrams, 60,000 word vocabulary
 - A large training set may contain 200 million words
- Data sparsity + maximum likelihood=zero probabilities!
- If word sequence $w_i w_{i+1} w_{i+2}$ is not observed then $P(w_{i+2} | w_i w_{i+1})$ is set to 0 by maximum likelihood
- Any word sequence including the sub-sequence $w_i w_{i+1} w_{i+2}$ will have probability 0
- Any word sequences that was not present in the training data cannot be recognized.

Smoothing

- MLE assigns a zero probability to unseen events
- Zeros propagate and give bad estimates
- Smoothing
 - Decrease the probability of previously seen events
 - A little bit of probability mass left over for previously unseen events

Smoothing

- Smooth the probability estimates so that no word sequences is given a probability of 0.
- Discounting
 - Adjust probability estimators, so that 0 relative frequency in the training data does not imply 0 relative counts
- Model combination
 - Combine models (unigram, bigram, trigram, ...) in such a way we use the most precise model available
 - interpolation and back-off

The intuition of smoothing

- When we have sparse statistics:

$P(w | \text{denied the})$

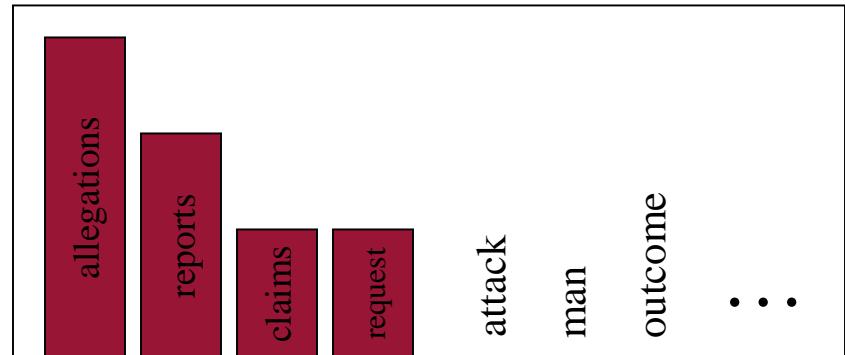
3 allegations

2 reports

1 claims

1 request

7 total



- Steal probability mass to generalize better

$P(w | \text{denied the})$

2.5 allegations

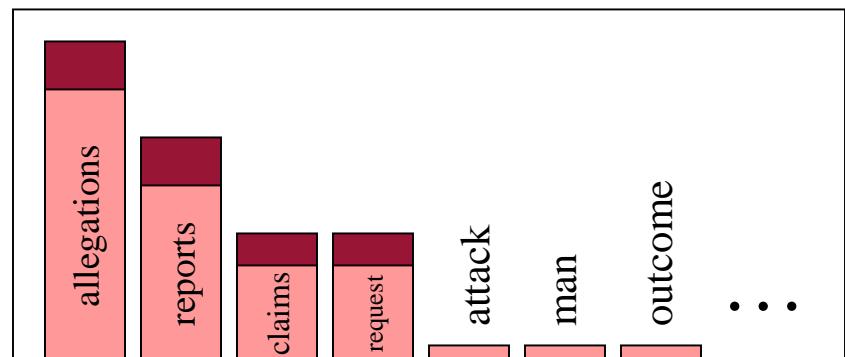
1.5 reports

0.5 claims

0.5 request

2 other

7 total



Discounting: Laplace

- Add 1 to every count

- MLE

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$



- Add-1 estimate

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

- Problem

- Give far too much of the probability space to unseen events.

Laplace-Smoothed Bigram Counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Add 1 to each cell →

每一格都增加一次，表示
每個word隨著增加V次

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Laplace-Smoothed Bigram Probabilities

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

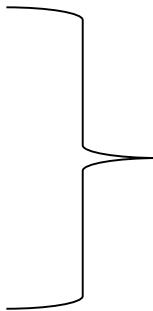
V=1446

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Reconstituted Counts

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

$$P_{\text{Laplace}}^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$



$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Reconstituted Counts (2)

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Lidstone and Jeffreys-Perks

- Add a smaller value λ .
- *Lidstone's Law*

$$P_{Lid}(w_1, \dots, w_n) = \frac{C(w_1, \dots, w_n) + \lambda}{N + B\lambda} \quad (\lambda > 0)$$

- If $\lambda=1/2$
 - Expected Likelihood Estimation (ELE) or the Jeffreys-Perks Law

Good-Turing

■ Good-Turing discounting

- use the count of things we've seen once to help estimate the count of things we've never seen
- high frequency words are quite accurate, so one does not need to discount them

$$c^{*}_{GT} = (c+1) \frac{N_{c+1}}{N_c} \quad c < k$$

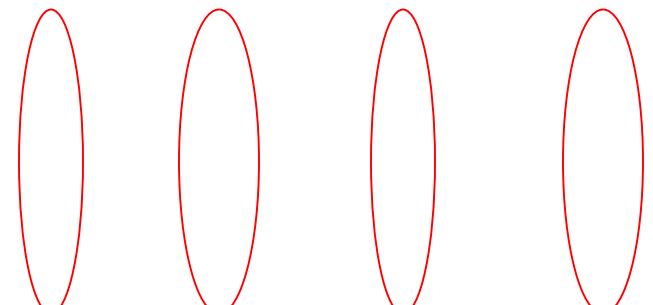
$$c^{*}_{GT} = c \text{ for } c > k$$

- k is a threshold
- Note that for $r=0$,

$$c^{*}_{GT} = \frac{N_1}{N_0}$$

原tokens數
增加的個數

N_0	0	N_1	$2N_2$	$3N_3$
		N_0	N_1	N_2



Good-Turing

- The discounted probabilities are thus:

$$P_{GT}(w_{n-1}, w_n) = \frac{c^*}{N}$$

- Formula only applies when $r < k$ where k is small (e.g., 5)
- Need to renormalize to ensure everything sums to 1.

$$c^* = \frac{(c+1)\frac{N_{c+1}}{N_c} - c\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}}, \text{ for } 1 \leq c \leq k$$

One More Consideration

- The estimate c^* for N_c depends on $N_{c+1} \rightarrow$ when $N_{c+1}=0$, $c^{*}_{GT} = (c+1) \frac{N_{c+1}}{N_c}$ is undefined
- Simple Good-Turing
 - Smooth N_c counts to replace any zeros in the sequence by linear regression

$$\log(N_c) = a + b \log(c)$$

Good-Turing

Josh Goodman Intuition

- Imagine you are fishing
 - There are 8 species: carp (鯉魚), perch (河鱸), whitefish (白魚), trout (鱒魚), salmon (鮭魚), eel (鰻魚), catfish (鲶魚), bass (鱸魚)
- You have caught up to now
 - 10 carp, 3 perch, 2 whitefish, 1 trout, 1 salmon, 1 eel = 18 fish
- How likely is it that the next fish to be caught is an eel?
- How likely is it that the next fish caught will be a member of newly seen species?

Good-Turing

- Notation: N_x is the frequency-of-frequency-x
 - $N_{10}=1$
 - Number of fish species seen 10 times is 1 (carp)
 - $N_1=3$
 - Number of fish species seen 1 is 3 (trout, salmon, eel)
- To estimate total number of unseen species
 - Use number of species (words) we've seen once
 - $c_0^* = c_1 \quad p_0 = N_1/N$
- All other estimates are adjusted downward to account for unseen probabilities

$$c^* = (c + 1) \frac{N_{c+1}}{N_c}$$

$$P(\text{eel}) = c^*(1) = (1+1) 1/ 3 = 2/3$$

GT Fish Example

	unseen (bass or catfish)	trout
c	0	1
MLE p	$p = \frac{0}{18} = 0$	$\frac{1}{18}$
c^*	Total count for unseen=3	$c^*(\text{trout}) = 2 \times \frac{N_2}{N_1} = 2 \times \frac{1}{3} = .67$
GT p_{GT}^*	$p_{\text{GT}}^*(\text{unseen}) = \frac{N_1}{N} = \frac{3}{18} = .17$	$p_{\text{GT}}^*(\text{trout}) = \frac{.67}{18} = \frac{1}{27} = .037$

Total probabilities for unseen

Absolute Discounting

- Absolute Discounting subtracts a constant δ from each non-zero count, and the “left over” probability is distributed over unseen events:

$$r^*_{Abs} = r - \delta \quad \text{if } r > 0$$

$$= \frac{\delta(B - N_0)}{N_0} \quad \text{if } r = 0$$

- Estimate δ from held-out data – value of around 0.75 would work well in the example

Absolute discounting: just subtract a little from each count

- Suppose we wanted to subtract a little from a count of 4 to save probability mass for the zeros
- How much to subtract ?
- Church and Gale's clever idea
- Divide up 22 million words of AP Newswire
 - Training and held-out set
 - for each bigram in the training set
 - see the actual count in the held-out set!

Bigram count in training	Bigram count in held out set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

It sure looks like $c^* = (c - .75)$

Combining Estimators

- If we have several models of how the history predicts what comes next, then we might wish to combine them in the hope of producing an even better model.
- Backoff
 - use trigram if you have good evidence
 - otherwise bigram, otherwise unigram
- Interpolation
 - mix unigram, bigram, trigram
- Interpolation works better

Interpolation

- As well as adjusting counts by discounting, it is better to use less-specific models, rather than relying on the default “unseen probability”
- Interpolation: interpolate lower-models with higher order models, e.g., for trigrams:

$$P_{\text{int}}(w_n \mid w_{n-2}, w_{n-1}) = \lambda_3 P_3(w_n \mid w_{n-2}, w_{n-1}) + \lambda_2 P_2(w_n \mid w_{n-1}) \\ + \lambda_1 P_1(w_n) + \frac{1}{N}$$

- Constraints: $\sum_i \lambda_i = 1; \quad 0 \leq \lambda_i \leq 1$
- More generally λ can be a function of (w_{n-2}, w_{n-1}, w_n) (e.g. $\lambda(r)$)

How to set the lambdas?

- Use a held-out corpus



- Choose λ s to maximize the probability of held-out data:
 - Fix the N-gram probabilities (on the training data)
 - Then search for λ s that give largest probability to held-out set

$$\log P(w_1 \dots w_n \mid M(/_1 \dots /_k)) = \sum_i \log P_{M(/_1 \dots /_k)}(w_i \mid w_{i-1})$$

N-gram Smoothing Summary

- Add-1 smoothing:
 - OK for text categorization, not for language modeling
- The most commonly used method:
 - Extended Interpolated Kneser-Ney
- For very large N-grams like the Web:
 - Stupid backoff

Kneser-Ney Smoothing I

- Better estimate for probabilities of lower-order unigrams!
 - Shannon game: I can't see without my reading _____?
 - "Francisco" is more common than "glasses"
 - ... but "Francisco" always follows "San"
- The unigram is useful exactly when we haven't seen this bigram!
- Instead of $P(w)$: "How likely is w "
- $P_{\text{continuation}}(w)$: "How likely is w to appear as a novel continuation?
 - For each word, count the number of bigram types it completes
 - Every bigram type was a novel continuation the first time it was seen

$$P_{\text{CONTINUATION}}(w) \propto \frac{1}{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}$$

Kneser-Ney Smoothing II

- How many times does w appear as a novel continuation:

$$P_{CONTINUATION}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

- Normalized by the total number of word bigram types

$$|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|$$

$$P_{CONTINUATION}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|}$$

Kneser-Ney Smoothing III

- Alternative metaphor: The number of # of word types seen to precede w

$$|\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

- normalized by the # of words preceding all words:

$$P_{CONTINUATION}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{\sum_{w'} |\{w'_{i-1} : c(w'_{i-1}, w') > 0\}|}$$

- A frequent word (Francisco) occurring in only one context (San) will have a low continuation probability

Kneser-Ney Smoothing IV

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + / (w_{i-1}) P_{CONTINUATION}(w_i)$$

λ is a normalizing constant; the probability mass we've discounted

$$/ (w_{i-1}) = \frac{d}{c(w_{i-1})} \left| \{w : c(w_{i-1}, w) > 0\} \right|$$

the normalized discount

The number of word types that can follow w_{i-1}
= # of word types we discounted
= # of times we applied normalized discount

$w_{i-1} w_i$ 減少 d 次，相當於 w_{i-1} 連帶也少了 d 次，
減少比率為 $d/c(w_{i-1})$

San Jose
San Franciso

Kneser-Ney Smoothing: Recursive Formulation

$$P_{KN}(w_i | w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^i) - d, 0)}{c_{KN}(w_{i-n+1}^{i-1})} + / (w_{i-n+1}^{i-1}) P_{KN}(w_i | w_{i-n+2}^{i-1})$$

$$c_{KN}(\cdot) = \begin{cases} \text{↑} & \text{count}(\cdot) \quad \text{for the highest order} \\ \text{↑} & \text{continuationcount}(\cdot) \quad \text{for lower order} \end{cases}$$

Continuation count = Number of unique single word contexts for •

Smoothing for Web-scale N-grams

- Stupid backoff
- No discounting, just use relative frequencies

$$S(w_i | w_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{count}(w_{i-k+1}^i)}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^i) > 0 \\ 0.4S(w_i | w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

Lecture 4: Neural Language Model and Word Embedding

The Language Modeling Task

- A goal of **statistical language modeling** is to learn the joint probability function of sequences of words in a language.

$$P(w_1, w_2, \dots, w_T)$$

- A **statistical model of language** can be represented by the conditional probability of the next word given all the previous ones

$$\begin{aligned} P(w_1, w_2, \dots, w_T) &= \prod_{t=1}^T P(w_t | w_1, \dots, w_{t-1}) \\ &\approx \prod_{t=1}^T P(w_t | w_1, \dots, w_{t-n+1}) \end{aligned}$$

Unseen N-grams

- What happens when a new combination of n words appears, but was not seen in the training corpus?
 - Zero probability!!
$$P(w_1, w_2, \dots, w_T) \approx \prod_{t=1}^T P(w_t | w_1, \dots, w_{t-n+1})$$
- A simple answer is to look at the probability predicted using a smaller context size, as done in:
 - Back-off n-gram models
 - Interpolated n-gram models
 - Modified Kneser Ney smoothing

Two Issues

- It does not take into account the contexts farther than 1 or 2 words.
- It does not take into account the “similarity” between words.
 - The **cat** is **walking** in the **bedroom**
 - A **dog** was **running** in a **room**

Pros and Cons of MLE-based Language Models

- Pros
 - Easy to train
 - Scale to large corpora
 - Work well in practice
- Cons
 - A fixed backing-up order, that needs to be designed **by hand**
 - Hard to scale towards **larger n-grams** in order to capture **long-range dependencies** (refer to next 4 slides)
 - Statistics for larger n-grams will be **sparse**
 - Lack of generalization across contexts
 - Having observed **black car** and **blue car** does not influence our estimates of the event **red car** if we haven't seen it before

Local and Non-Local Dependencies

- n A ***local dependency*** is a dependency between two words expressed within the same syntactic rule
- n A ***non-local dependency*** is an instance in which two words can be syntactically dependent even though they occur far apart in a sentence (e.g., ***subject-verb agreement***; ***long-distance dependencies*** such as ***wh-extraction***).
- n Non-local phenomena are a challenge for certain ***statistical NLP approaches*** (e.g., ***n-grams***) that model local dependencies.

Unbounded Long-Range Dependency

- n ***wh*-movement (relative clauses, *wh*-questions)**
 - *the guy that [I believe Peter told me you thought] you like.*
 - *who do [you believe Peter told you I thought] I like?*
- n **Topicalization:**
 - *That guy, [I believe Peter told me you thought] you like.*
- n **Clefts:** (cleft sentence 分裂句)
 - *It's that guy that [I believe Peter told me you thought] you like*

Nonlocal dependency annotation in Penn Treebanks: unindexed empty elements, dislocations, and control.

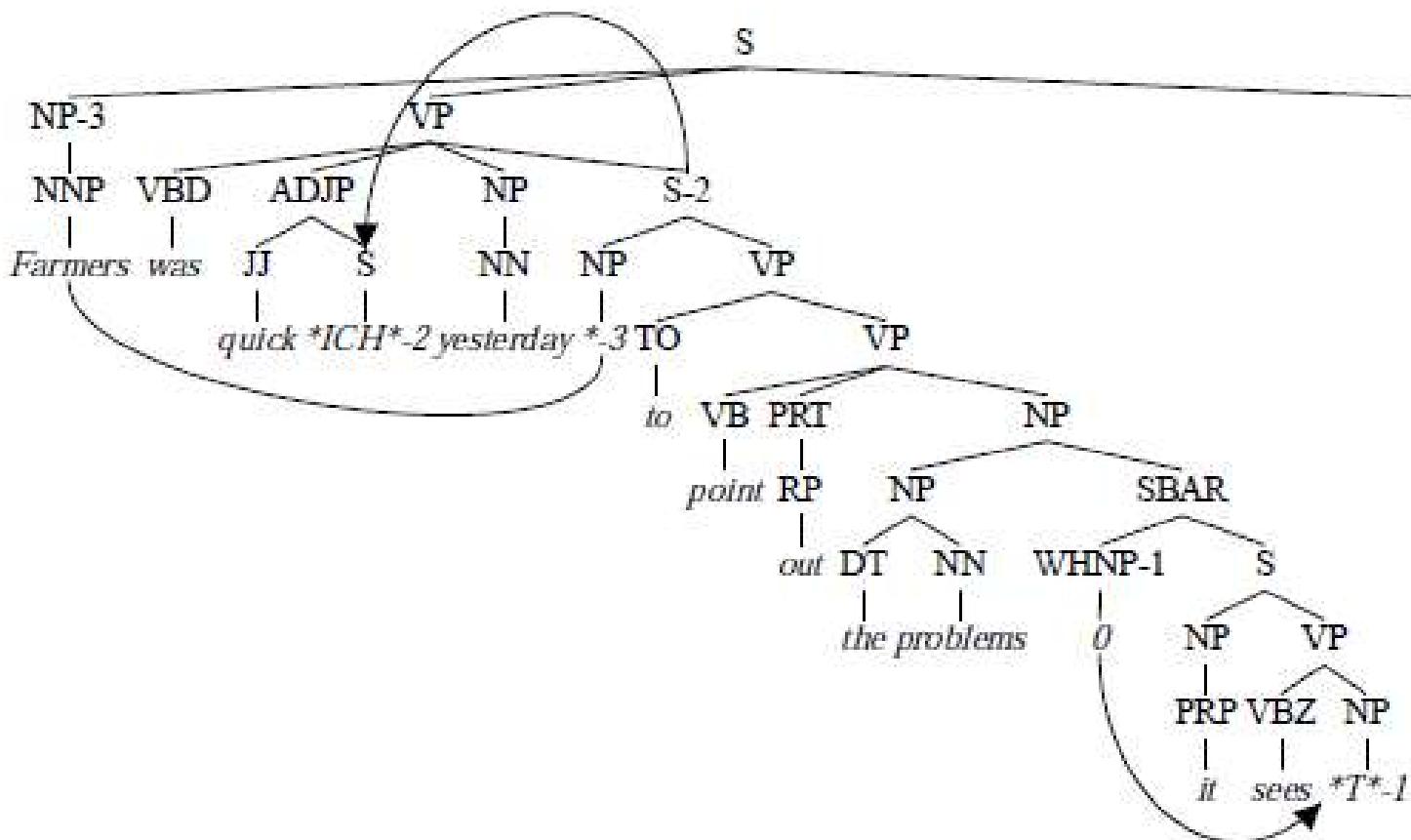


Figure 1: Example of empty and nonlocal annotations from the Penn Treebank of English, including null complementizers (\emptyset), relativization ($*T^*-I$), right-extraposition ($*ICH^*-2$), and syntactic control ($*-3$).

Unbounded Long-Range Dependency

- n **Right-node raising:**
 - *[[she would have bought] and [he might sell]] shares.*
- n **Argument-cluster coordination:**
 - *I give [[you an apple] and [him a pear]].*
- n **Gapping:**
 - *[She likes sushi], and [he sashimi].*

Distributed Representations

- Associate each word in the vocabulary with a distributed word feature vector.
- Express the joint probability function of word sequences in terms of the feature vectors of these words in the sequence.
- Learn simultaneously the word feature vectors and the parameters of that probability function.

Why does it work?

- If we know that dog and cat played similar roles (semantically and syntactically), and similarly for (the, a), (bedroom, room), (is, was), (running, walking), we could naturally **generalize** (i.e., **transfer probability mass**)

The cat is walking in the bedroom



A dog was running in a room



The cat is running in a room
A dog is walking in a bedroom
The dog was walking in the room

...

A Neural Network Language Model

- The training set is a sequence w_1, w_2, \dots, w_T of words $w_t \in V$, where the vocabulary V is a large but finite set.
- The objective is to learn a good predicting model

$$f(w_{t-n+1}, w_{t-n+2}, \dots, w_t) = P(w_t | w_{t-n+1}^{t-1})$$

$$\sum_{i=1}^V f(w_{t-n+1}, w_{t-n+2}, \dots, w_{t-1}, w_i) = 1$$

$$f(\cdot) > 0$$

A Neural Network Language Model

- We decompose $f(w_{t-n+1}, w_{t-n+2}, \dots, w_t) = P(w_t | w_{1-n+1}^{t-1})$ into two parts.
- A mapping C from any element i of V to a real vector $C(i) \in \mathbb{R}^m$
 - It represents the distributed feature vectors associated with each word in the vocabulary.
- A function g maps an input sequence of feature vectors for words in context, $[C(w_{t-n+1}), \dots, C(w_{t-1})]$, to a conditional probability distribution over words in V for the next word.

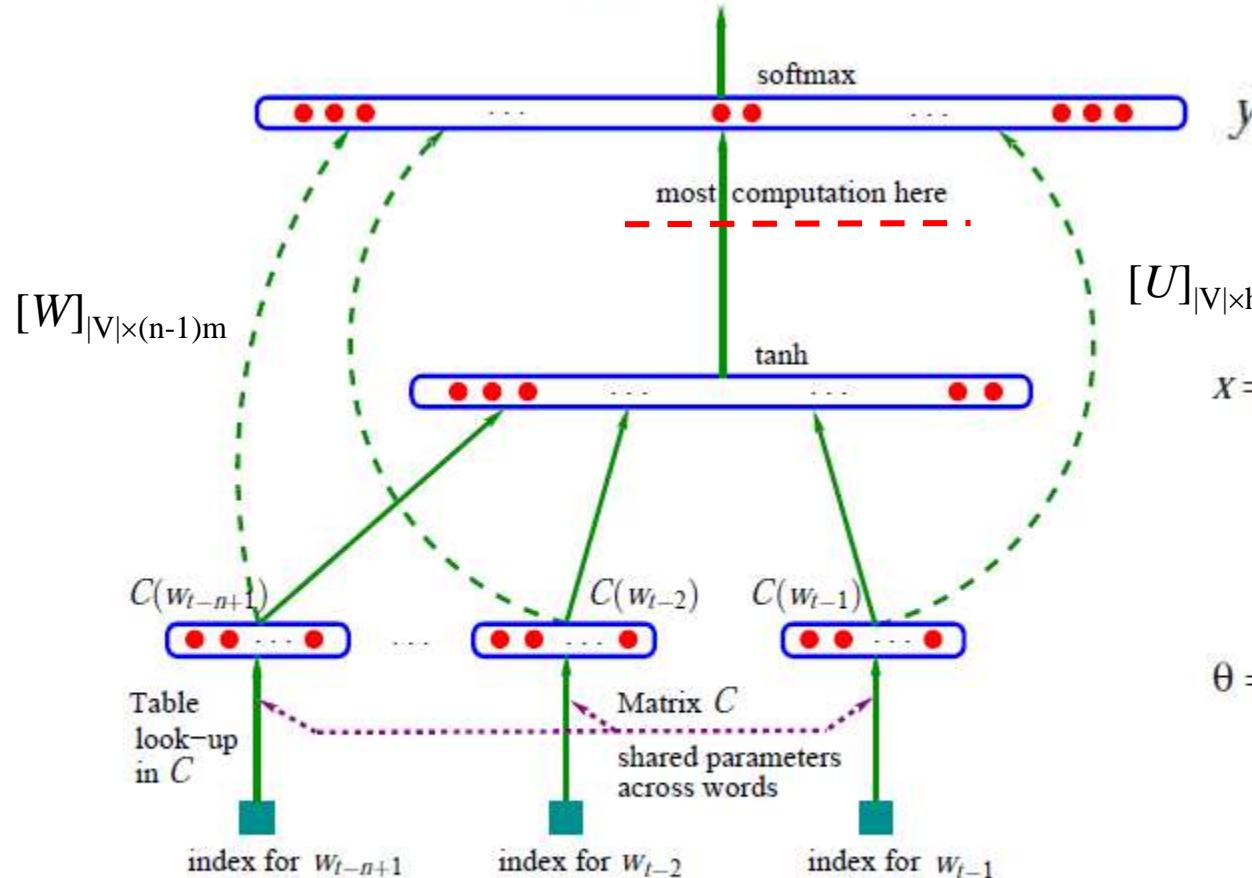
Neural Model

Output layer

$$i\text{-th output} = P(w_t = i \mid \text{context})$$

Hidden layer

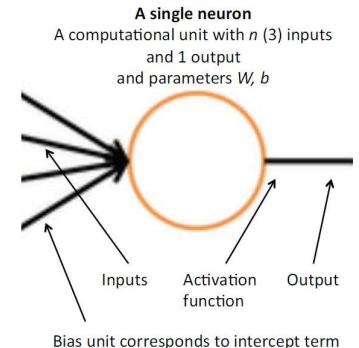
Input layer



$$\hat{P}(w_t | w_{t-1}, \dots, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}.$$

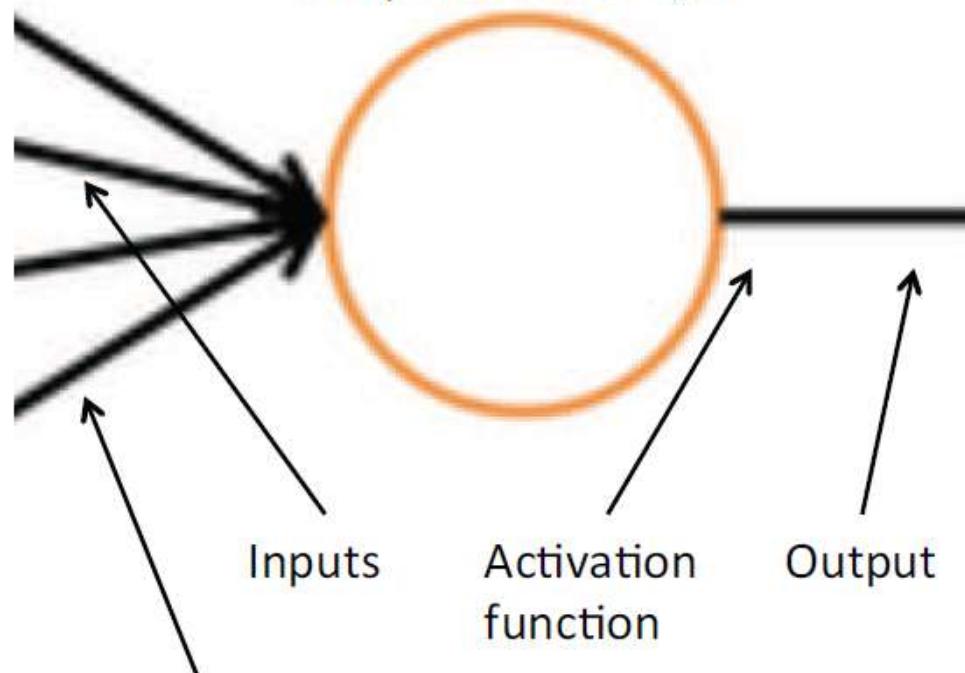
$$y = b + Wx + Utanh(d + Hx)$$

Neural architecture: $f(i, w_{t-1}, \dots, w_{t-n+1}) = g(i, C(w_{t-1}), \dots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i -th word feature vector.



A single neuron

A computational unit with n (3) inputs
and 1 output
and parameters W, b



Bias unit corresponds to intercept term

A Neural Network Language Model

- Training is achieved by looking for θ that **maximizes** the training corpus penalized log-likelihood:

$$L = \frac{1}{T} \sum_t \log f(w_{t-(n-1)}, \dots, w_{t-1}, w_t; \theta) + R(\theta)$$

where $R(\theta)$ is a regularization term (weight decay penalty)

T : total number of tokens
(total number of training instances)
Append n-1 dummy start symbols to the beginning of the text of T words

- The neural network computes the following function, with a softmax output layer, which guarantees positive probabilities summing to 1:

$$P(w_t | w_{t-1}, \dots, w_{t-(n-1)}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

Issues

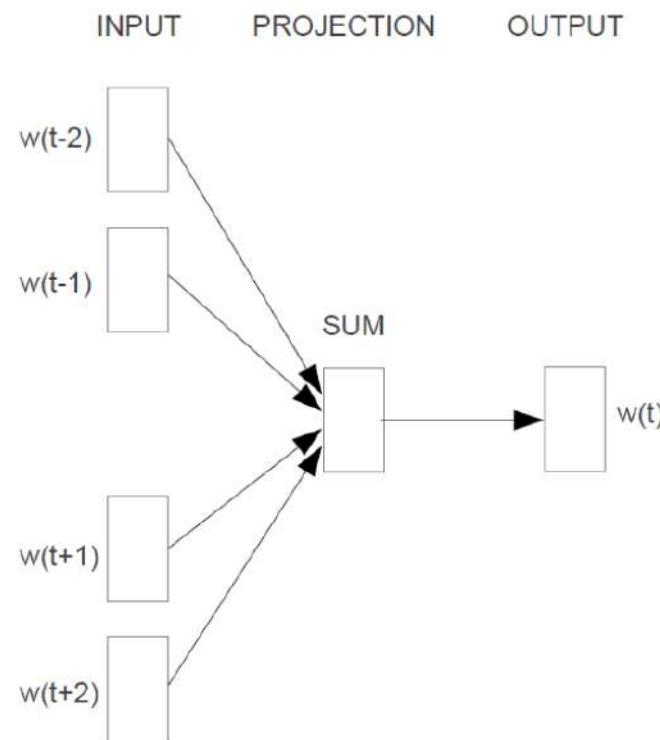
- The main observation from the previous works was that most of the complexity is caused by the non-linear hidden layer in the model.
- Simpler models that might not be able to represent the data as precisely as neural networks, but can possibly be trained on much more data efficiently.
- CBOW and Skip-Gram

T. Mikolov, and *et al.*, "Efficient estimation of word representations in vector space," in *Proc. of Workshop at ICLR*, 2013.
T. Mikolov, and *et al.*, "Distributed representations of words and phrases and their compositionality," in *Proc. of NIPS*, 2013.

The Continuous Bag-Of-Word Model

- The training objective of the CBOW model is to find word representations that are useful for predicting the current word by its surrounding words

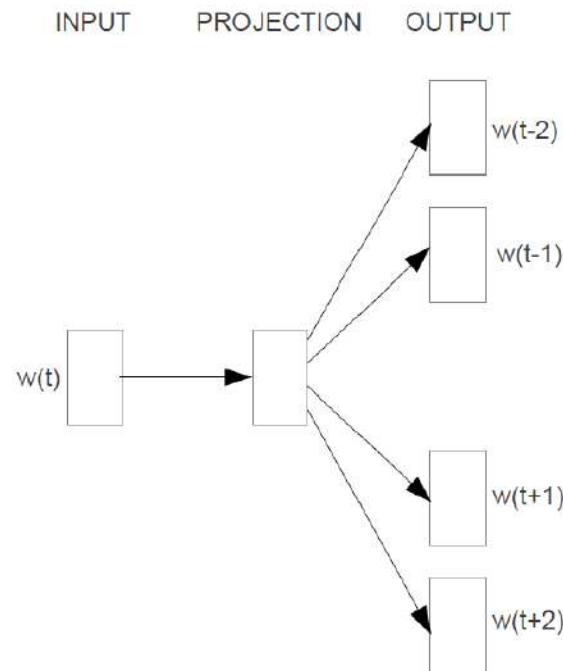
$$L = \left| \sum_{t=1}^T \log P(w_t | W_{t+c}^{t-c} - w_t) \right|$$



The Skip-gram Model

- The training objective of the Skip-gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document

$$L = \sum_{t=1}^T \sum_{\substack{-c \leq j \leq c \\ j \neq 0}} \log P(w_{t+j} | w_t)$$



Word Representation

- One-hot representation/One-of-N coding
 - In vector space terms, this is a vector with one 1 and a lot of zeroes
 - motel: [0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 - hotel: [0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 - [0 0 0 0 0 0 0 0 0 1 0 0 0 0] **and** [0 0 0 0 0 0 1 0 0 0 0 0 0 0]=0
- Count-based representation
 - Explicit representation: construct a high dimensional sparse matrix M , where each row represents a word w in the vocabulary and each column a potential context.
 - Context: sentence, paragraph, document
 - Singular Value Decomposition (SVD) of term-context matrix
 - Term-document matrix when context is document
- Prediction-based (neural) embeddings
 - Continuous Bag-of-Words (CBOW)
 - Skip-Grams
 - Global Vectors (GloVe)

Word embedding

- The collective name for a set of **language modeling** and **feature learning** techniques in natural language processing where **words** or **phrases** from the vocabulary are mapped to **vectors of real numbers** in a low-dimensional space relative to the vocabulary size ("continuous space")

Word Vector Representation Methods

- Dimension reduction on word co-occurrence matrix
 - Latent Semantic Analysis (LSA)
 - Principal Component Analysis (PCA)
 - Latent Dirichlet Allocation (LDA)
- Neural networks (neural embeddings, embeddings)
 - CBOW (Continuous Bag-of-Word)
 - Skip-Gram
 - GloVe (Global Vectors for Word Representation)
 - CWINDOW
 - structured skip-gram
 - word to vector → phrase to vector → document to vector
- Explicit Vector Space Representation (bag of contexts)

Distributed Representation of Words

Learning Word Representations

- Distributional hypothesis: words with similar meanings are likely to appear in similar contexts
 - Meaning as use!
 - You shall know a word by the company it keeps. (J.R. Firth, 1957)
- Attributional similarities between vocabulary items: words that appear in similar contexts will be close to each other in the projected space
- You can get a lot of value by representing a word by means of its neighbors

government debt problems turning into banking crises as has happened in
saying that Europe needs unified banking regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

Neural Probabilistic Language Model

- A conditional probability distribution over words in V for the next word w_t

$$p(w_t | w_{t-1}, \dots, w_{t-(n-1)}) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}$$

where

$$y = b + Wx + Utanh(d + Hx)$$

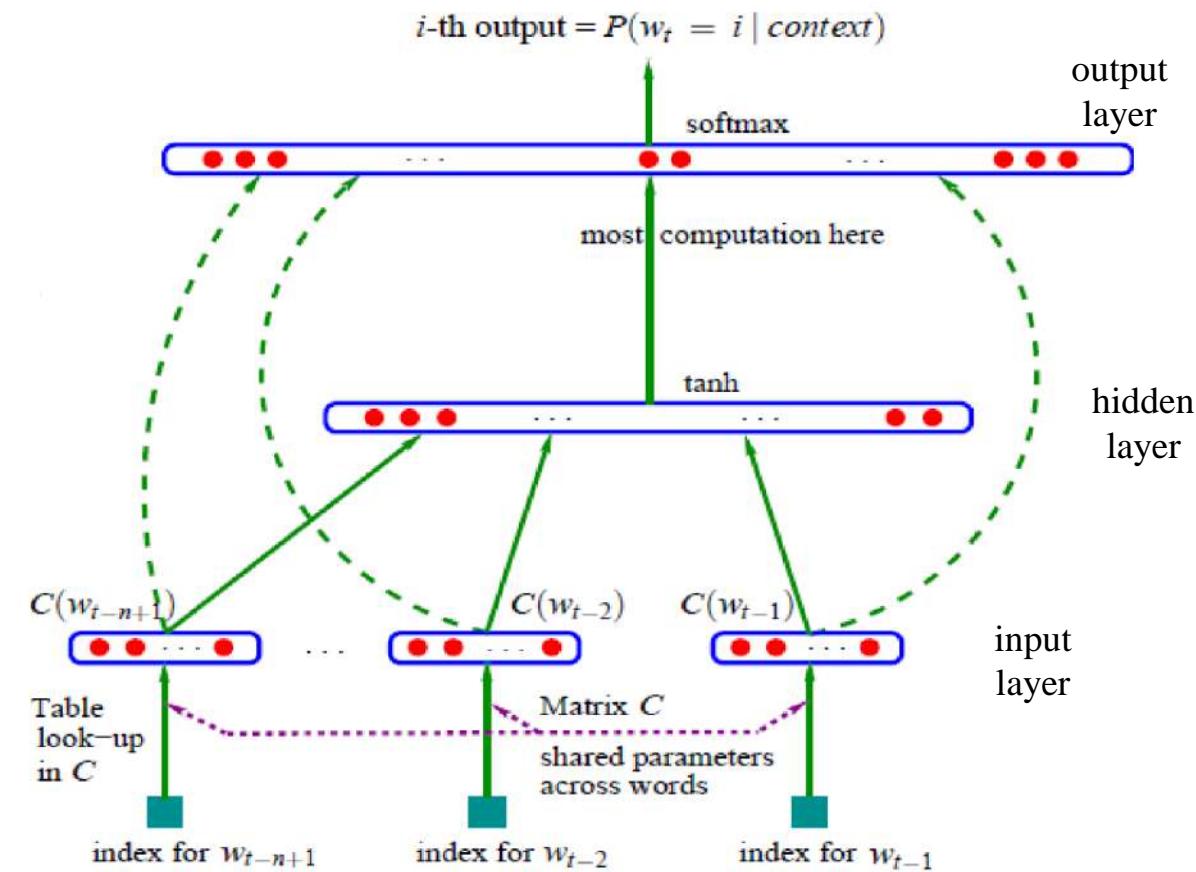
$$x = (C(w_{t-1}), C(w_{t-2}), \dots, C(w_{t-(n-1)}))$$

$$\theta = (b, d, W, U, H, C)$$

red: model parameters, green: vector representation

Maximize the average (regularized) log-likelihood

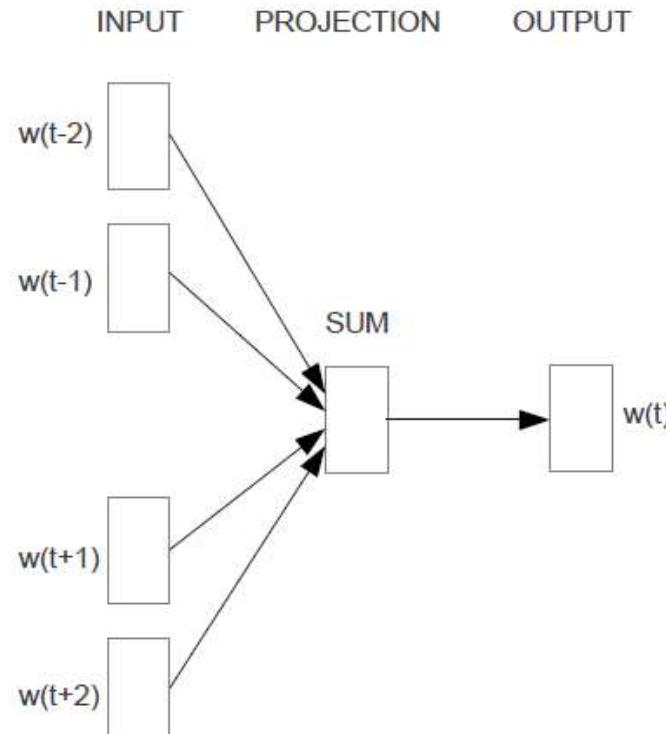
$$L = \frac{1}{T} \sum_t \log f(w_t, w_{t-1}, \dots, w_{t-(n-1)}; \theta)$$



Distributed Vector Representation of Words

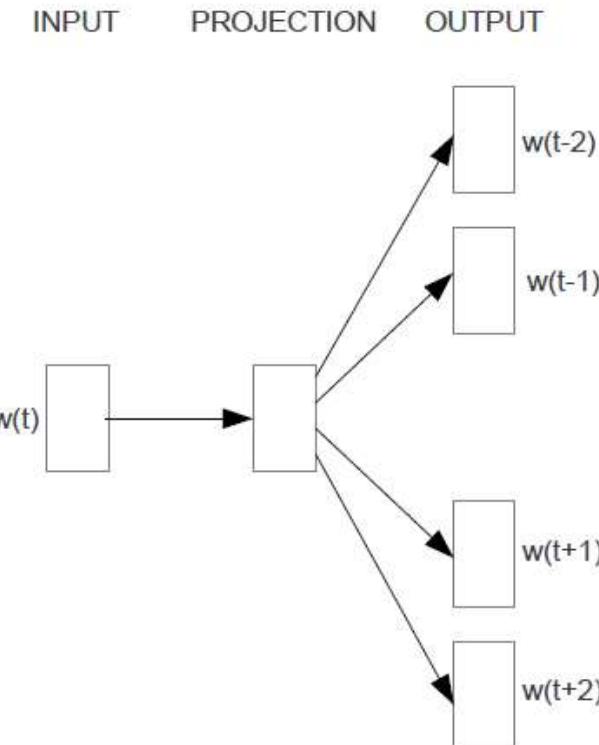
- Predict a word given the other words in a context
 - Every word is mapped to a unique vector, represented by a column in a matrix C .
 - The column is indexed by position of the word in the vocabulary.
 - The **concatenation** or **sum** of the vectors is then used as features for prediction of the next word in a sentence.

Word Representations in Vector Space



CBOW

predict the current word based on the context



Skip-gram

predict surrounding words given the current word

Continuous Bag-of-Words (CBOW)

- Predict the current word based on the context
- $y = b + Wx$
- $\theta = (b, W, C)$

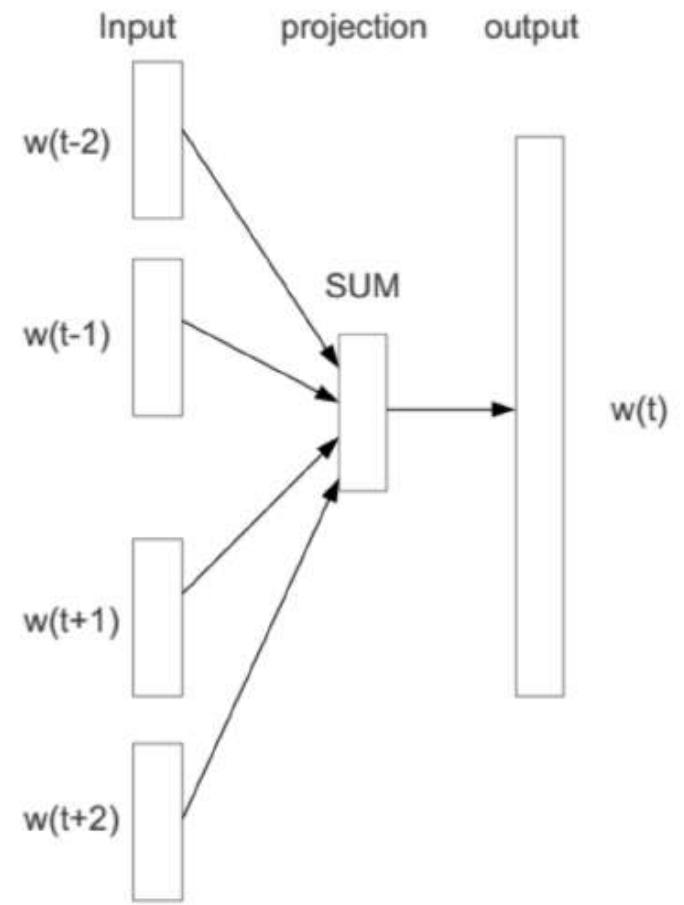
Notes:

The non-linear hidden layer is removed

The projection layer is shared for all words

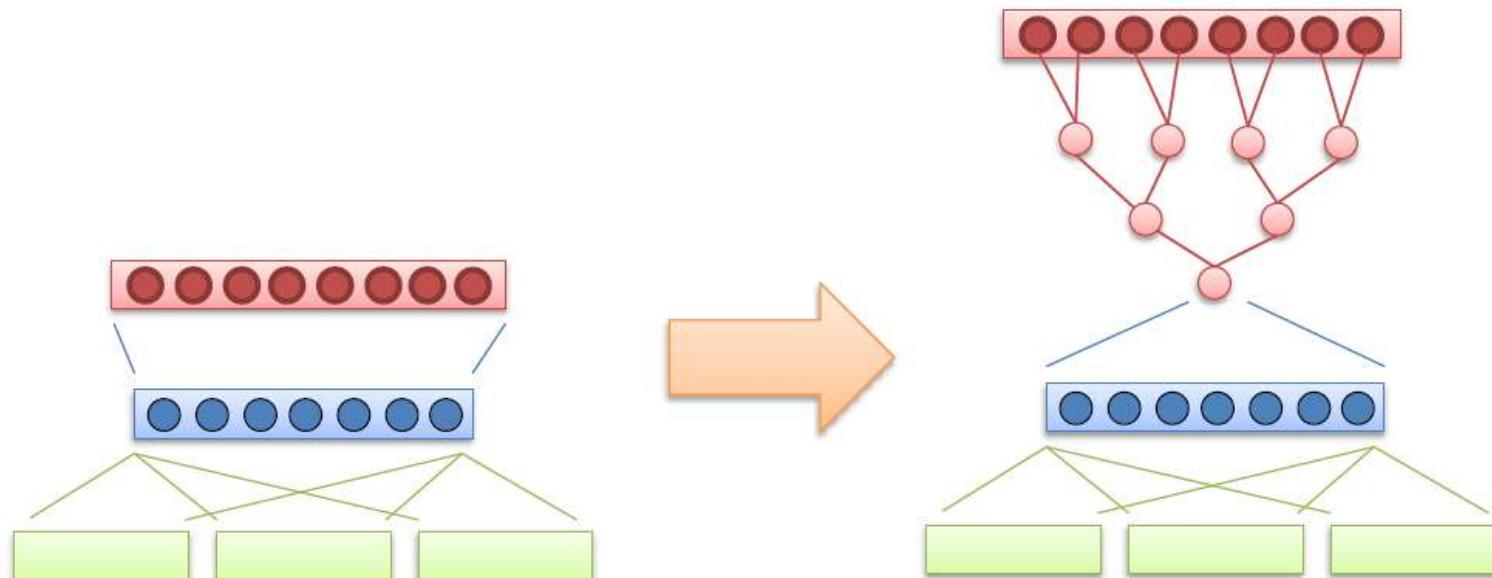
All words get projected into the same position (their vectors are averaged)

The order of words in the history does not influence the projection

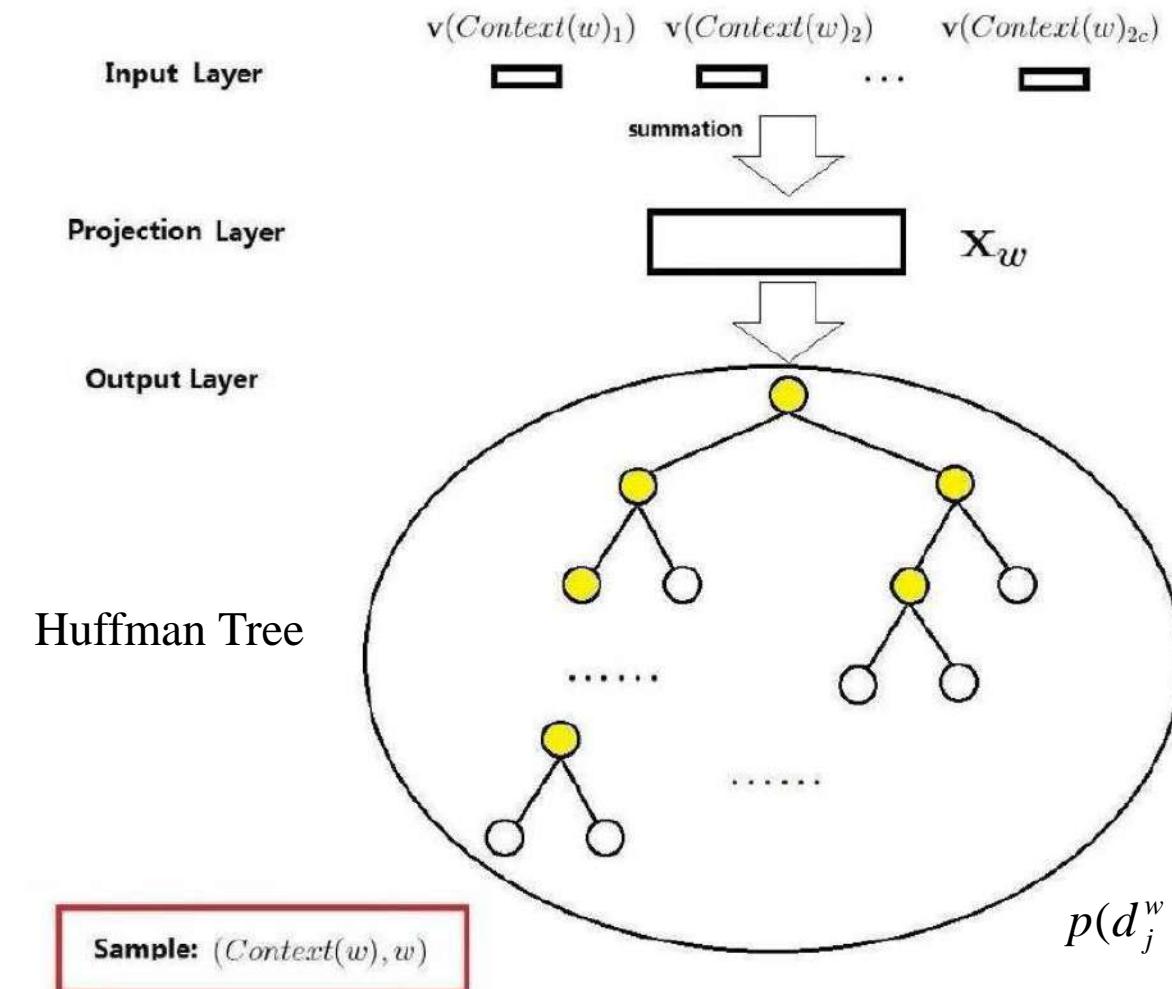


Hierarchical Softmax

- The hierarchical softmax uses a binary tree representation of the output layer with the words as its leaves and, for each node, explicitly represents the relative probabilities of its child nodes.



CBOW Using Hierarchical Softmax



internal node: logistic regression classifier (n-1 classifiers)
 external node: word (vocabulary size: n)

logistic regression classifier

$$h_\theta(\mathbf{x}) = \sigma(\theta^\top \mathbf{x}) = \frac{1}{1 + e^{-\theta^\top \mathbf{x}}}$$

$$y(\mathbf{x}) = \begin{cases} 1, & h_\theta(\mathbf{x}) \geq 0.5 \\ 0, & h_\theta(\mathbf{x}) < 0.5 \end{cases}$$

positive probability: $\sigma(\mathbf{x}_w^\top \theta) = \frac{1}{1 + e^{-\mathbf{x}_w^\top \theta}}$

negative probability: $1 - \sigma(\mathbf{x}_w^\top \theta)$

$$p(w | \text{Context}(w)) = \prod_{j=2}^{l^w} p(d_j^w | X_w, \theta_{j-1}^w)$$

path from root to w :
 1 (root), 2, ..., j (w)

classifiers: $\theta_1, \theta_2, \dots, \theta_{j-1}$

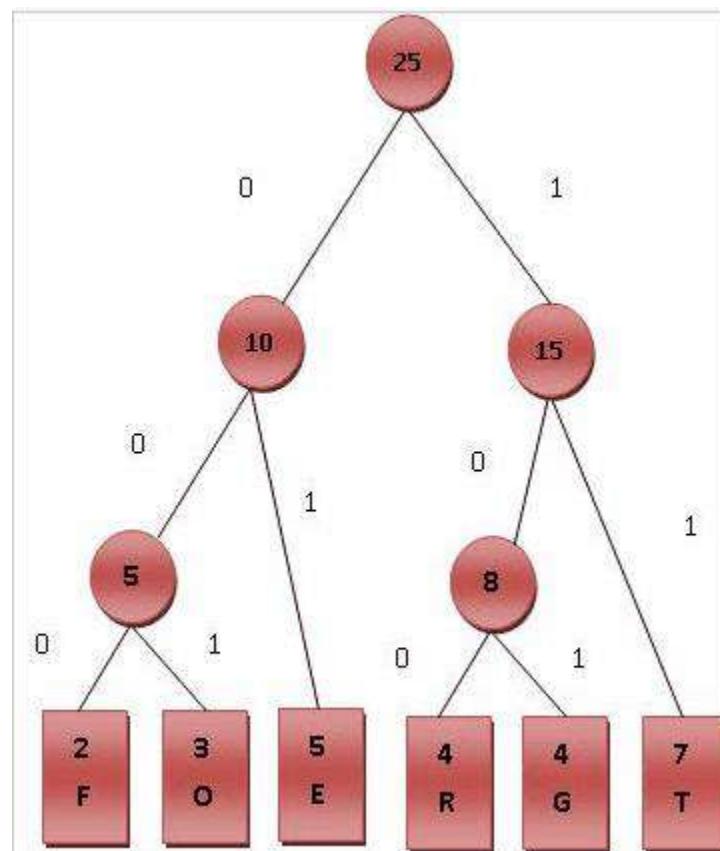
The probability to left (1) or
 right (0), in classifier θ_{j-1}

$$p(d_j^w | X_w, \theta_{j-1}^w) = \begin{cases} \sigma(X_w^\top \cdot \theta_{j-1}^w), d_j^w = 0 & \text{positive category} \\ 1 - \sigma(X_w^\top \cdot \theta_{j-1}^w), d_j^w = 1 & \text{negative category} \end{cases}$$

$$p(d_j^w | X_w, \theta_{j-1}^w) = [\sigma(X_w^\top \cdot \theta_{j-1}^w)]^{(1-d_j^w)} \cdot [1 - \sigma(X_w^\top \cdot \theta_{j-1}^w)]^{d_j^w}$$

Huffman Tree

Symbol	F	O	R	G	E	T
Frequency	2	3	4	4	5	7

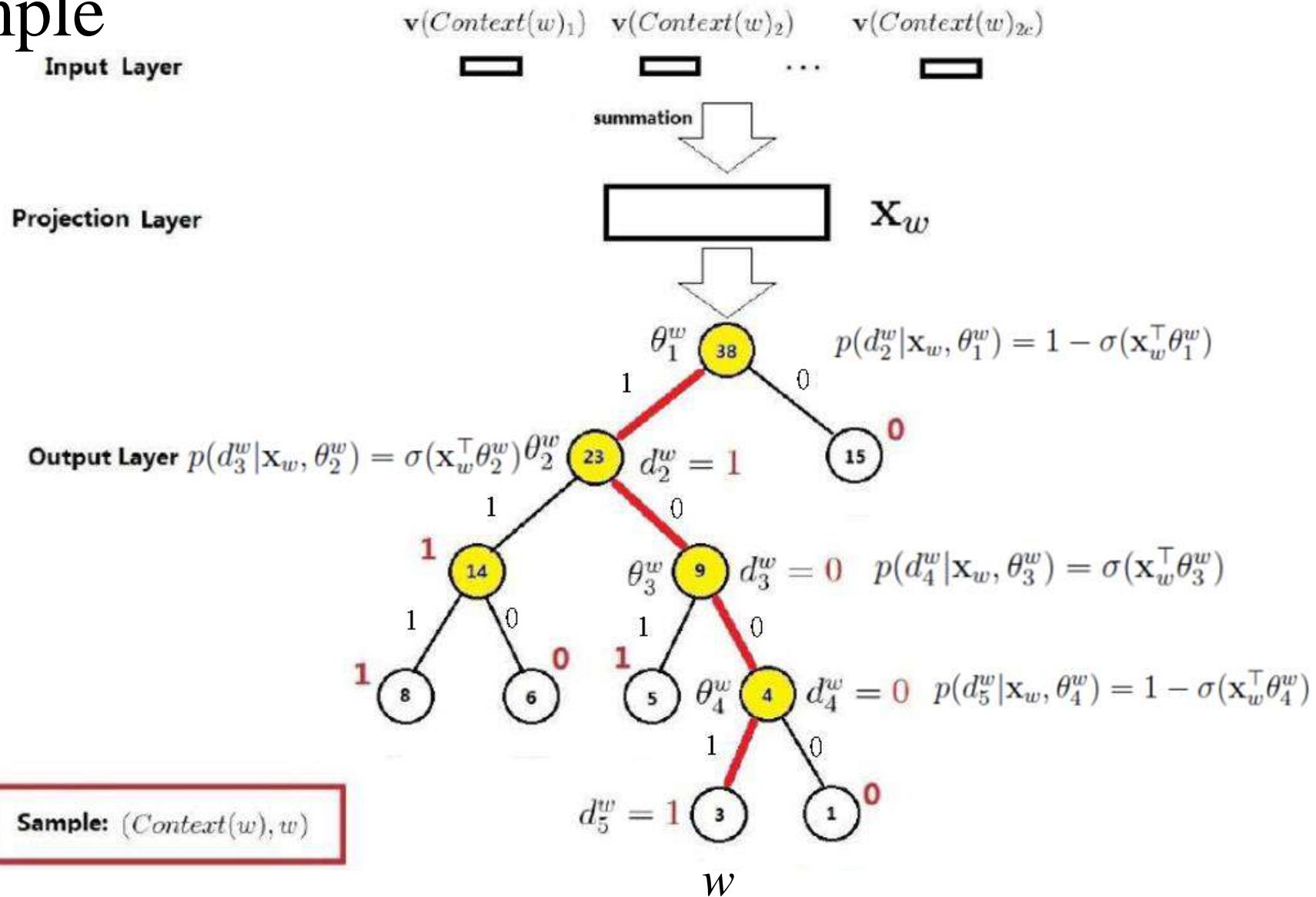


Symbol	F	O	R	G	E	T
Frequency	2	3	4	4	5	7
CODE	000	001	100	101	01	11

Note 0/1 is reversed in CBOW implementation.

Most frequent words, less number of codes (i.e., higher levels in the tree)

An Example



$$p(w | Context(w)) = (1 - \sigma(\mathbf{x}_w^\top \theta_1^w)) \times \sigma(\mathbf{x}_w^\top \theta_2^w) \times \sigma(\mathbf{x}_w^\top \theta_3^w) \times (1 - \sigma(\mathbf{x}_w^\top \theta_4^w))$$

CBOW Using Hierarchical Softmax

- Basic

$$p(w | \text{Context}(w)) = \prod_{j=2}^{l^w} p(d_j^w | X_w, \theta_{j-1}^w)$$
$$p(d_j^w | X_w, \theta_{j-1}^w) = \begin{cases} \sigma(X_w^T \cdot \theta_{j-1}^w), & d_j^w = 0 \\ 1 - \sigma(X_w^T \cdot \theta_{j-1}^w), & d_j^w = 1 \end{cases}$$
$$p(d_j^w | X_w, \theta_{j-1}^w) = [\sigma(X_w^T \cdot \theta_{j-1}^w)]^{1-d_j^w} \cdot [1 - \sigma(X_w^T \cdot \theta_{j-1}^w)]^{d_j^w}$$

- Given a corpus C consisting of S sentences,

$$L(X, \theta) = \prod_{s \in C} \prod_{w \in s} p(w | \text{Context}(w)) = \prod_{s \in C} \prod_{w \in s} \prod_{j=2}^{l^w} p(d_j^w | X_w, \theta_{j-1}^w)$$

- Log likelihood function

$$\log L(X, \theta) = \sum_{s \in C} \sum_{w \in s} \sum_{j=2}^{l^w} \log p(d_j^w | X_w, \theta_{j-1}^w)$$
$$= \sum_{s \in C} \sum_{w \in s} \sum_{j=2}^{l^w} [(1 - d_j^w) \cdot \log \sigma(X_w^T \cdot \theta_{j-1}^w) + d_j^w \cdot \log(1 - \sigma(X_w^T \cdot \theta_{j-1}^w))]$$

Stochastic Gradient Descent/Ascent

- Let

$$L(w, j) = (1 - d_j^w) \cdot \log \sigma(X_w^T \cdot \theta_{j-1}^w) + d_j^w \cdot \log(1 - \sigma(X_w^T \cdot \theta_{j-1}^w))$$

$$\begin{aligned} [\log \sigma(x)]' &= 1 - \sigma(x) \\ [\log(1 - \sigma(x))]' &= -\sigma(x) \end{aligned}$$

$$\frac{\partial L(w, j)}{\partial \theta_{j-1}^w} = [1 - d_j^w - \sigma(X_w^T \cdot \theta_{j-1}^w)] \cdot X_w$$

$$\frac{\partial L(w, j)}{\partial X_w} = [1 - d_j^w - \sigma(X_w^T \cdot \theta_{j-1}^w)] \cdot \theta_{j-1}^w$$

$$\begin{aligned} \frac{\partial L(w, j)}{\partial \theta_{j-1}^w} &= \frac{\partial}{\partial \theta_{j-1}^w} \{(1 - d_j^w) \cdot \log \sigma(X_w^T \cdot \theta_{j-1}^w) + d_j^w \cdot \log(1 - \sigma(X_w^T \cdot \theta_{j-1}^w))\} \\ &= \{(1 - d_j^w)[1 - \sigma(X_w^T \cdot \theta_{j-1}^w)]X_w - d_j^w \cdot \sigma(X_w^T \cdot \theta_{j-1}^w)X_w\} \\ &= \{(1 - d_j^w)[1 - \sigma(X_w^T \cdot \theta_{j-1}^w)] - d_j^w \cdot \sigma(X_w^T \cdot \theta_{j-1}^w)\}X_w \\ &= [1 - d_j^w - \sigma(X_w^T \cdot \theta_{j-1}^w)] \cdot X_w \end{aligned}$$

- Update formula

$$\theta_{j-1}^w := \theta_{j-1}^w + \eta \cdot \frac{\partial L(w, j)}{\partial \theta_{j-1}^w} \quad \eta: \text{learning rate}$$

$$V(\tilde{w}) := V(\tilde{w}) + \eta \cdot \sum_{j=2}^{l^w} \frac{\partial L(w, j)}{\partial X_w}, \tilde{w} \in \text{Context}(w)$$

Skip-gram Model

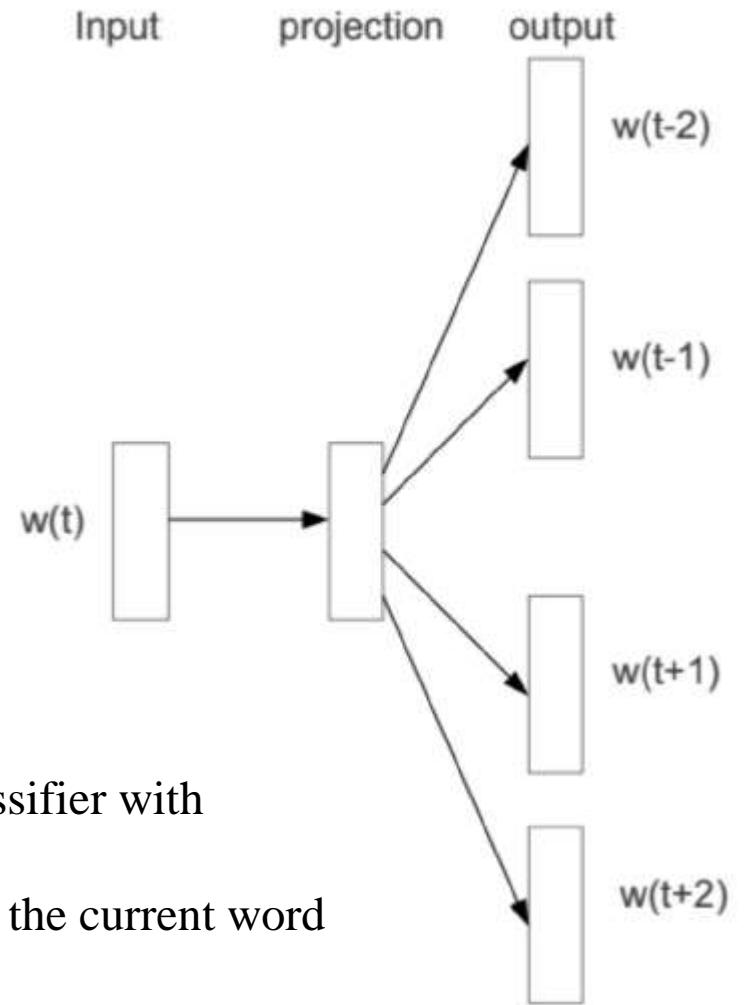
- Predict the surrounding words
- $f = \prod_{-l \leq j \leq l, j \neq 0} p(w_{t+j} | w_t)$

where

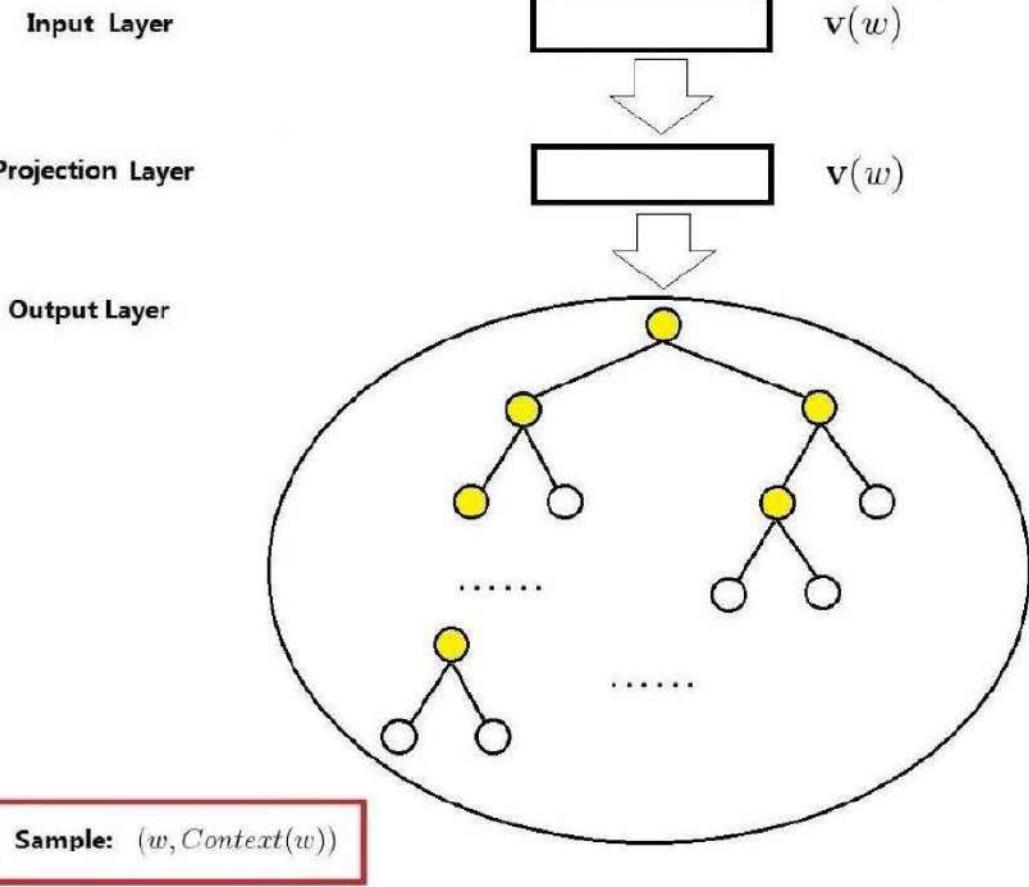
$$p(w_{t+j} | w_t) = \frac{\exp(y_{w_{t+j}}^T y_{w_t})}{\sum_i^{|V|} \exp(y_{w_i}^T y_{w_t})}$$

- $y_i = C(w_i)$
- $\theta = C$

Use each current word as an input to a log-linear classifier with continuous projection layer
Predict words within a certain range before and after the current word



Skip-Gram using Hierarchical Softmax



$$p(\text{Context}(w)|w) = \prod_{u \in \text{Context}(w)} p(u|w)$$

$$p(u|w) = \prod_{j=2}^{l^u} p(d_j^u | \mathbf{v}(w), \theta_{j-1}^u)$$

$$p(d_j^u | \mathbf{v}(w), \theta_{j-1}^u) = [\sigma(\mathbf{v}(w)^T \cdot \theta_{j-1}^u)]^{1-d_j^u} \cdot [1 - \sigma(\mathbf{v}(w)^T \cdot \theta_{j-1}^u)]^{d_j^u}$$

$$L(X, \theta) = \prod_{w \in C} p(\text{Context}(w) | w) = \prod_{w \in C} \prod_{u \in \text{Context}(w)} \prod_{j=2}^{l^u} p(d_j^u | v(w), \theta_{j-1}^u)$$

$$\log L(X, \theta) = \sum_{w \in C} \log \prod_{u \in \text{Context}(w)} \prod_{j=2}^{l^u} p(d_j^u | v(w), \theta_{j-1}^u)$$

$$= \sum_{w \in C} \log \prod_{u \in \text{Context}(w)} \prod_{j=2}^{l^u} [\sigma(v(w)^T \cdot \theta_{j-1}^u)]^{1-d_j^u} \cdot [1 - \sigma(v(w)^T \cdot \theta_{j-1}^u)]^{d_j^u}$$

$$= \sum_{w \in C} \sum_{u \in \text{Context}(w)} \sum_{j=2}^{l^u} [(1 - d_j^u) \cdot \log \sigma(v(w)^T \cdot \theta_{j-1}^u) + d_j^u \cdot \log(1 - \sigma(v(w)^T \cdot \theta_{j-1}^u))]$$

Update Formula

$$\begin{aligned}
\frac{\partial L(w, u, j)}{\partial \theta_{j-1}^u} &= \frac{\partial}{\partial \theta_{j-1}^u} \{(1 - d_j^u) \cdot \log \sigma(v(w)^T \cdot \theta_{j-1}^u) + d_j^u \cdot \log(1 - \sigma(v(w)^T \cdot \theta_{j-1}^u))\} \\
&= (1 - d_j^u)[1 - \sigma(v(w)^T \cdot \theta_{j-1}^u)]v(w) - d_j^u \cdot \sigma(v(w)^T \cdot \theta_{j-1}^u)v(w) \\
&= \{(1 - d_j^u)[1 - \sigma(v(w)^T \cdot \theta_{j-1}^u)] - d_j^u \cdot \sigma(v(w)^T \cdot \theta_{j-1}^u)\}v(w) \\
&= [1 - d_j^u - \sigma(v(w)^T \cdot \theta_{j-1}^u)] \cdot v(w)
\end{aligned}$$

$$[\log \sigma(x)]' = 1 - \sigma(x)$$

$$[\log(1 - \sigma(x))]' = -\sigma(x)$$

$$\frac{\partial L(w, u, j)}{\partial v(w)} = [1 - d_j^u - \sigma(v(w)^T \cdot \theta_{j-1}^u)] \cdot \theta_{j-1}^u$$

$$\theta_{j-1}^u := \theta_{j-1}^u + \eta \cdot \frac{\partial L(w, u, j)}{\partial \theta_{j-1}^u} = \theta_{j-1}^u + \eta \cdot [1 - d_j^u - \sigma(v(w)^T \cdot \theta_{j-1}^u)] \cdot v(w)$$

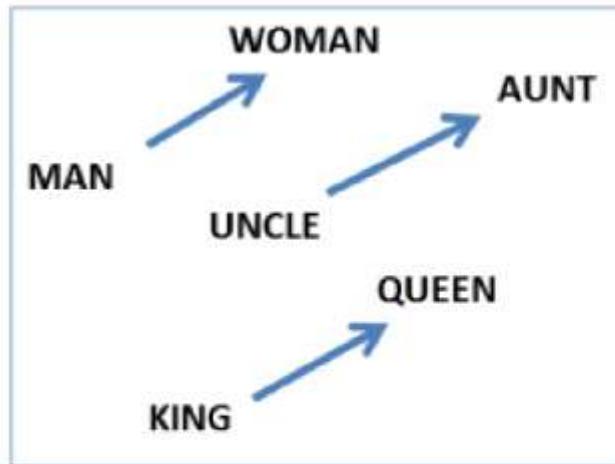
$$v(w) := v(w) + \eta \sum_{u \in \text{Context}(w)} \sum_{j=2}^{l^u} \frac{\partial L(w, u, j)}{\partial v(w)} = v(w) + \eta \sum_{u \in \text{Context}(w)} \sum_{j=2}^{l^u} [1 - d_j^u - \sigma(v(w)^T \cdot \theta_{j-1}^u)] \cdot \theta_{j-1}^u$$

word2vec

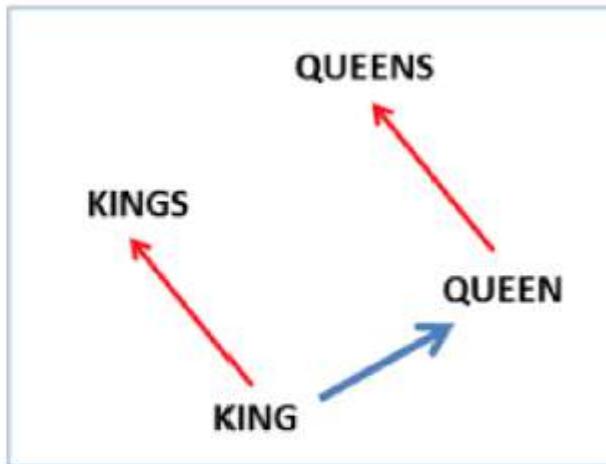
- word2vec is **not** a single algorithm
- It is a **software package** for representing words as vectors
 - Two distinct models
 - CBOW
 - Skip-Gram
 - Various training methods
 - Negative Sampling
 - Hierarchical Softmax
 - A rich preprocessing pipeline
 - Dynamic Context Windows
 - Subsampling
 - Deleting Rare Words

Word Vector - Linguistic Regularities

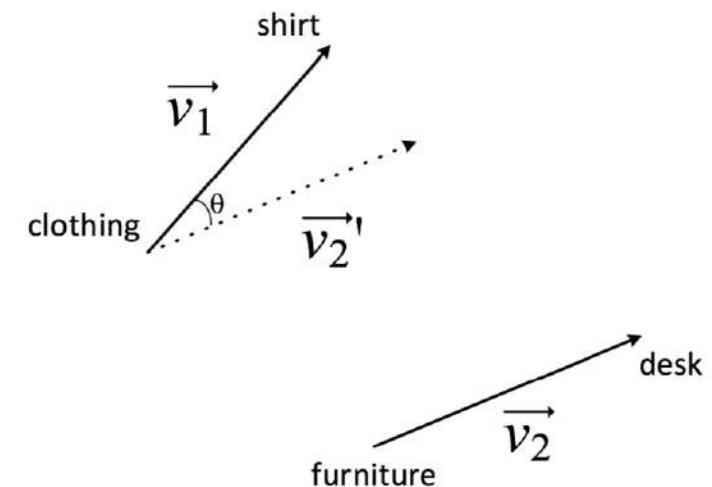
- Embeddings encode not only attributional similarities between words, but also similarities between pairs of words (relational similarity)
- One can do nearest neighbor search around result of vector operation



→ semantic



→ syntactic

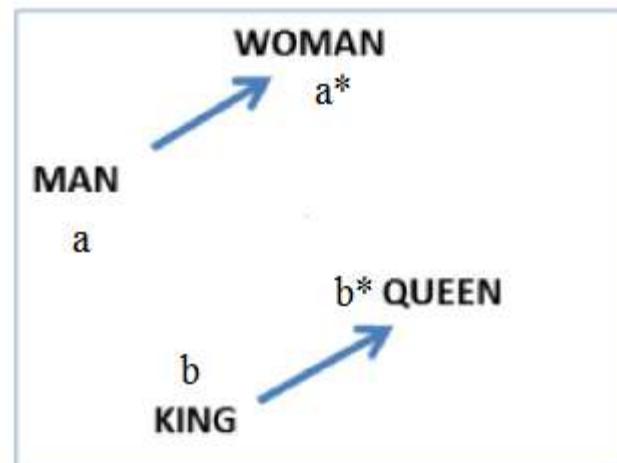


“King – man + woman” and obtain “Queen”

Vector Arithmetic

- Analogy Questions
 - Given two pairs of words that share a relation (e.g. “man:woman”, “king:queen”), identify the identity of the fourth word, i.e., (“queen”) is hidden based on the other three (e.g. answering the question: “man is to woman as king is to — ?”).
 - $a:a^* = b:b^*$
 - The vector of the hidden word b^* will be similar to the vector $b-a+a^*$

$$\arg \max_{b^* \in V} (\text{sim}(b^*, b - a + a^*))$$



$$\begin{aligned} &\mathbf{WOMAN} (a^*) - \mathbf{MAN} (a) \\ &= b^* - \mathbf{KING} (b) \end{aligned}$$

$$b^* = b - a + a^*$$

Similarity Measurement

- Word Analogies

$$\cos(u, v) = \frac{u \cdot v}{\|u\| \|v\|}$$

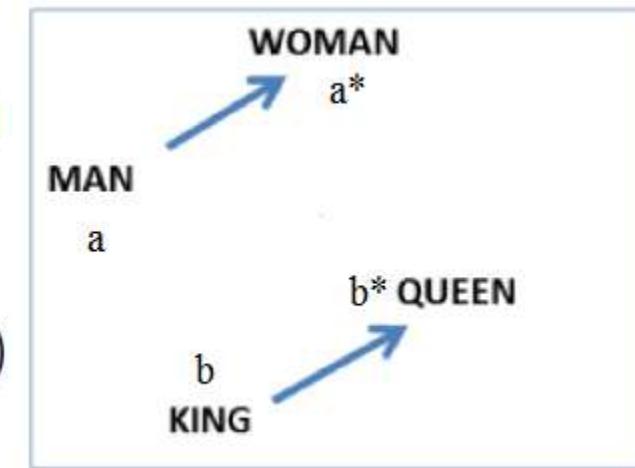
$$\arg \max_{b^* \in V} (\cos(b^*, b - a + a^*))$$

- PAIR DIRECTION

$$\arg \max_{b^* \in V} (\cos(b^* - b, a^* - a))$$

- 3COSADD method

$$\arg \max_{b^* \in V} (\cos(b^*, b) - \cos(b^*, a) + \cos(b^*, a^*))$$



seeking a word b^* which is similar to b and a^* but is different from a

Compositionality of Multiword Expressions

- Estimate the compositionality of an MWE
 - Based on the similarity between the expression and its component words in vector space
- Methods
 - Count-Based Distributional Similarity
 - word2vec

$$\begin{aligned}comp_1(\text{MWE}) &= \alpha sim(\text{MWE}, \mathbf{C}_1) \\&+ (1 - \alpha) sim(\text{MWE}, \mathbf{C}_2)\end{aligned}$$

$$comp_2(\text{MWE}) = sim(\text{MWE}, \mathbf{C}_1 + \mathbf{C}_2)$$

Sense Embedding

Ting-Yu Yen, Yang-Yin Lee, Hen-Hsen Huang and Hsin-Hsi Chen
Joint Sense Retrofitting from Contextual and Ontological Information
(WWW 2018)

Introduction

- Most of the word embedding models use one vector to represent a word, and are problematic in some natural language processing applications that require sense level representation (e.g., word sense disambiguation)

$$\hat{q}_{bank} = (0.3, 0.1, \dots, 0.5)$$

- Word2vec (Mikolov, et al., 2013)
- GloVe (Pennington, Socher, & Manning, 2014)

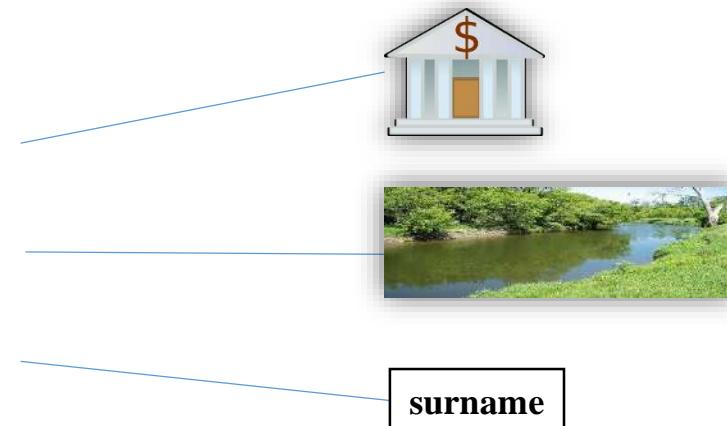
Introduction

- We propose a joint sense embedding learning model that retrofits a pre-trained word embedding using contextual and ontological information, from

$$\hat{q}_{bank} = (0.3, 0.1, \dots, 0.5)$$

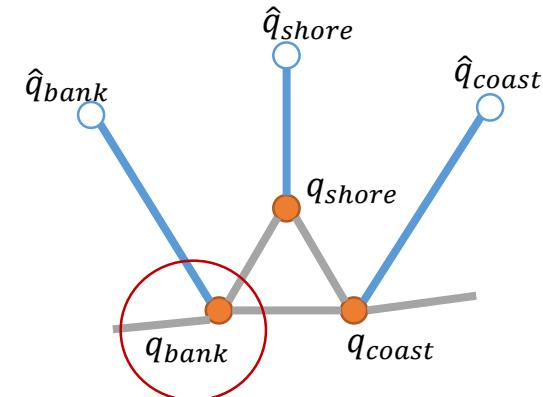
- To

$$\begin{cases} \hat{q}_{bank_1} = (0.3, 0.1, \dots, 0.5) \\ \hat{q}_{bank_2} = (0.4, 0.2, \dots, 0.3) \\ \hat{q}_{bank_3} = (0.5, 0.1, \dots, 0.2) \end{cases}$$



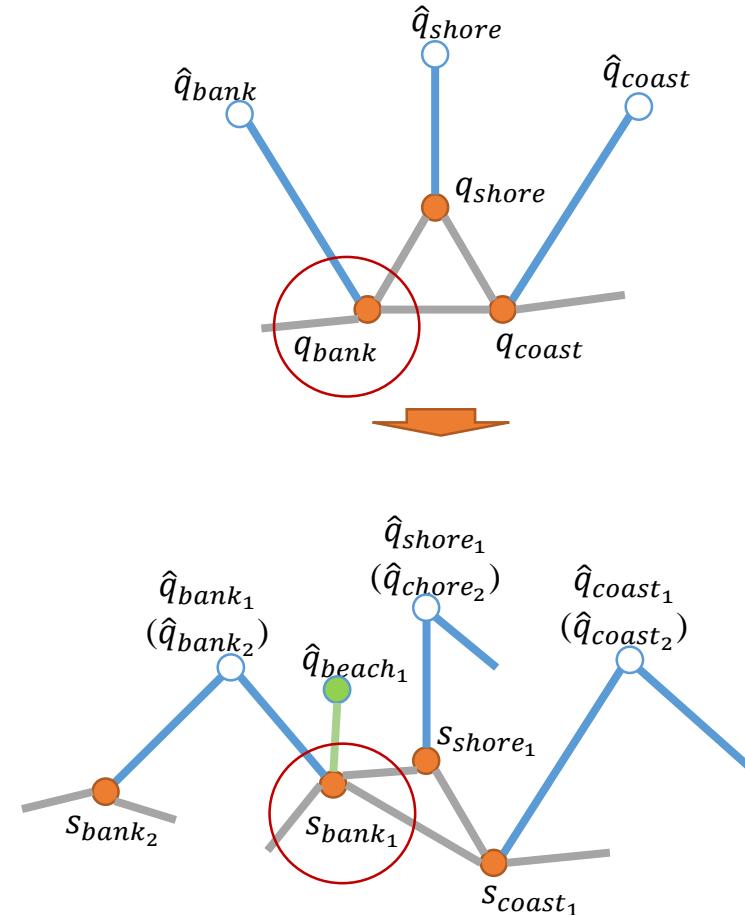
Word Level Retrofitting

- $V = \{w_1, \dots, w_n\}$: a vocabulary of a pre-trained word embedding
- The matrix \hat{Q} will be the pre-trained collection of vector representations $\hat{q}_i \in \mathbb{R}^d$
 - d : the dimensionality of a word vector



Joint Sense Retrofitting

- $\Omega = (T, E)$: an ontology that contains the semantic relationship
 - $T = \{t_1, \dots, t_m\}$: the set of all senses
- The edge $(i, j) \in E$ indicates a semantic relationship of interest (e.g., synonym) between t_i and t_j
- Split E into two disjoint subsets E_s and E_c
 - $(i, j) \in E_s$ if and only if there is more than one sense of the word form of t_j
 - $(i, j) \in E_c$ if and only if t_j has only one sense
- \hat{q}_{t_j} : denote the word form vector of t_j

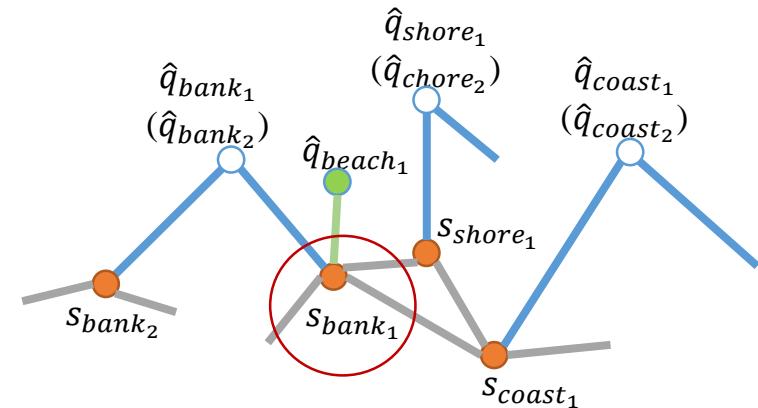


Joint Sense Retrofitting

The objective of our model is to learn a new matrix $S = (s_1, \dots, s_m)$ such that each new sense vector is close to

1. Its word form vertex
 2. Its contextual neighbors
 3. Its sense neighbors
- The objective to be minimized is:

$$\sum_{i=1}^m \left[\underbrace{\alpha_i \|s_i - \hat{q}_{t_i}\|^2}_1 + \underbrace{\sum_{(i,j) \in E_c} \beta_{ij} \|s_i - \hat{q}_{t_j}\|^2}_2 + \underbrace{\sum_{(i,k) \in E_s} \gamma_{ik} \|s_i - s_k\|^2}_3 \right]$$



- α, β, γ : control the relative strength of the sense relations

Joint Sense Retrofitting

- Initially, the sense vectors are set to their corresponding word form vectors
 - $s_i \leftarrow \hat{q}_{t_i} \forall i$
- Then in the following iterations, the updating formula for s_i would be:

$$s_i = \frac{\sum_{k:(i,k) \in E_S} \gamma_{ik} s_k + \sum_{j:(i,j) \in E_C} \beta_{ij} \hat{q}_{t_j} + \alpha_i \hat{q}_{t_i}}{\sum_{k:(i,k) \in E_S} \gamma_{ik} + \sum_{j:(i,j) \in E_C} \beta_{ij} + \alpha_i}$$

Experiments

- Four benchmark datasets
 - MEN, Mturk, Rare Words (RW), and WordSim353 (WS353)
- Evaluation
 - Spearman's correlation between human score and AvgSim / MaxSim

$$\text{AvgSim}(w, w') \stackrel{\text{def}}{=} \frac{1}{K_w K_{w'}} \sum_{j=1}^{K_w} \sum_{k=1}^{K_{w'}} \cos(v_{w_j}, v_{w'_k})$$

$$\text{MaxSim}(w, w') \stackrel{\text{def}}{=} \max_{1 \leq j \leq K_w, 1 \leq k \leq K_{w'}} \cos(v_{w_j}, v_{w'_k})$$

Experiments

- Pre-trained word embedding model: GloVe
 - Trained on Wikipedia and Gigaword-5 (6B tokens, 400k vocab, uncased, 50d vectors)
- Ontologies
 - WordNet (**WN**): β s and γ s are set to 1.0 for synonyms and 0.5 for hypernyms or hyponyms
 - Roget's 21st Century Thesaurus (**Roget**): β s and γ s to 1.0, 0.6 and 0.3 for the strongest to the weakest synonyms
- α : the sum of β s and γ s
- Baseline models
 - GloVe (Pennington, Socher, & Manning, 2014)
 - retro (Jauhar, Dyer, & Hovy, 2015)

Results

	MEN	MTurk	RW	WS353
GloVe	65.7	61.9	30.3	50.3
retro-WN [5]	62.4 / 67.7	57.4 / 60.1	15.1 / 26.9	43.9 / 51.1
retro-Roget [5]	48.7 / 52.1	47.3 / 49.3	24.4 / 26.1	27.8 / 29.4
joint-WN	64.0 / 68.9	57.3 / 62.1	20.1 / 28.5	47.2 / 49.6
joint-Roget	66.5 / 67.5	62.0 / 62.6	32.3 / 32.5	50.9 / 52.8

$\rho \times 100$ of (MaxSim / AvgSim) on test datasets

2020年3月26日
自然語言處理課程

Neural Language Model

Revisit

Neural Network Methods for Natural Language Processing

Neural Probabilistic Language Model

- A conditional probability distribution over words in V for the next word w_t

$$p(w_t | w_{t-1}, \dots, w_{t-(n-1)}) = \frac{\exp(y_{w_t})}{\sum_i \exp(y_i)}$$

where

$$y = b + Wx + Utanh(d + Hx)$$

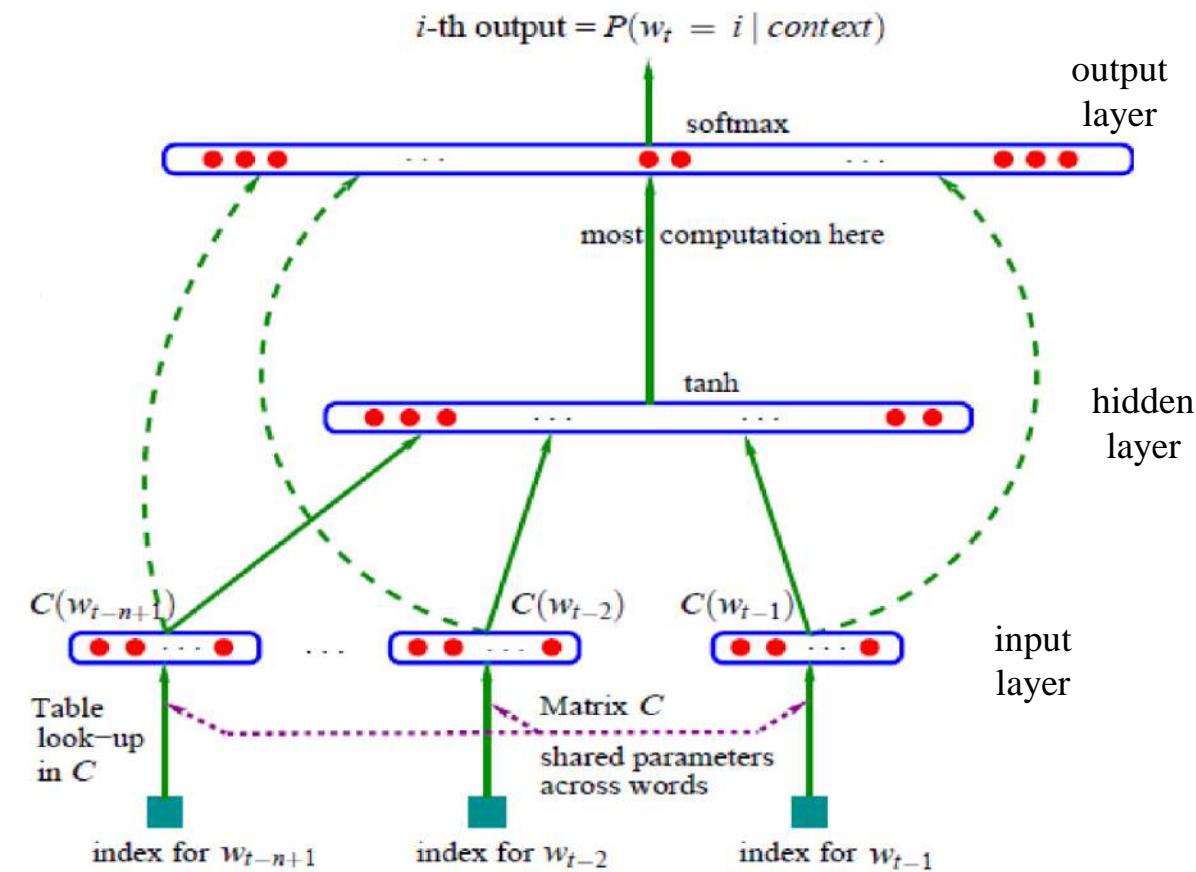
$$x = (C(w_{t-1}), C(w_{t-2}), \dots, C(w_{t-(n-1)}))$$

$$\theta = (b, d, W, U, H, C)$$

red: model parameters, green: vector representation

Maximize the average (regularized) log-likelihood

$$L = \frac{1}{T} \sum_t \log f(w_t, w_{t-1}, \dots, w_{t-(n-1)}; \theta)$$



Neural Language Model

The input to the neural network is a k -gram of words $w_{1:k}$, and the output is a probability distribution over the next word. The k context words $w_{1:k}$ are treated as a word window: each word w is associated with an embedding vector $v(w) \in \mathbb{R}^{d_w}$, and the input vector \mathbf{x} a concatenation of the k words:

$$\mathbf{x} = [v(w_1); v(w_2); \dots; v(w_k)]$$

The input \mathbf{x} is then fed to an MLP with one or more hidden layers:

(MultiLayer Perceptron)

$$\hat{\mathbf{y}} = P(w_i | w_{1:k}) = LM(w_{1:k}) = \text{softmax}(\mathbf{h}\mathbf{W}^2 + \mathbf{b}^2)$$

$$\mathbf{h} = g(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$$

$$\mathbf{x} = [v(w_1); v(w_2); \dots; v(w_k)]$$

$$v(w) = \mathbf{E}_{[w]}$$

Embedding Matrix

$$w_i \in V \quad \mathbf{E} \in \mathbb{R}^{|V| \times d_w} \quad \mathbf{W}^1 \in \mathbb{R}^{k \cdot d_w \times d_{\text{hid}}} \quad \mathbf{b}^1 \in \mathbb{R}^{d_{\text{hid}}} \quad \mathbf{W}^2 \in \mathbb{R}^{d_{\text{hid}} \times |V|} \quad \mathbf{b}^2 \in \mathbb{R}^{|V|}$$

Memory and Computational Efficiency

- Count-based language models
- From k -gram to $(k+1)$ -gram
 - $|V|^k \rightarrow |V|^{k+1}$
- Prediction-based language models
 - Move from k to $k + 1$ words
 - dimensions of the weight matrix W^1
 $k \cdot d_w \times d_{\text{hid}}$ to $(k + 1) \cdot d_w \times d_{\text{hid}}$
 - The softmax at the output layer
 - the matrix $W^2 \in \mathbb{R}^{d_{\text{hid}} \times |V|}$,
 - followed by $|V|$ exponentiations
 $\hat{y} = \text{softmax}(hW^2 + b^2)$

Most computation

$$\hat{y}_{[i]} = \frac{e^{(hW^2 + b^2)_{[i]}}}{\sum_j e^{(hW^2 + b^2)_{[j]}}}.$$

Memory and Computational Efficiency

- Count-based language models
- From $|V|$ to $|V|+1$
 - $|V|^k \rightarrow (|V|+1)^{k+1}$
- Prediction-based language models
- Move from $|V|$ to $|V| + 1$ words
 - dimensions of the input vocabulary E

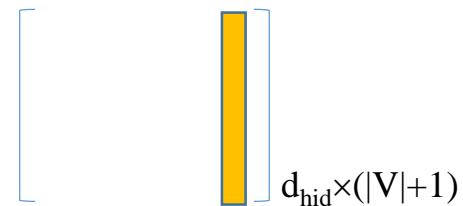


- the softmax at the output layer
 - $|V|+1$ expensive operation followed by $|V|$ exponentiations
- dimensions of the weight matrix W^2

$$\hat{y} = \text{softmax}(hW^2 + b^2)$$

$$\hat{y}_{[i]} = \frac{e^{(hW^2 + b^2)_{[i]}}}{\sum_j e^{(hW^2 + b^2)_{[j]}}}.$$

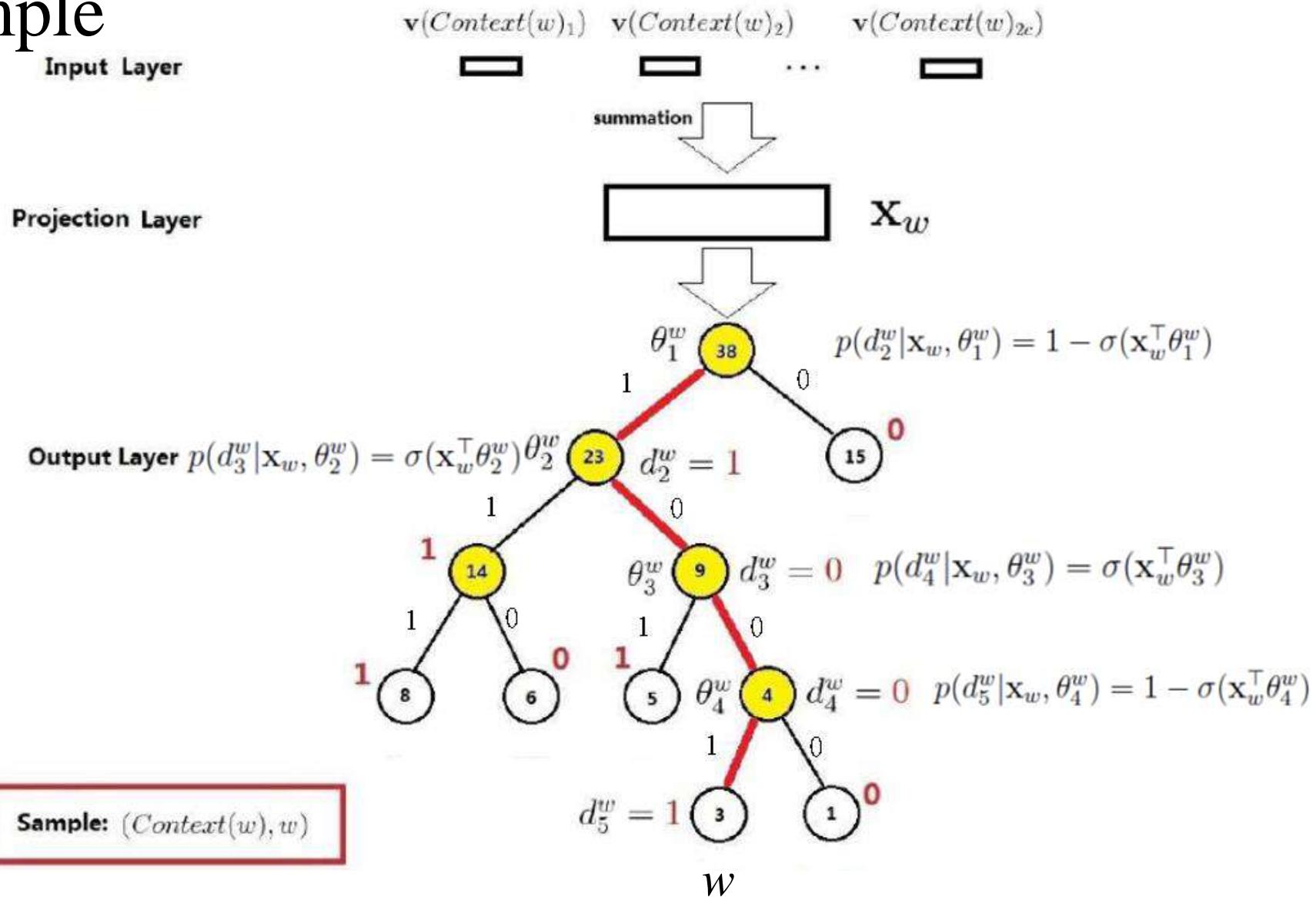
from $|V|$ to $|V| + 1$



Hierarchical softmax

- One of the approaches to deal with large output spaces
- Structuring the softmax computation as tree traversal
- The probability of each word as the product of branch selection decisions
- Compute the probability of a single word in $O(\log |V|)$ time rather than $O(|V|)$

An Example



$$p(w | Context(w)) = (1 - \sigma(\mathbf{x}_w^\top \theta_1^w)) \times \sigma(\mathbf{x}_w^\top \theta_2^w) \times \sigma(\mathbf{x}_w^\top \theta_3^w) \times (1 - \sigma(\mathbf{x}_w^\top \theta_4^w))$$

Desirable Properties

- Achieves much better perplexities than state-of-the-art traditional language models such as Kneser-Ney smoothed models
- The flexibility of the k -gram orders
 - Parameters are associated only with individual words, and not with k -grams
 - The words in different positions share parameters
 - The hidden layers of the models are in charge of finding informative word combinations
- Generalize across contexts

The flexibility of considering only parts of the conditioning context and the ability to generalize to unseen contexts

Limitations

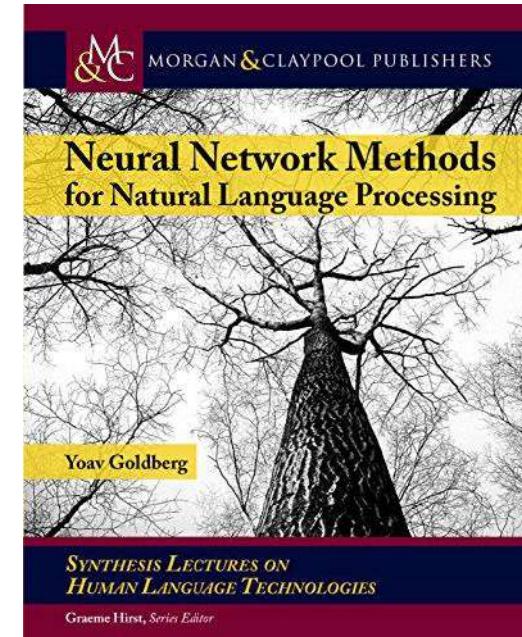
- Predicting the probability of a word in context is much more expensive than using a traditional language model
- Working with large vocabulary sizes and training corpora can become prohibitive
- Neural language models do not always improve the translation quality over traditional Kneser-Ney smoothed language models
 - A model is asked to assign a probability to the sentence **red horse**.
 - A traditional model will assign it a very low score, as it likely observed such an event only a few times, if at all.
 - A neural language model may have seen **brown horse**, **black horse** and **white horse**, and also learned independently that **black**, **white**, **brown** and **red** can appear in similar contexts.
 - Such a model will **assign a much higher probability** to the event **red horse**, which is undesired.

Lecture 5: Sequence Modeling

Neural Network Methods for Natural Language Processing

Text/Reference Books

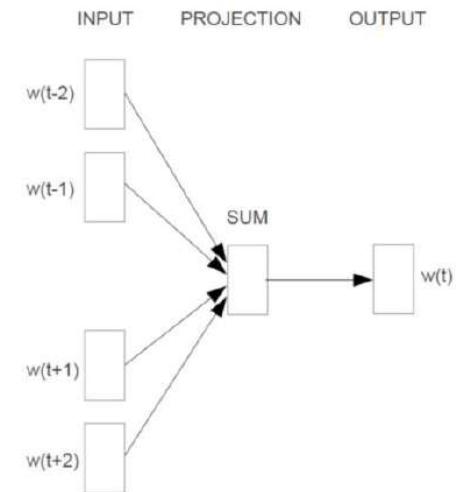
- Neural Network Methods in Natural Language Processing (Synthesis Lectures on Human Language Technologies)



Yoav Goldberg, A Primer on Neural Network Models for
Natural Language Processing, Journal of Artificial Intelligence
Research 57 (2016) 345–420

Sequences

- Sequence data
 - words: sequences of letters
 - sentences: sequences of words
 - documents: sequences of sentences
- CBOW
 - Use of vector concatenation and vector addition
 - Disregard the order of features
- RNN
 - Represent arbitrarily sized sequential inputs in a fixed-size vector
 - Allow for language models that do not make the Markov assumption, and condition the next word on the entire history



Function of RNN

takes as input an arbitrary length ordered sequence of n d_{in} -dimensional vectors $x_{1:n} = x_1, x_2, \dots, x_n$, ($x_i \in \mathbb{R}^{d_{in}}$) and returns as output a single d_{out} dimensional vector $y_n \in \mathbb{R}^{d_{out}}$:

$$y_n = \text{RNN}(x_{1:n})$$

Sequence Representation

$$x_i \in \mathbb{R}^{d_{in}} \quad y_n \in \mathbb{R}^{d_{out}}.$$

$$y_{1:n} = \text{RNN}^*(x_{1:n})$$
$$y_i = \text{RNN}(x_{1:i})$$

(Sub)Sequence Representation

$$x_i \in \mathbb{R}^{d_{in}} \quad y_i \in \mathbb{R}^{d_{out}}$$

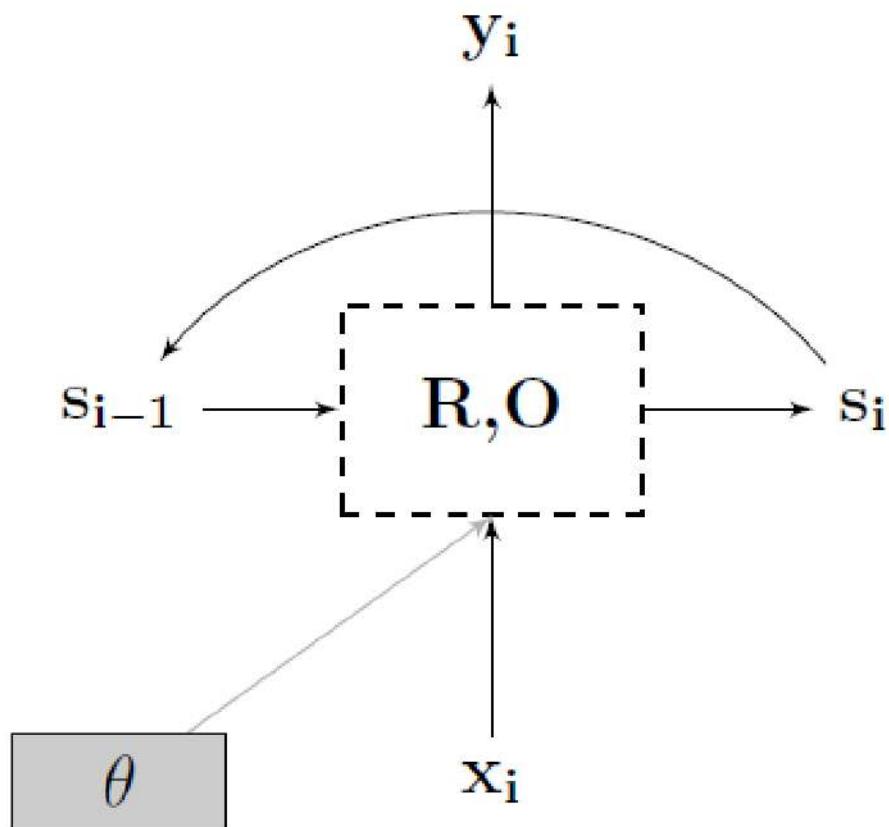
Methodology:
Embedding-Encoding-Prediction

$$w_{1:n} \rightarrow x_{1:n}$$

$$y_n = \text{RNN}(x_{1:n})$$

$$p(e = j | x_{1:n}) = \text{softmax}(\text{RNN}(x_{1:n}) \cdot W + b)_{[j]}$$

RNN Abstraction



$$\text{RNN}^*(x_{1:n}; s_0) = y_{1:n}$$

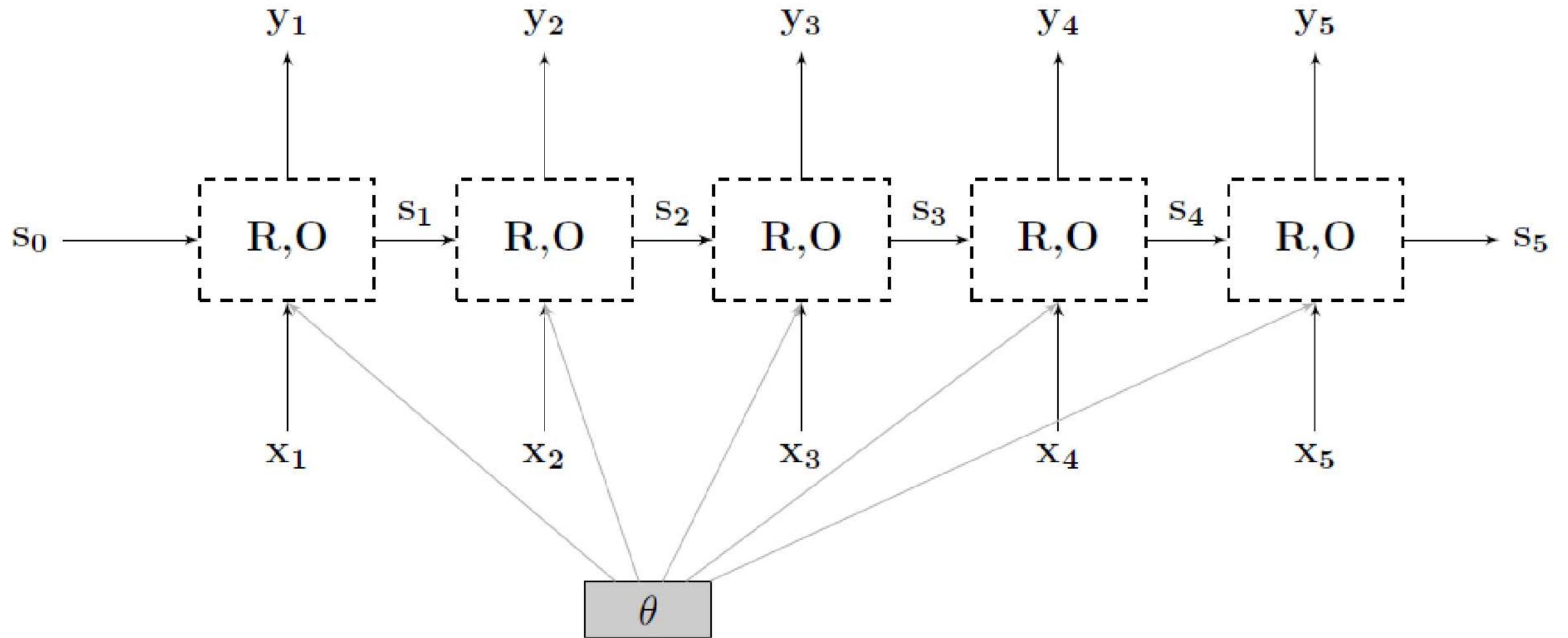
$$y_i = O(s_i)$$

$$s_i = R(s_{i-1}, x_i)$$

vectors $x_{1:n} = x_1, x_2, \dots, x_n$, ($x_i \in \mathbb{R}^{d_{in}}$)
vector $y_n \in \mathbb{R}^{d_{out}}$:

$$x_i \in \mathbb{R}^{d_{in}}, \quad y_i \in \mathbb{R}^{d_{out}}, \quad s_i \in \mathbb{R}^{f(d_{out})}$$

For the Simple RNN (Elman RNN) and the GRU architectures, O is the identity mapping, and for the LSTM architecture O selects a fixed subset of the state.



the same across the sequence positions, i.e., parameters are shared across all time steps

Expand the Recursion

$$s_4 = R(s_3, x_4)$$

$$\begin{aligned} &= R(\overbrace{R(s_2, x_3)}^{s_3}, x_4) \\ &= R(R(\overbrace{R(s_1, x_2)}^{s_2}, x_3), x_4) \\ &= R(R(R(\overbrace{R(s_0, x_1)}^{s_1}, x_2), x_3), x_4) \end{aligned}$$

RNN keeps track of the states of computation through the state vector s_i that is kept and being passed across invocations of R .

Long-distance dependency

那本書，我讀了e後，非常喜歡e。

Encoding as Representation

- s_n and y_n can be thought of as encoding the entire input sequence
- The output vector y_n is used for further prediction

$$\mathbf{y}_{1:n} = \text{RNN}^*(\mathbf{x}_{1:n})$$

$$\mathbf{y}_i = \text{RNN}(\mathbf{x}_{1:i}) \quad p(e = j | \mathbf{x}_{1:n}) = \text{softmax}(\text{RNN}(\mathbf{x}_{1:n}) \cdot \mathbf{W} + \mathbf{b})_{[j]}$$

$$\mathbf{x}_i \in \mathbb{R}^{d_{\text{in}}} \quad \mathbf{y}_i \in \mathbb{R}^{d_{\text{out}}}$$

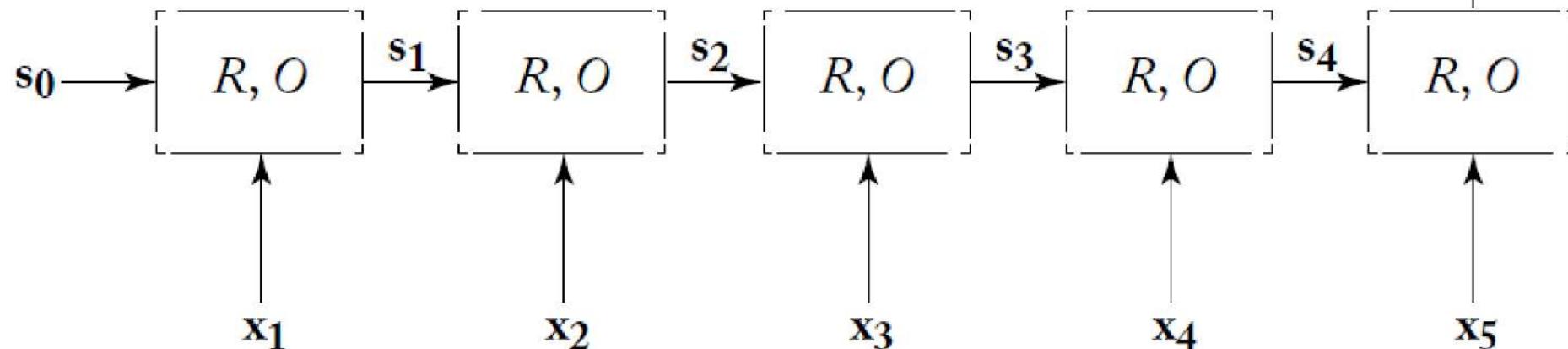
Condition on the entire history x_1, \dots, x_i without resorting to the Markov assumption

Acceptor RNN Training Graph

Embedding-Encoding-Prediction

$$w_{1:n} \rightarrow x_{1:n}$$
$$y_n = \text{RNN}(x_{1:n})$$
$$p(e = j|x_{1:n}) = \text{softmax}(\text{RNN}(x_{1:n}) \cdot W + b)_{[j]}$$

RNN: a trainable component in a larger network (e.g., sentiment analysis)



BackPropagation Through Time (BPTT):
Compute the gradients with respect to the loss

Common RNN-Usage: Acceptor

- Acceptor acceptor vs. transducer
 - Read in an input sequence, and produce a binary or multi-class answer
 - Observe the final state, and then decide on an outcome
 - Output vector $y_n = O(s_n)$ is fed into a fully connected layer or an MLP
- Examples
 - Read the characters of a word one by one and then use the final state to predict the part-of-speech of that word u•g•l•y → adj h•a•p•p•i•l•y → adv
 - Read in a sentence and, based on the final state decides if it conveys positive or negative sentiment 在夫子廟入口的地方，遍布我喜歡的小吃店。 (positive)
 - Read in a sequence of words and decides whether it is a valid noun-phrase 夫子廟入口的地方 → yes
在夫子廟入口的地方 → no

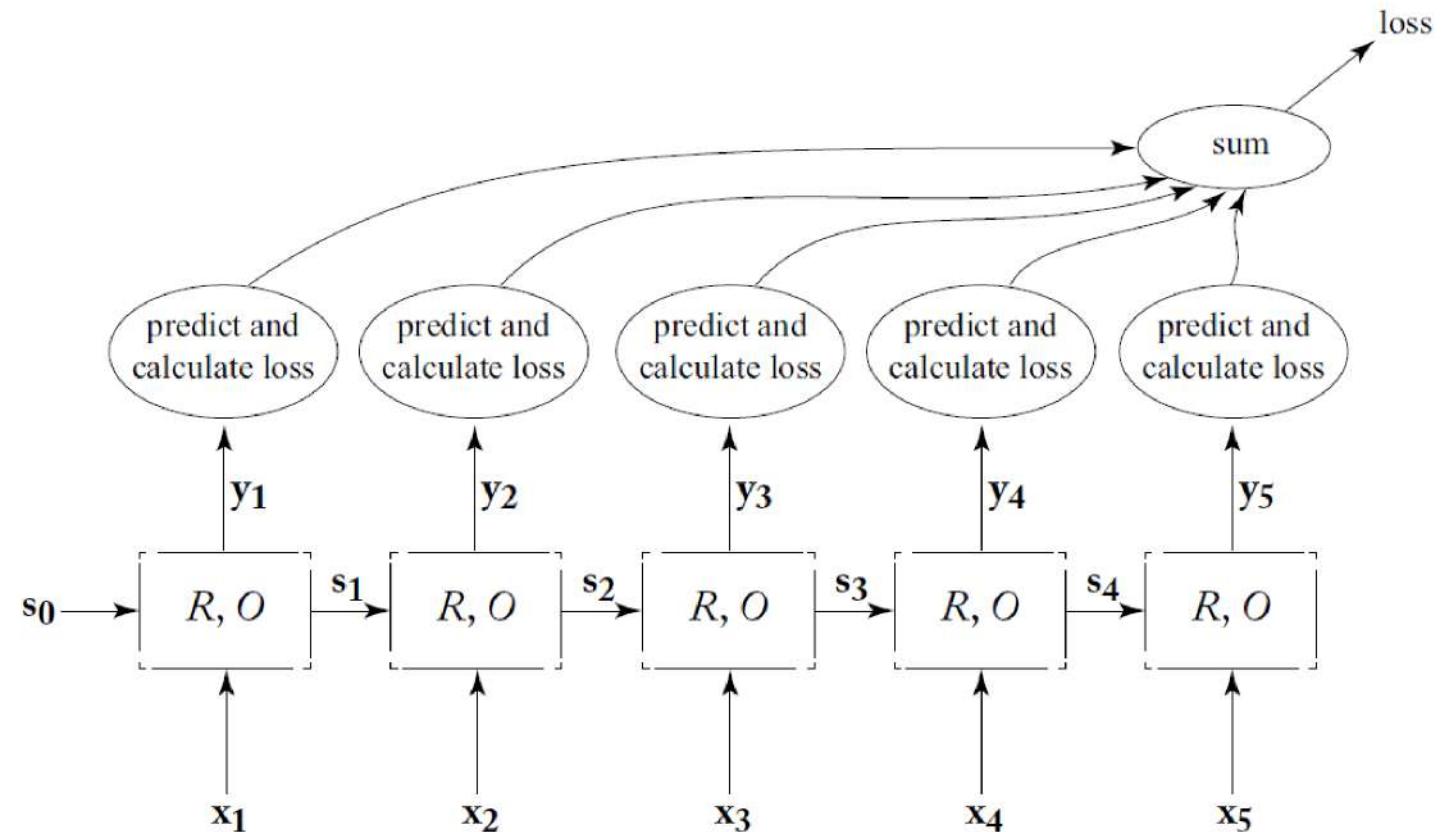
Common RNN-Usage: Encoder

- Acceptor
 - Use only the final output vector, y_n
 - Prediction is made solely on the basis of the final vector.
- Encoder
 - The final output vector, y_n
 - Treated as an encoding of the information in the sequence
 - Used as additional information together with other signals
 - Extractive document summarization
 - Run over the document with an RNN
 - Result in a vector y_n summarizing the entire document
 - Use y_n and other features to select the sentences to be included in the summarization

Common RNN-Usage: Transducer

- Produce an output \hat{t}_i for each input it reads in.
 - 在 夫子廟 入口 的 地方 → P21 Na Hcb DE Neb (POS tagging)
- Compute a local loss signal $L_{local}(\hat{t}_i, t_i)$ for each of the outputs \hat{t}_i based on a true label t_i

$$L(\hat{\mathbf{t}}_{1:n}, \mathbf{t}_{1:n}) = \sum_{i=1}^n L_{local}(\hat{\mathbf{t}}_i, \mathbf{t}_i)$$

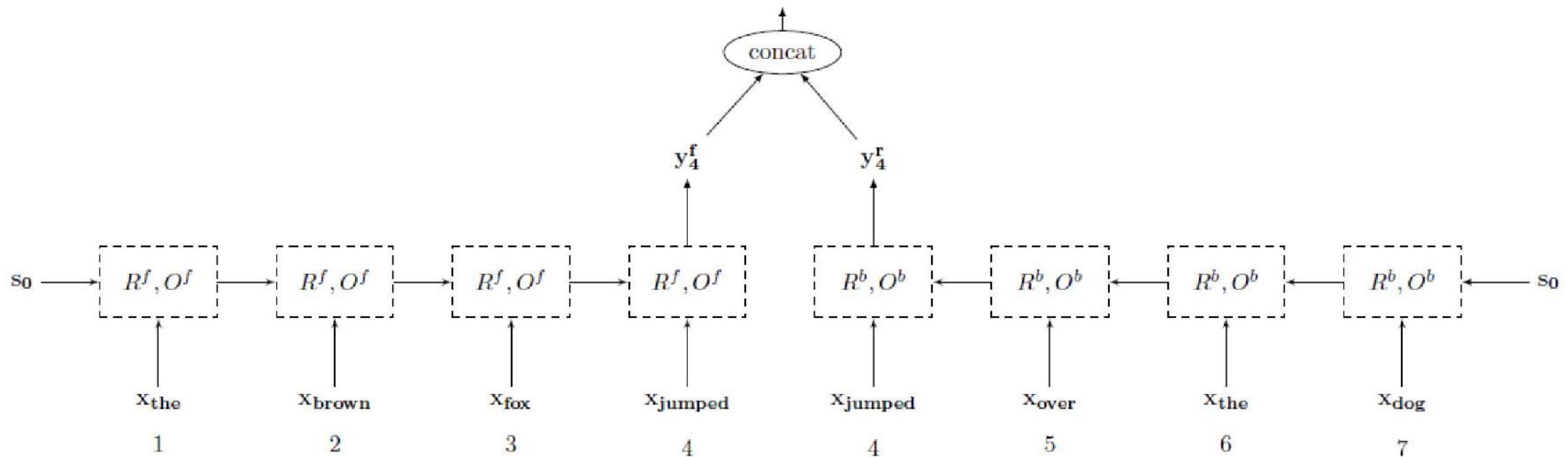


Language Modeling

- The sequence of words $\mathbf{x}_{1:i}$ is used to predict a distribution over the $(i + 1)$ th word
- Relax the Markov assumption: condition on the entire prediction history
- Special cases of the RNN transducer
 - The RNN generator
 - The related conditioned-generation (also called encoder-decoder) architecture
 - The conditioned-generation with attention architecture

bidirectional-RNN (biRNN)

- biRNN **relaxes the fixed window size assumption**, allowing to look arbitrarily far at both **the past** and **the future** within the sequence



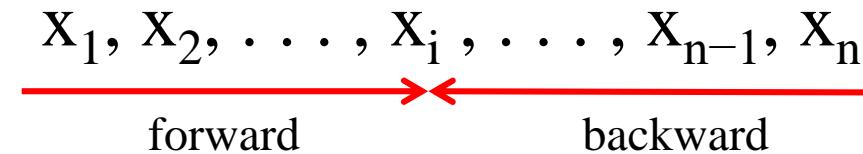
Computing the biRNN representation of the word “jumped” in the sentence “the brown fox jumped over the dog.”

biRNN

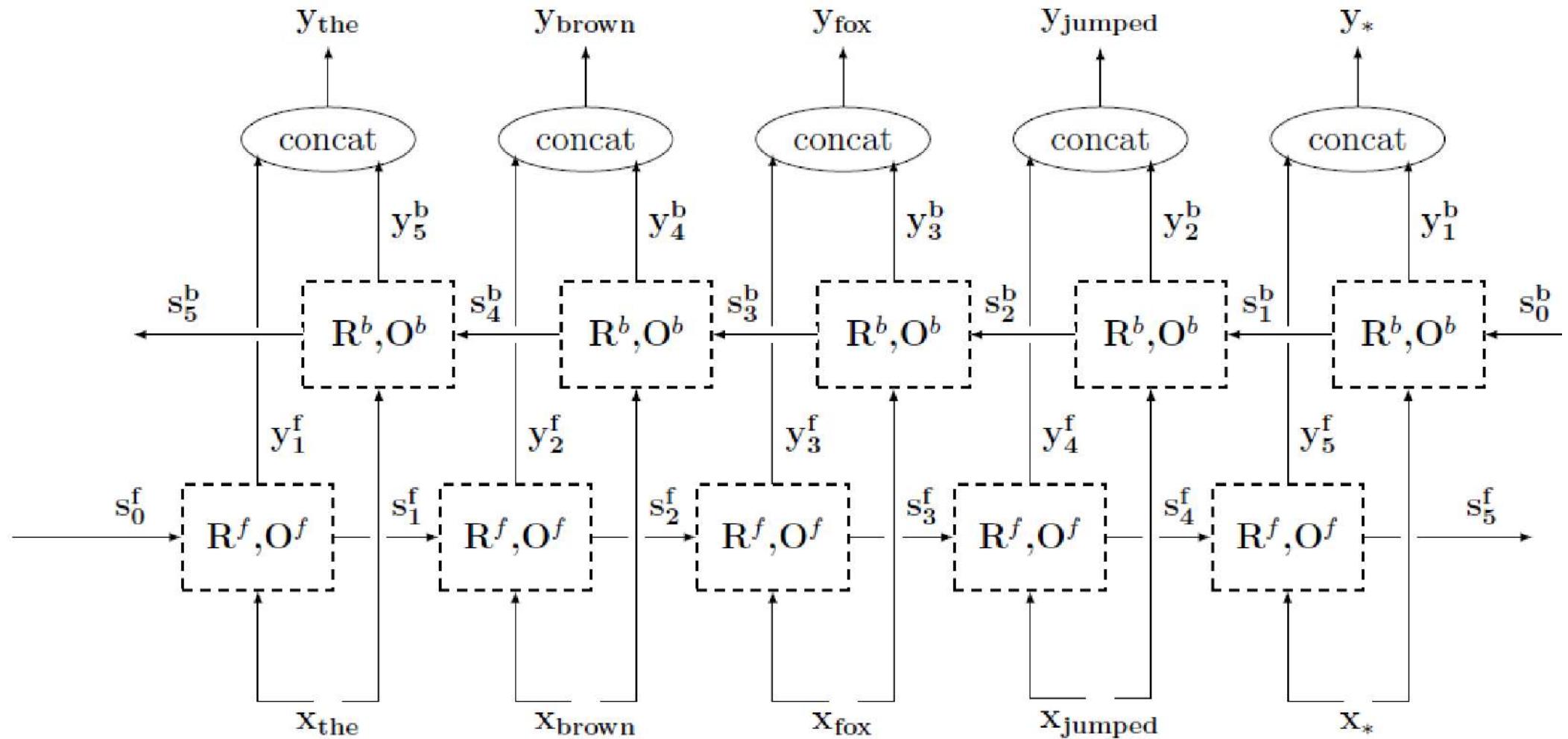
- Generate forward state s_i^f based on x_1, x_2, \dots, x_i
- Generate backward state s_i^b based on x_n, x_{n-1}, \dots, x_i
- RNN (R^f, O^f): feed the input sequence $\mathbf{x}_{1:n}$
- RNN (R^b, O^b): feed the input sequence $\mathbf{x}_{n:1}$
- Output at position i : $\mathbf{y}_i = [y_i^f; y_i^b] = [O^f(s_i^f); O^b(s_i^b)]$
 - The biRNN encoding of the i th word in a sequence: **the concatenation** of two RNNs

$$\text{biRNN}(\mathbf{x}_{1:n}, i) = \mathbf{y}_i = [\text{RNN}^f(\mathbf{x}_{1:i}); \text{RNN}^b(\mathbf{x}_{n:i})]$$

$x_1, x_2, \dots, x_i, \dots, x_{n-1}, x_n$



Computing the biRNN for the sentence “the brown fox jumped .”

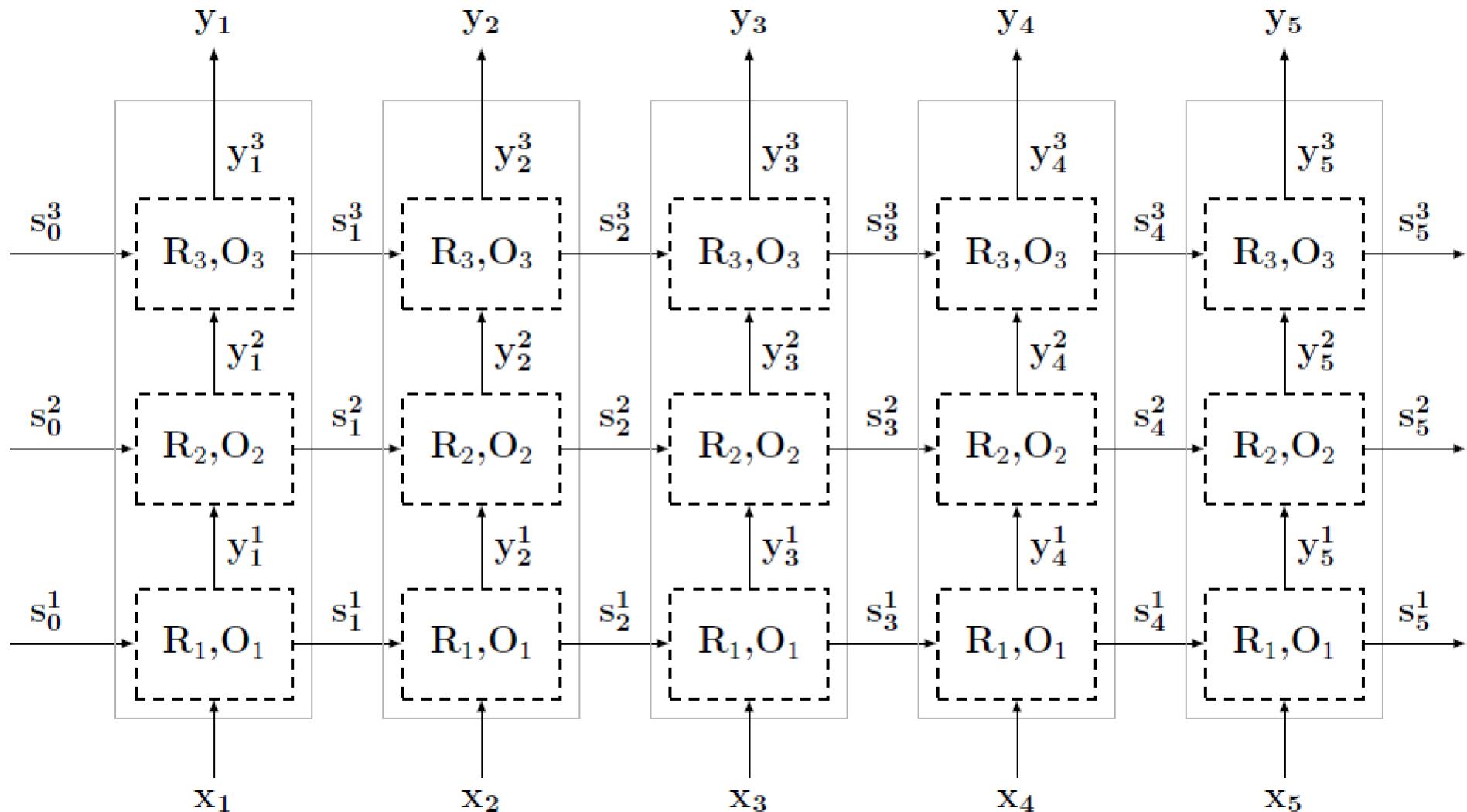


$$\text{biRNN}^*(\mathbf{x}_{1:n}) = \mathbf{y}_{1:n} = \text{biRNN}(\mathbf{x}_{1:n}, 1), \dots, \text{biRNN}(\mathbf{x}_{1:n}, n)$$

Multi-Layer (Stacked) RNNs: Deep RNNs

- k RNNs: $\text{RNN}_1, \dots, \text{RNN}_k$
- The j th RNN has states $s_{1:n}^j$ and outputs $y_{1:n}^j$
- The input for the first RNN are $\mathbf{x}_{1:n}$
- The input of the j th RNN ($j \geq 2$) are the outputs of the RNN below it, $y_{1:n}^{j-1}$.
- The output of the entire formation is the output of the last RNN, $y_{1:n}^k$
- Deep RNNs work better than shallower ones on some tasks empirically.

3-layer (deep) RNN



Concrete RNN Architectures

Specific Instantiations of RNN

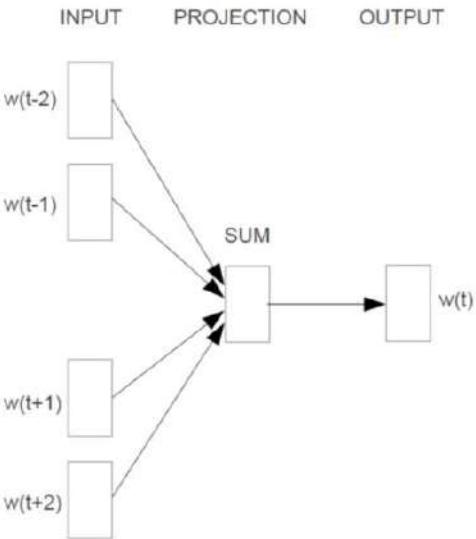
CBOW as an RNN

- Simple choice of R : the addition function

$$\mathbf{s}_i = R_{\text{CBOW}}(\mathbf{x}_i, \mathbf{s}_{i-1}) = \mathbf{s}_{i-1} + \mathbf{x}_i$$

$$\mathbf{y}_i = O_{\text{CBOW}}(\mathbf{s}_i) = \mathbf{s}_i$$

$$\mathbf{s}_i, \mathbf{y}_i \in \mathbb{R}^{d_s}, \quad \mathbf{x}_i \in \mathbb{R}^{d_s},$$



- The state resulting from inputs $\mathbf{x}_{1:n}$ is the sum of these inputs.
- This instantiation of the RNN ignores the sequential nature of the data.
 - continuous-bag-of-words model

Simple RNN (S-RNN, Elman Network)

$$\mathbf{s}_i = R_{\text{SRNN}}(\mathbf{x}_i, \mathbf{s}_{i-1}) = \underline{g(\mathbf{s}_{i-1} \mathbf{W}^s + \mathbf{x}_i \mathbf{W}^x + \mathbf{b})}$$

$$\mathbf{y}_i = O_{\text{SRNN}}(\mathbf{s}_i) = \mathbf{s}_i$$

Linear Transformation

Non-linear Transformation

$$\mathbf{s}_i, \mathbf{y}_i \in \mathbb{R}^{d_s}, \quad \mathbf{x}_i \in \mathbb{R}^{d_x}, \quad \mathbf{W}^x \in \mathbb{R}^{d_x \times d_s}, \quad \mathbf{W}^s \in \mathbb{R}^{d_s \times d_s}, \quad \mathbf{b} \in \mathbb{R}^{d_s}$$

or

$$\mathbf{s}_i = R_{\text{SRNN}}(\mathbf{x}_i, \mathbf{s}_{i-1}) = \underline{g([\mathbf{s}_{i-1}; \mathbf{x}_i] \mathbf{W} + \mathbf{b})}$$

$$\mathbf{y}_i = O_{\text{SRNN}}(\mathbf{s}_i) = \mathbf{s}_i$$

Linear Transformation

Non-linear Transformation

$$\mathbf{s}_i, \mathbf{y}_i \in \mathbb{R}^{d_s}, \quad \mathbf{x}_i \in \mathbb{R}^{d_x}, \quad \mathbf{W} \in \mathbb{R}^{(d_x + d_s) \times d_s}, \quad \mathbf{b} \in \mathbb{R}^{d_s}$$

RNN as a general purpose computing device

- The state s_i represents a finite memory
- Application of the function R
 - Reads in an input x_{i+1}
 - Reads in the current memory s_i
 - Operates on them in some way
 - Writes the result into memory, resulting in a new memory state s_{i+1}
- Problem: the memory access is not controlled
- How to provide more controlled memory access?
 - Soft gating rather than hard gating

Gating-based Architectures

- Consider a memory $s \in \mathbb{R}^d$, an input $x \in \mathbb{R}^d$, and a gate $g \in \{0,1\}^d$
- Using binary gate vector g to control access to memory s'

$$s' \leftarrow g \odot x + (1 - g) \odot (s)$$

- $g=1$: “reads” the entries in x and write them into memory
- $g=0$: copied from the memory s to the new memory s'

$$\begin{bmatrix} 8 \\ 11 \\ 3 \\ 7 \\ 5 \\ 15 \end{bmatrix} \leftarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \odot \begin{bmatrix} 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \odot \begin{bmatrix} 8 \\ 9 \\ 3 \\ 7 \\ 5 \\ 8 \end{bmatrix}$$

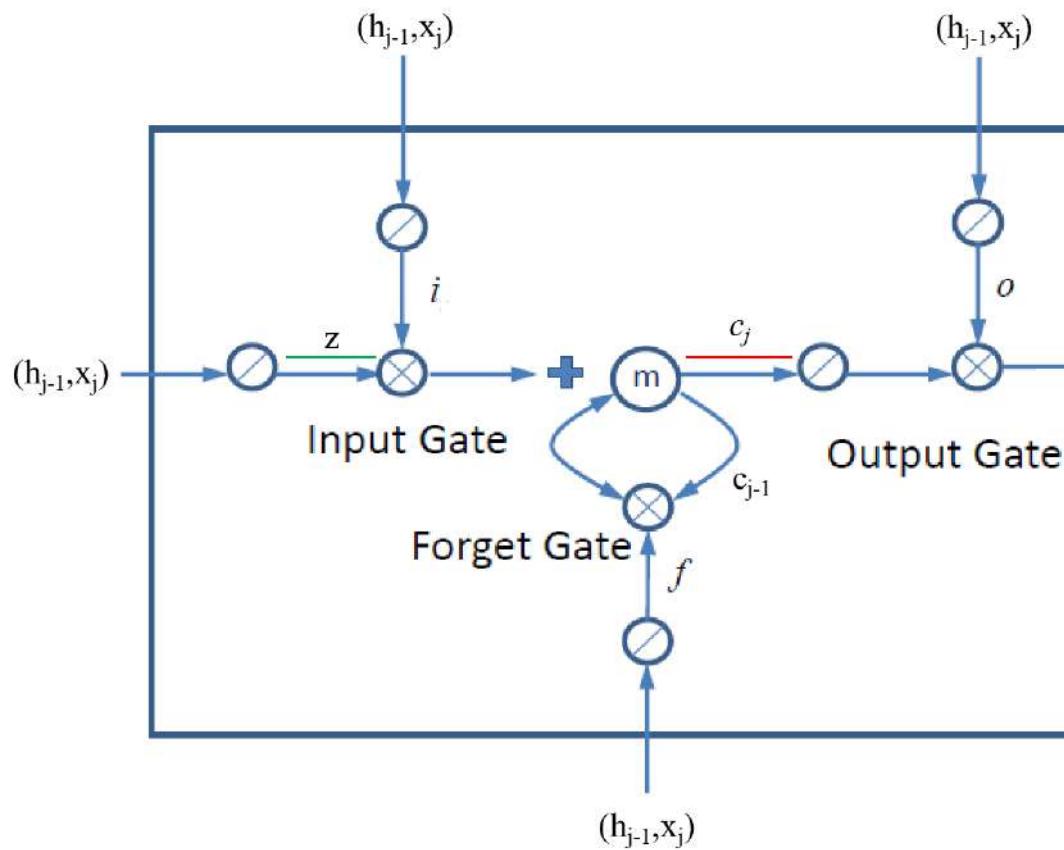
s' g x $(1 - g)$ s

Controllable Gating Mechanism

- Gates
 - Not be static
 - Controlled by the current memory state and the input
 - Behavior should be learned
- Differentiable gates
 - Replace the requirement $g \in \{0,1\}^n$
 - Allow arbitrary real numbers, $g' \in \mathbb{R}^n$
 - Pass through a sigmoid function $\sigma(g')$
- $\sigma(g')$ for $\sigma(g') \odot \mathbf{x}$
 - near-one values: the corresponding x are passed
 - near-zero values: the corresponding x are blocked
- Interpretation
 - Which parts of the inputs will be written to memory?
 - Which parts of memory will be overwritten (forgotten)?

Long Short-Term Memory (LSTM) Architecture

- Splits the state vector s_i into two halves
 - memory cells
 - working memory
- Controllable Gating
 - How much of the new input should be **written to the memory cell?**
 - How much of the current content of **the memory cell** should be **forgotten?**



- The state at time j , s_j , is composed of
 - vector c_j : memory component
 - vector h_j : hidden state component

$$s_j = R_{\text{LSTM}}(s_{j-1}, x_j) = [c_j; h_j]$$

$$c_j = f \odot c_{j-1} + i \odot z$$

$$h_j = o \odot \tanh(c_j)$$

- i : input gate
- f : forget gate
- o : output gate

$$i = \sigma(x_j W^{xi} + h_{j-1} W^{hi})$$

$$f = \sigma(x_j W^{xf} + h_{j-1} W^{hf})$$

$$o = \sigma(x_j W^{xo} + h_{j-1} W^{ho})$$

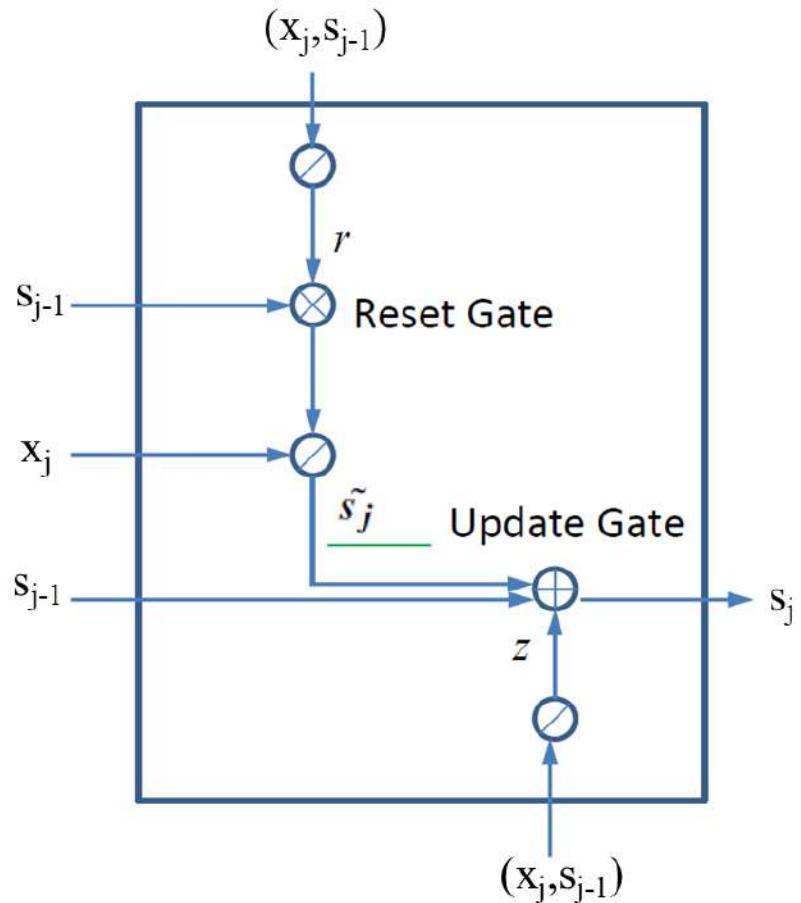
- Update candidate z : linear combinations of

- current input x_j
- previous state h_{j-1}
- passed through a tanh activation function.

- Output $y_j = O_{\text{LSTM}}(s_j) = h_j$

$$s_j \in \mathbb{R}^{2 \cdot d_h}, \quad x_i \in \mathbb{R}^{d_x}, \quad c_j, h_j, i, f, o, z \in \mathbb{R}^{d_h}, \quad W^{xo} \in \mathbb{R}^{d_x \times d_h}, \quad W^{ho} \in \mathbb{R}^{d_h \times d_h}$$

Gated Recurrent Unit (GRU)



- Reset gate (r)
 - control access to the previous state s_{j-1}
 - compute a proposed update \tilde{s}_j .
- Updated state s_j : interpolation of
 - the previous state s_{j-1}
 - the proposed state \tilde{s}_j .

$$s_j = R_{\text{GRU}}(s_{j-1}, x_j) = (1 - z) \odot s_{j-1} + z \odot \tilde{s}_j$$

$$z = \sigma(x_j W^{xz} + s_{j-1} W^{sz})$$

$$r = \sigma(x_j W^{xr} + s_{j-1} W^{sr})$$

$$\tilde{s}_j = \tanh(x_j W^{xs} + (r \odot s_{j-1}) W^{sg})$$

$$y_j = O_{\text{GRU}}(s_j) = s_j$$

$$s_j, \tilde{s}_j \in \mathbb{R}^{d_s}, \quad x_i \in \mathbb{R}^{d_x}, \quad z, r \in \mathbb{R}^{d_s}, \quad W^{xo} \in \mathbb{R}^{d_x \times d_s}, \quad W^{so} \in \mathbb{R}^{d_s \times d_s}$$

Modeling with Recurrent Networks

Acceptors

- Read in an input sequence, and produce a binary or multi-class answer at the end
- Sentence-Level Sentiment Classification
 - Given a sentence (e.g., part of a review),
 - Assign it one of two values: Positive or Negative
- Document Level Sentiment Classification

Sentence-Level Sentiment Classification

- Model using an RNN-acceptor
 - Tokenization
 - Reads in the words of the sentence one at a time
 - Fed the final state of RNN into an MLP followed by a softmax-layer with two outputs
 - k-way classification

$$p(\text{label} = k \mid w_{1:n}) = \hat{\mathbf{y}}_{[k]}$$
$$\hat{\mathbf{y}} = \text{softmax}(\text{MLP}(\text{RNN}(\mathbf{x}_{1:n})))$$
$$\mathbf{x}_{1:n} = \mathbf{E}_{[w_1]}, \dots, \mathbf{E}_{[w_n]}$$

two RNNs: read in from
left-to-right and right-to-left

$$p(\text{label} = k \mid w_{1:n}) = \hat{\mathbf{y}}_{[k]}$$
$$\hat{\mathbf{y}} = \text{softmax}(\text{MLP}([\text{RNN}^f(\mathbf{x}_{1:n}); \text{RNN}^b(\mathbf{x}_{n:1})]))$$
$$\mathbf{x}_{1:n} = \mathbf{E}_{[w_1]}, \dots, \mathbf{E}_{[w_n]}$$

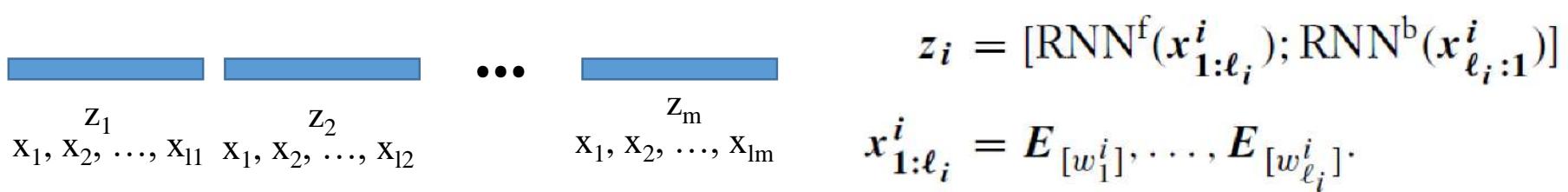
Longer Sentences

- Hierarchical Architecture
 - The sentence is split into smaller spans based on punctuation
 - Each span is fed into a forward and a backward RNN
 - Sequence of resulting vectors (one for each span) are then fed into an RNN acceptor

given a sentence $w_{1:n}$ which is split into m spans, $w_{1:\ell_1}^1, \dots, w_{1:\ell_m}^m$, the architecture is given by:

$$p(\text{label} = k \mid w_{1:n}) = \hat{\mathbf{y}}_{[k]}$$

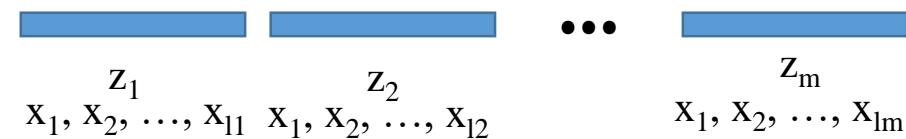
$$\hat{\mathbf{y}} = \text{softmax}(\text{MLP}(\text{RNN}(\mathbf{z}_{1:m})))$$



Document Level Sentiment Classification

- Differences

- Consist of multiple sentences
- Result given at the end



- Hierarchical Structure Approach

- Each sentence s_i is encoded using a gated RNN producing a vector z_i
- The vectors $z_{1:n}$ are then fed into a second gated RNN, producing a vector $h = RNN(z_{1:n})$
- Use the result h for prediction: $\hat{y} = softmax(MLP(h))$

- Alternative

all the intermediate vectors

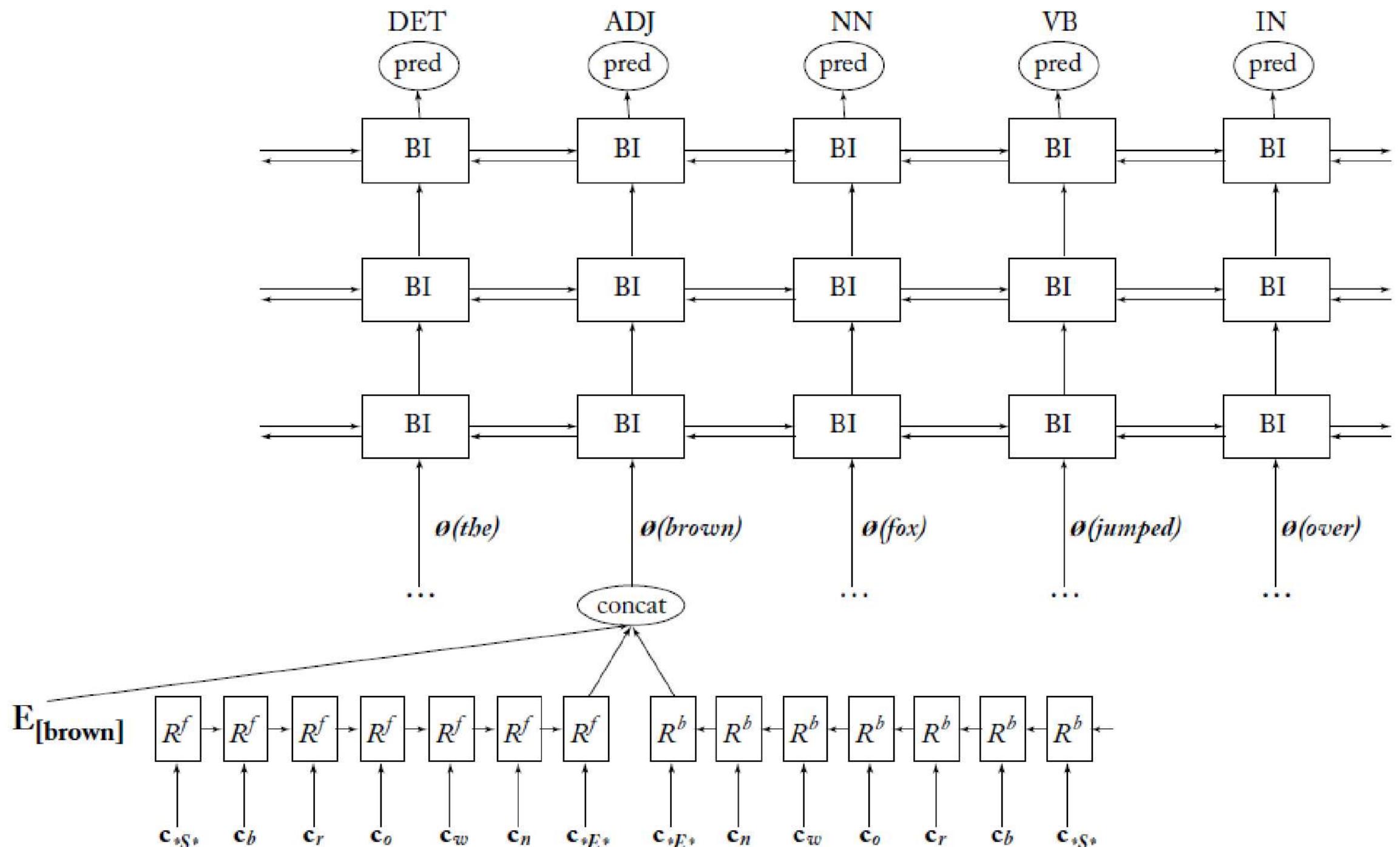
$$MLP \quad (h_{1:n} = RNN^*(z_{1:n}), \hat{y} = softmax(MLP(\frac{1}{n} \sum_{i=1}^n h_i))).$$

RNNs as Feature Extractors

- deep biRNN skeleton for part-of-speech tagging
 - Given a sentence with words $s=w_{1:n}$, translate them into input vectors $x_{1:n}$ using a feature function $x_i=\phi(s,i)$
 - Fed the input vectors into a deep biRNN, resulting in $y_{1:n}=\text{biRNN}^*(x_{1:n})$
 - Fed y_i into an MLP to predict one of k possible output tag
- From words to inputs with character-level RNNs
 - Map a word w_i to an input vector x_i with an embedding matrix
 - Use two character-level RNNs to deal with OOV problem

$$p(t_i = j | w_1, \dots, w_n) = \text{softmax}(\text{MLP}(\text{biRNN}(x_{1:n}, i)))_{[j]}$$

$$x_i = \phi(s, i) = [E_{[w_i]}; \text{RNN}^f(c_{1:\ell}); \text{RNN}^b(c_{\ell:1})]$$



Conditioned Generation (Encoder-Decoder)

Using Language Model for Generation

Shannon's Method

- Predict a probability distribution over the first word conditioned on the start symbol, and draw a random word according to the predicted distribution.
- Then, predict a probability distribution over the second word conditioned on the first, and so on, until predicting the end-of-sequence $\langle /s \rangle$ symbol.
- Options
 - Choose the highest scoring prediction (word) at each step
 - Sample a random word according to the predicted distribution
 - Use beam search in order to find a sequence with a globally-high probability

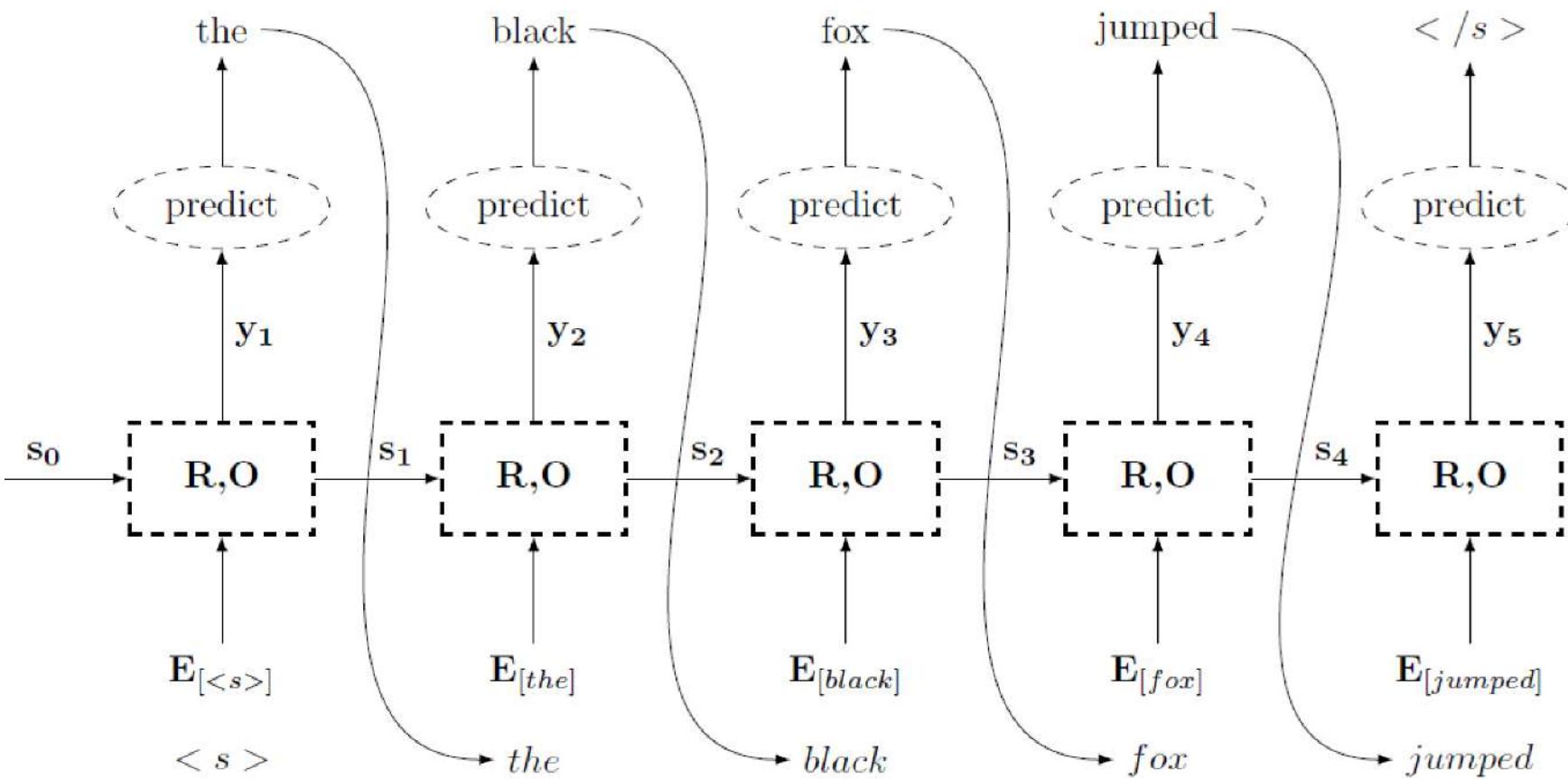
RNN Generators: Sequence Generation

- sequence generation

After predicting a distribution over the next output symbols

$$p(t_i = k | t_{1:i-1})$$

a token t is chosen and its corresponding embedding vector is fed as the input to the next step



This is modeled in the RNN framework as:

$$p(t_{j+1} = k \mid \hat{t}_{1:j}) = f(\text{RNN}(\hat{\mathbf{t}}_{1:\mathbf{j}}))$$
$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1})$$

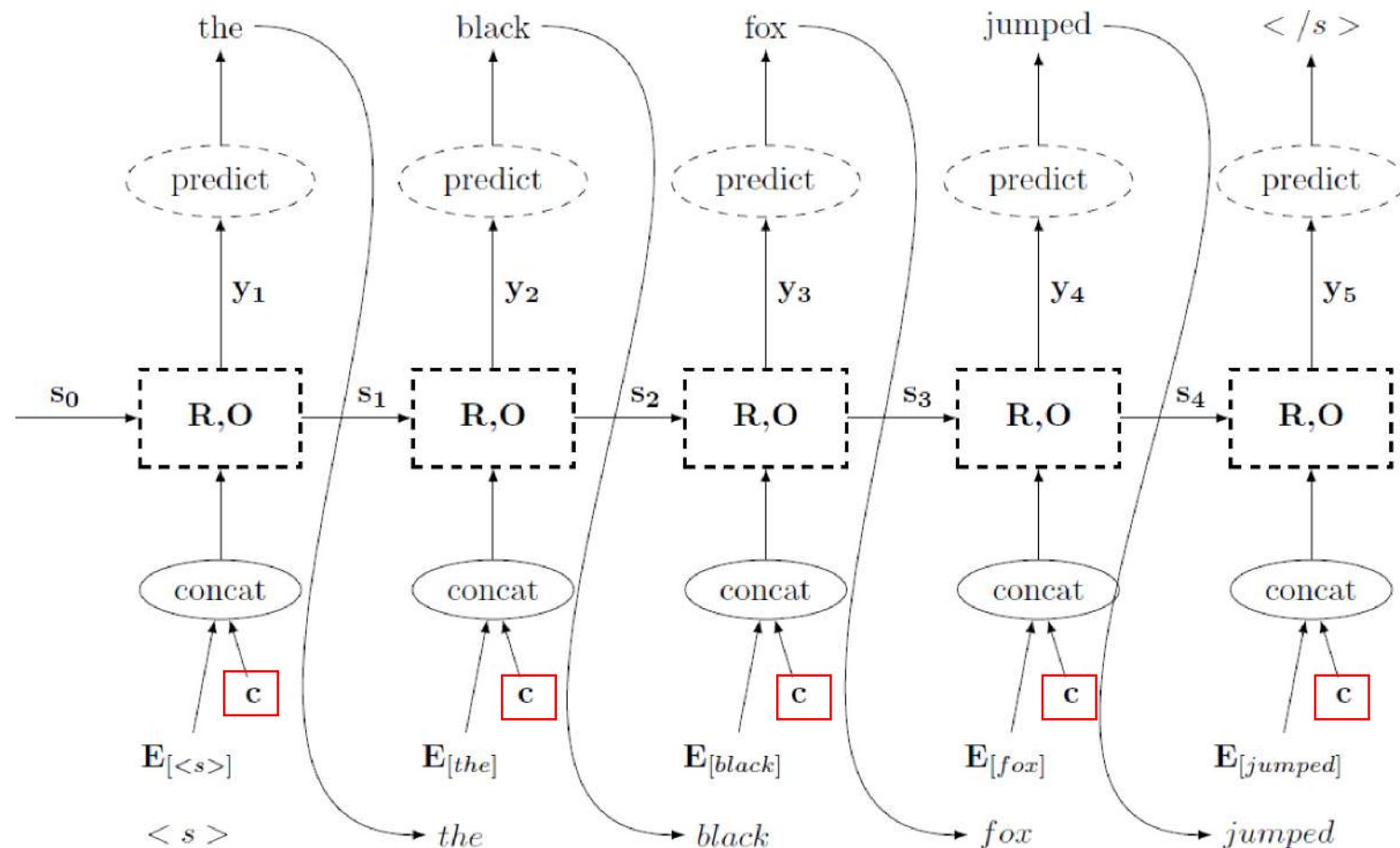
or, if using the more detailed recursive definition:

$$p(t_{j+1} = k \mid \hat{t}_{1:j}) = f(O(\mathbf{s}_{\mathbf{j}+1}))$$
$$\mathbf{s}_{\mathbf{j}+1} = R(\hat{\mathbf{t}}_{\mathbf{j}}, \mathbf{s}_j)$$
$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1})$$

where f is a parameterized function that maps the RNN state to a distribution over words, for example $f(\mathbf{x}) = \text{softmax}(\mathbf{x}\mathbf{W} + \mathbf{b})$ or $f(\mathbf{x}) = \text{softmax}(\text{MLP}(\mathbf{x}))$.

Conditioned Generation (Encoder-Decoder)

Generate the next token t_{j+1} based on the previously generated tokens $\hat{t}_{1:j}$ $\hat{t}_{j+1} \sim p(t_{j+1} = k \mid \hat{t}_{1:j})$



The Conditioned Generation Framework

- The next token is generated based on the previously generated tokens, and an additional conditioning context c .

$$\begin{aligned}\hat{t}_{j+1} &\sim p(t_{j+1} = k \mid \hat{t}_{1:j}, c) \\ p(t_{j+1} = k \mid \hat{t}_{1:j}, c) &= f(\text{RNN}(\mathbf{v}_{1:j})) \\ \mathbf{v}_i &= [\hat{\mathbf{t}}_i; \mathbf{c}] \\ \hat{t}_j &\sim p(t_j \mid \hat{t}_{1:j-1}, c)\end{aligned}$$

using the recursive definition:

$$\begin{aligned}p(t_{j+1} = k \mid \hat{t}_{1:j}, c) &= f(O(\mathbf{s}_{j+1})) \\ \mathbf{s}_{j+1} &= R(\mathbf{s}_j, [\hat{\mathbf{t}}_j; \mathbf{c}]) \\ \hat{t}_j &\sim p(t_i \mid \hat{t}_{1:j-1}, c)\end{aligned}$$

The Context c

- Topic as a conditioning context
 - a large corpus of items categorized into different topics
- Movie Review
 - Condition the generation on the genre of the movie
 - Condition the rating of the review
 - Condition the geographic region of the author
- Condition on *inferred* properties
 - Automatically extract from the text
- Form
 - fixed-length, set-like examples of conditioning contexts
 - takes c to be itself a sequence → sequence to sequence conditioned generation framework

Sequence to Sequence Conditioned Generation Framework

- Encoder-decoder framework
 - Given a source sequence $x_{1:n}$, generate a target output sequence $t_{1:m}$
 - Encode the source sentence $x_{1:n}$ into a vector using an encoder function $c = \text{Enc}(x_{1:n})$ commonly an RNN:
$$\mathbf{c} = \text{RNN}^{\text{enc}}(\mathbf{x}_{1:n})$$
 - A conditioned generator RNN (decoder) is then used to generate the desired output $t_{1:m}$

$$p(t_{j+1} = k \mid \hat{t}_{1:j}, c) = f(\text{RNN}(\mathbf{v}_{1:j}))$$

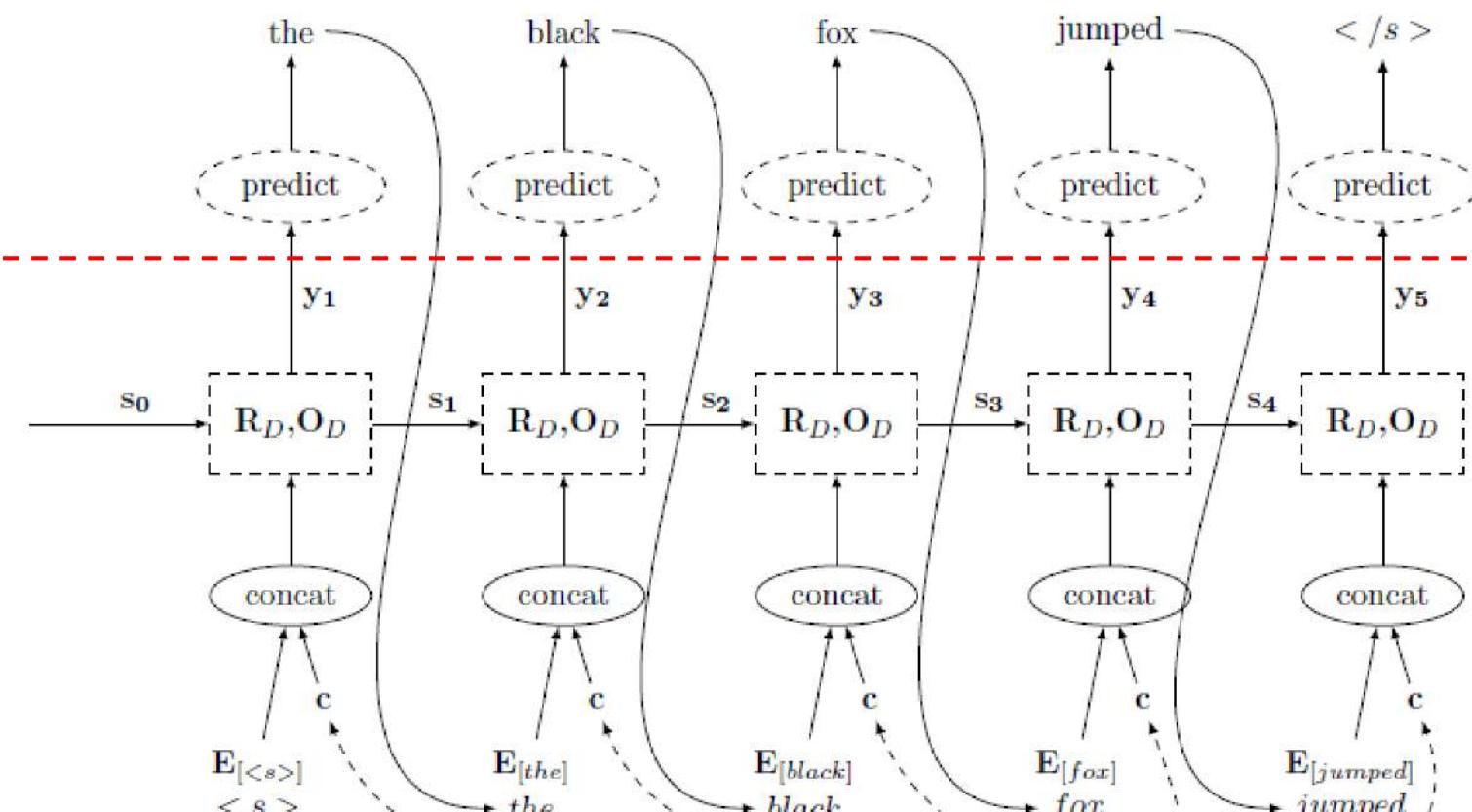
Decoder

Language Model

$$\mathbf{v}_i = [\hat{t}_i; c]$$

$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1}, c)$$

Train encoder and decoder RNNs jointly



Encoding the source sentence $\mathbf{x}_{1:n}$ into a vector using an encoder function

$$c = \text{Enc}(\mathbf{x}_{1:n})$$

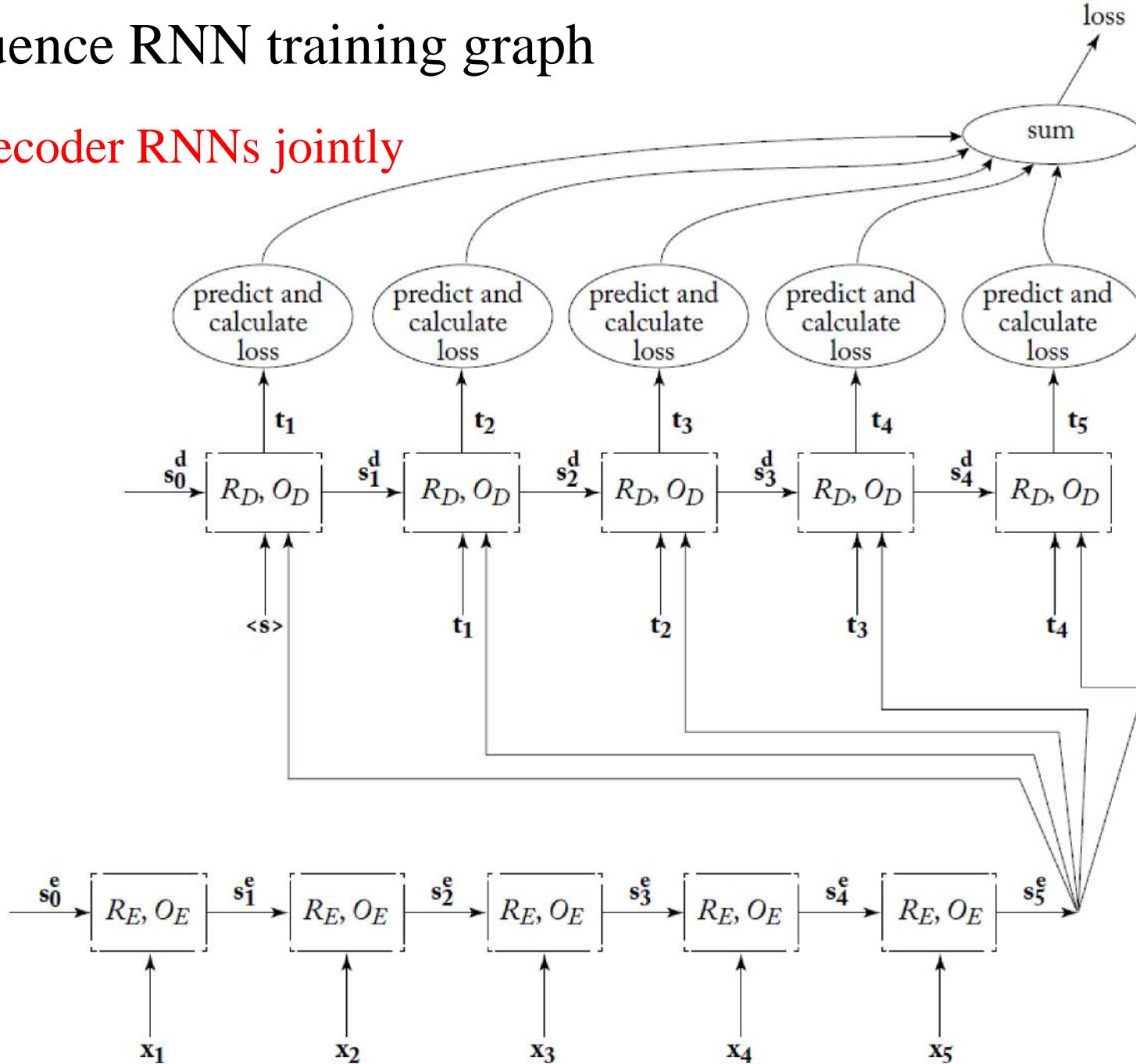
e.g.,

$$\text{RNN: } c = \text{RNN}^{\text{enc}}(\mathbf{x}_{1:n})$$

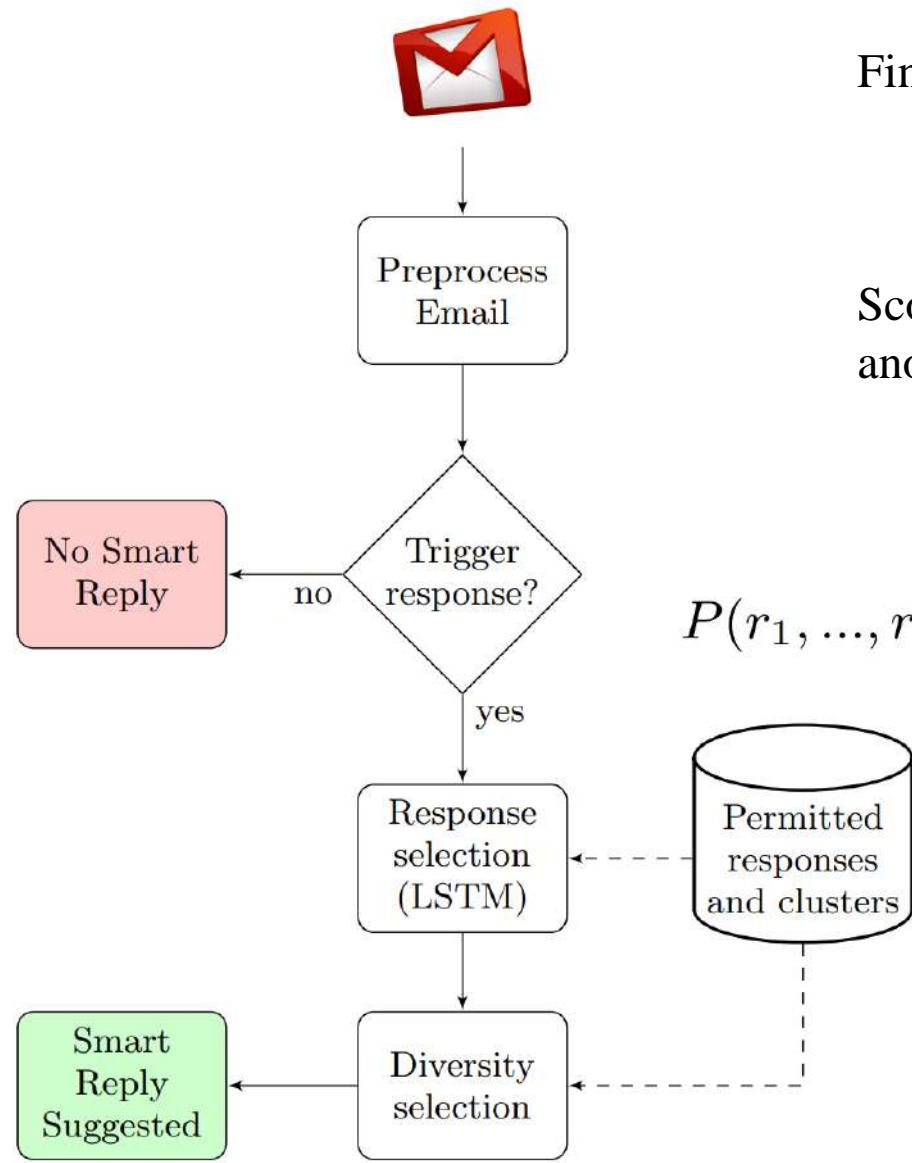
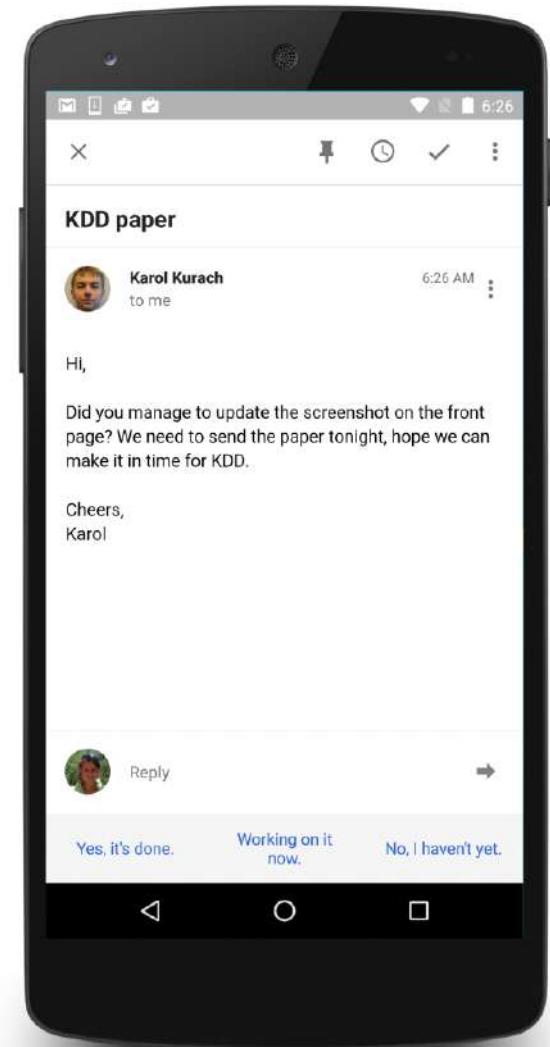
Encoder

Sequence-to-sequence RNN training graph

Train encoder and decoder RNNs jointly



Application: Email Auto-response



Find the most likely response given an original message.

$$\mathbf{r}^* = \operatorname{argmax}_{r \in R} P(\mathbf{r} | \mathbf{o})$$

Score one sequence of tokens \mathbf{r} , conditional on another sequence of tokens \mathbf{o} .

$$P(r_1, \dots, r_m | o_1, \dots, o_n)$$

$$P(r_1, \dots, r_m | o_1, \dots, o_n) = \prod_{i=1}^m P(r_i | o_1, \dots, o_n, r_1, \dots, r_{i-1})$$

Conditioning Contexts

- Text-based setting
 - RNN
 - A single word
 - A CBOW encoding
 - Generated by a convolutional network
 - Based on some other complex computation
- Dialogue-based setting
 - Trainable embedding vector associated with the user who wrote the response
 - Age, gender, social role, background knowledge, personality traits and many other latent factors
 - User Embedding
- Image captioning
 - An input image is encoded as a vector
 - A conditioning context for an RNN generator trained to predict image descriptions

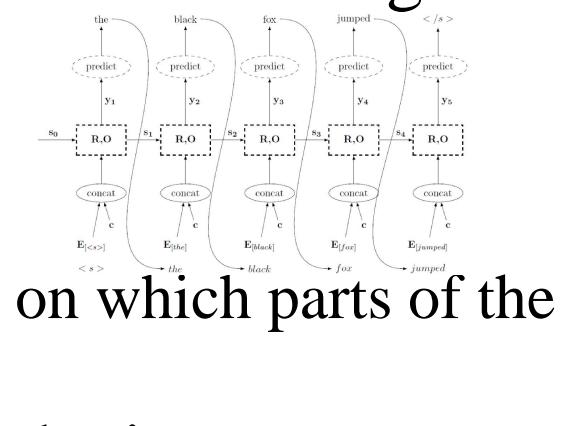
Conditioned Generation with Attention

- Relax the condition that the entire source sentence be encoded as a single vector
- The Encoder-Decoder-Attention Model
 - The input sentence is encoded as a sequence of vectors
 - The decoder uses a **soft attention mechanism** in order to decide on which parts of the encoding input it should focus.
- Encode a length n input sequence $x_{1:n}$ using a biRNN, producing n vectors $c_{1:n}$

$$c_{1:n} = \text{Enc}(x_{1:n}) = \text{biRNN}^*(x_{1:n})$$

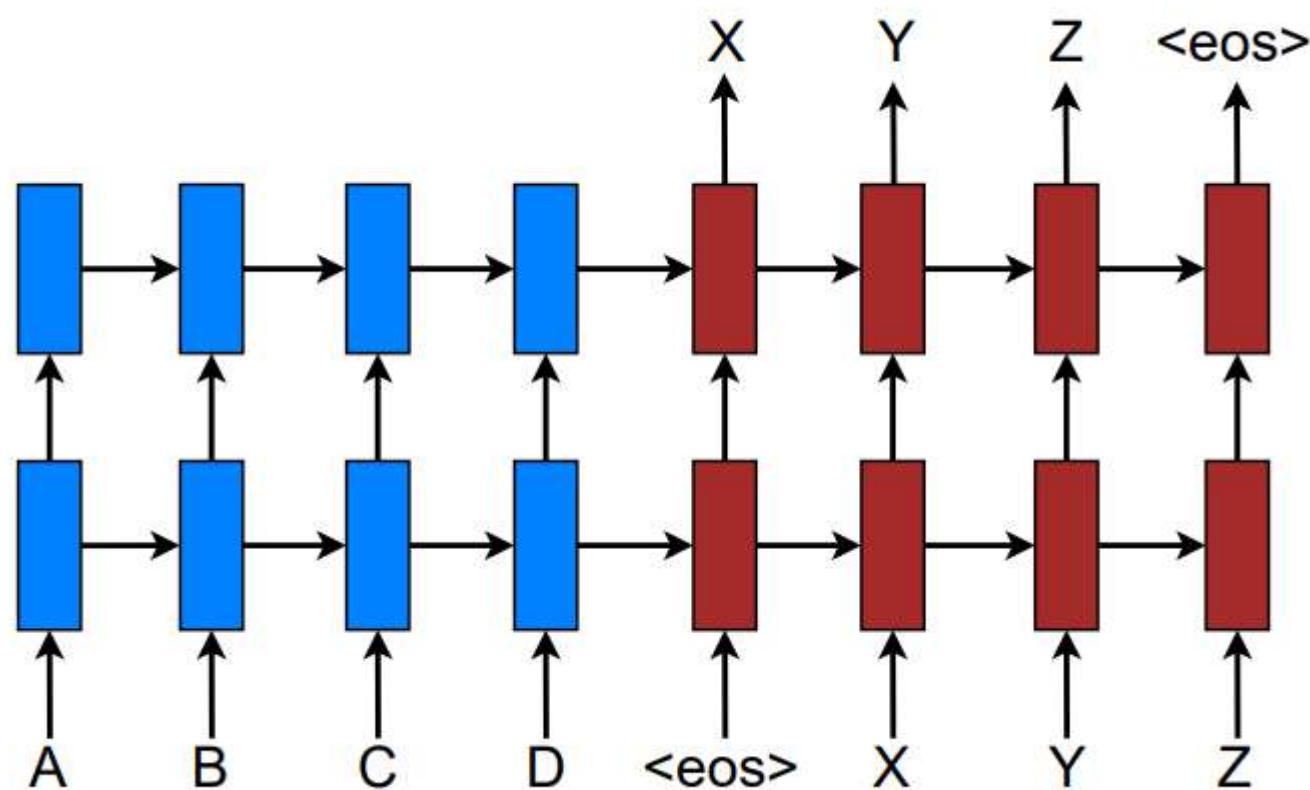
- The decoder chooses which of the vectors $c_{1:n}$ it should attend to, resulting in a focused context vector

$$c^j = \text{attend}(c_{1:n}, \hat{t}_{1:j}).$$

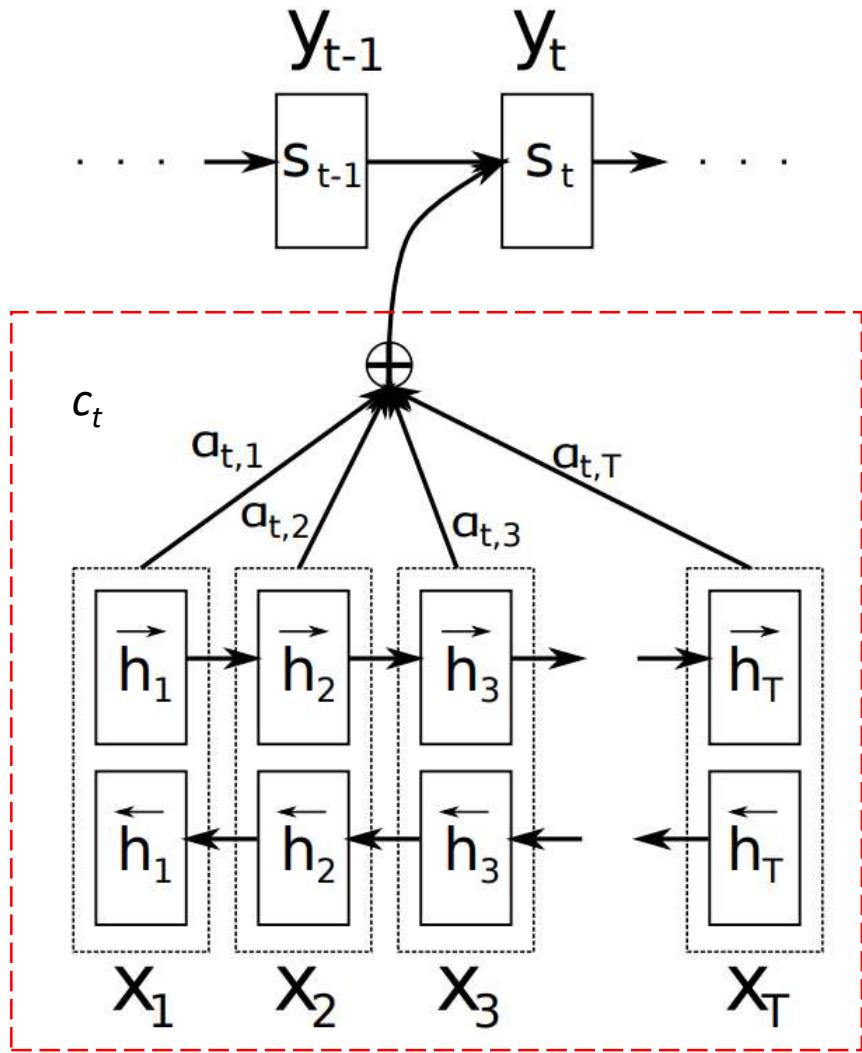


Neural Machine Translation

A stacking recurrent architecture for translating a source sequence A B C D into a target sequence X Y Z.



Attention-based Models



$$p(y_i | y_1, \dots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i),$$

where s_i is an RNN hidden state for time i , computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

where

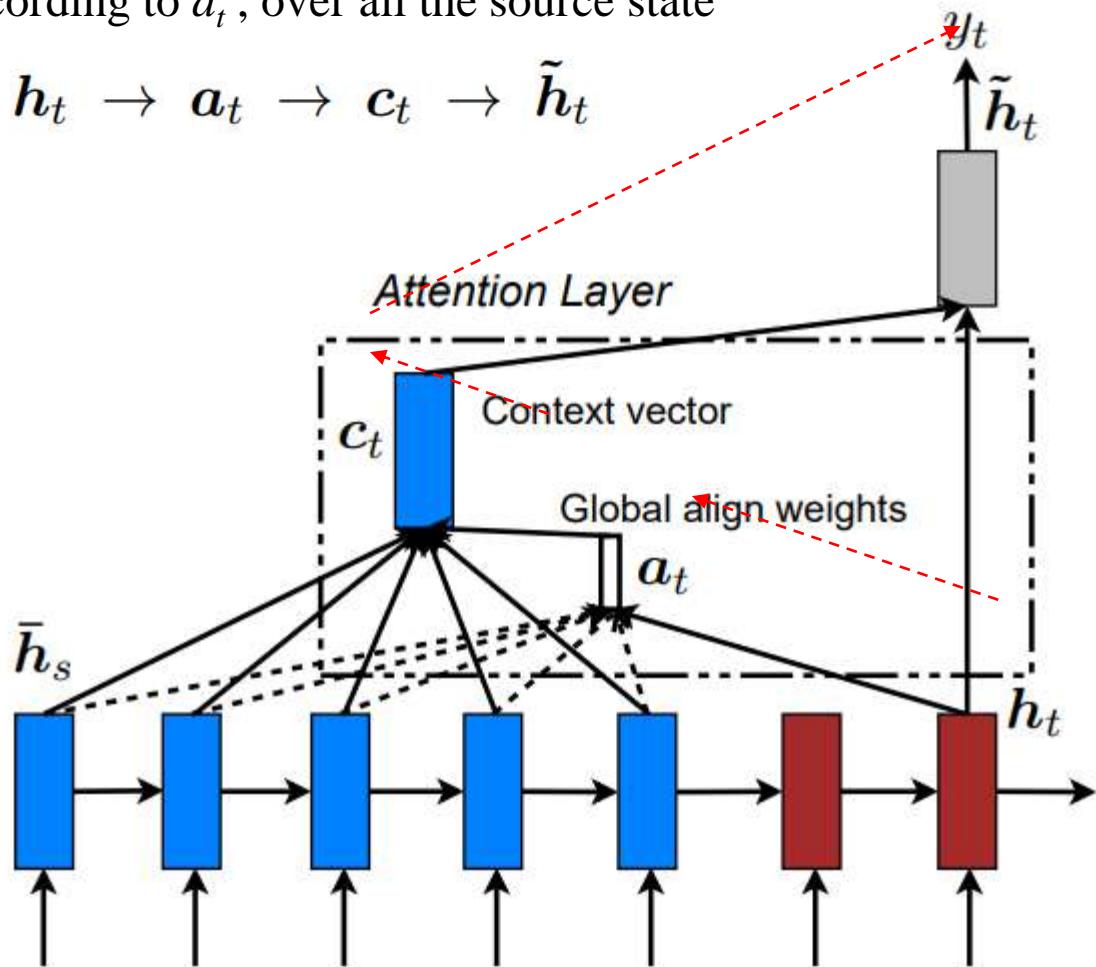
$$e_{ij} = a(s_{i-1}, h_j)$$

Global Attentional Model

Attend to all words on the source side for each target word

Vector c_t is computed as the weighted average, according to a_t , over all the source state

$$\mathbf{h}_t \rightarrow \mathbf{a}_t \rightarrow \mathbf{c}_t \rightarrow \tilde{\mathbf{h}}_t$$



$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t]) \quad (5)$$

The attentional vector $\tilde{\mathbf{h}}_t$ is then fed through the softmax layer to produce the predictive distribution formulated as:

$$p(y_t | y_{<t}, x) = \text{softmax}(\mathbf{W}_s \tilde{\mathbf{h}}_t) \quad (6)$$

$$\begin{aligned} \mathbf{a}_t(s) &= \text{align}(\mathbf{h}_t, \bar{\mathbf{h}}_s) \\ &= \frac{\exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s))}{\sum_{s'} \exp(\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_{s'}))} \end{aligned} \quad (7)$$

Content-based Function

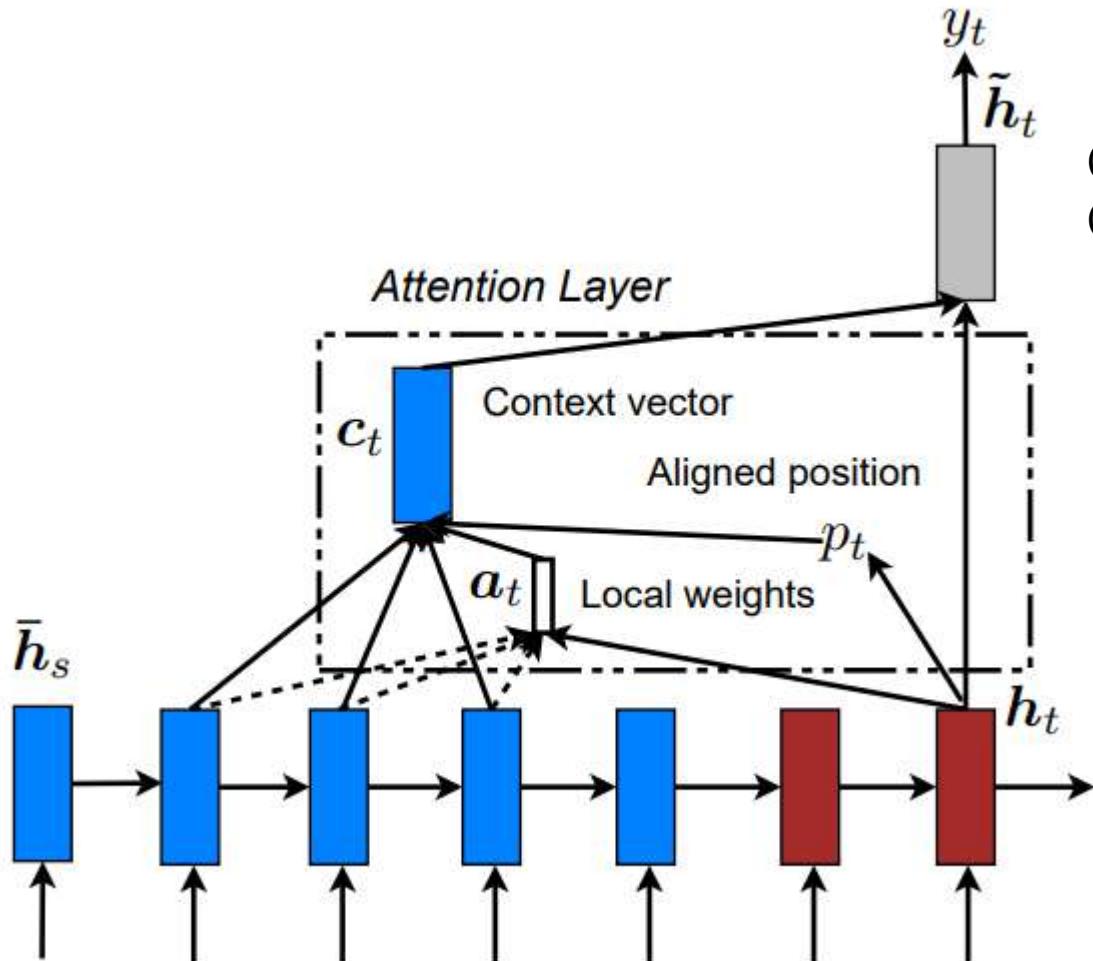
$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \begin{cases} \mathbf{h}_t^\top \bar{\mathbf{h}}_s & \text{dot} \\ \mathbf{h}_t^\top \mathbf{W}_a \bar{\mathbf{h}}_s & \text{general} \\ \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{h}_t; \bar{\mathbf{h}}_s]) & \text{concat} \end{cases}$$

Location-based Function

$$\mathbf{a}_t = \text{softmax}(\mathbf{W}_a \mathbf{h}_t) \quad \text{location}$$

Local Attention Model

Choose to focus only on a small subset of the source positions per target word.



- (1) Generates an aligned position p_t for each target word at time t.
- (2) Derive c_t as a weighted average over the set of source hidden states within the window $[p_t-D, p_t+D]$; D is empirically selected.

Monotonic alignment (local- m)

$$p_t = t$$

Predictive alignment (local- p)

$$p_t = S \cdot \text{sigmoid}(\mathbf{v}_p^\top \tanh(\mathbf{W}_p \mathbf{h}_t)),$$

where S is the source sentence length

Focused context vector

$$c^j = \text{attend}(c_{1:n}, \hat{t}_{1:j})$$

$$p(t_{j+1} = k \mid \hat{t}_{1:j}, x_{1:n}) = f(O(s_{j+1}))$$

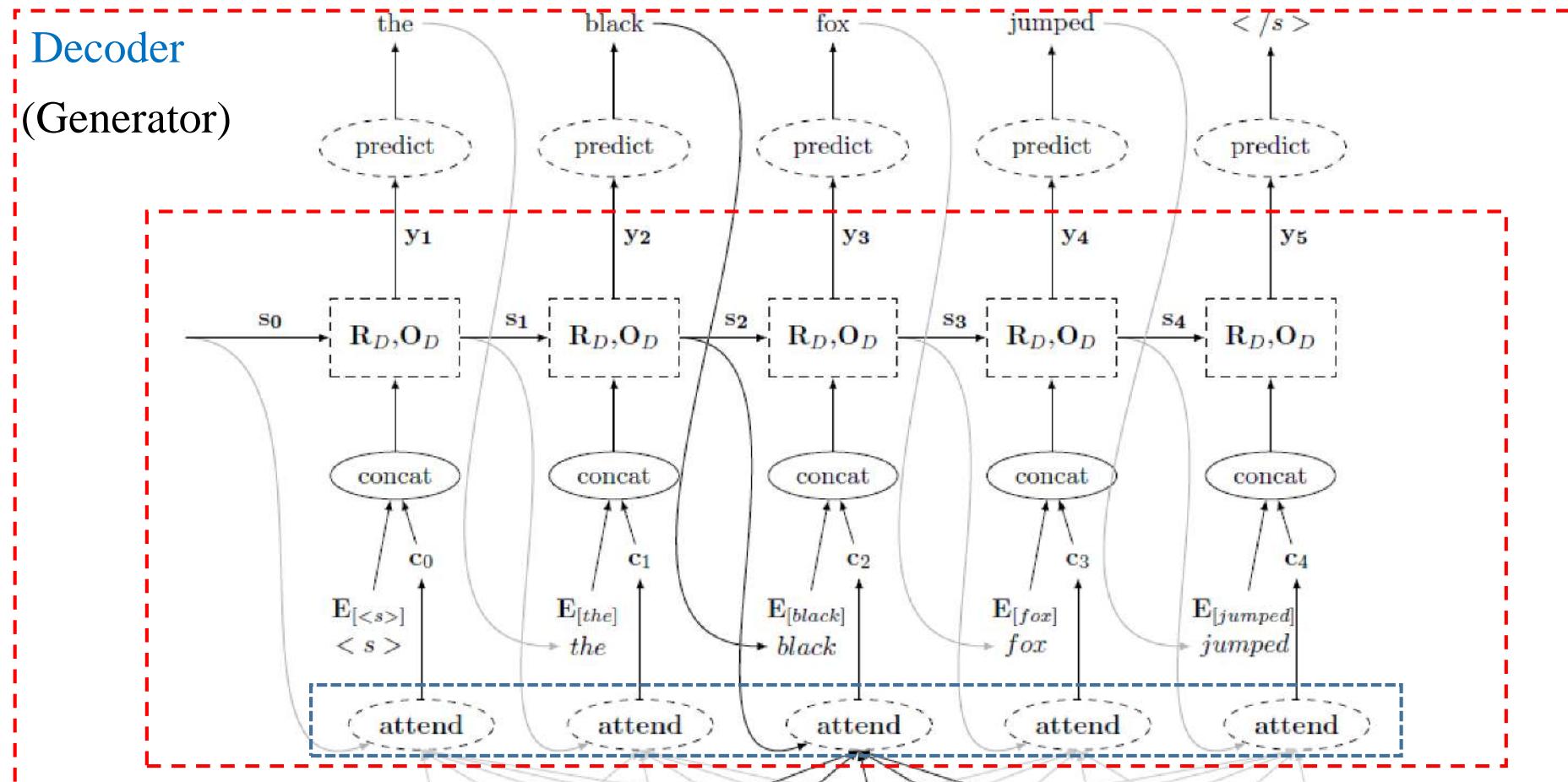
$$s_{j+1} = R(s_j, [\hat{t}_j; c^j])$$

$$c^j = \text{attend}(c_{1:n}, \hat{t}_{1:j})$$

$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1}, x_{1:n})$$

Encoding the source sentence $x_{1:n}$ into vectors $c_{1:n}$ using an encoder function

$$c_{1:n} = E_{\text{NC}}(x_{1:n}) = \text{biRNN}^*(x_{1:n})$$



Encoder

Attention Mechanism

- Special case $\text{attend}(c_{1:n}, \hat{t}_{1:j}) = c_n$
- Soft Context
 - At each stage the decoder sees a weighted average of the vectors
 - At stage j the soft attention produces a mixture vector c^j

$$c^j = \sum_{i=1}^n \alpha_{[i]}^j \cdot c_i$$

- The values of $\alpha_{[i]}^j$

$$\bar{\alpha}^j = \bar{\alpha}_{[1]}^j, \dots, \bar{\alpha}_{[n]}^j =$$

$$= \text{MLP}^{\text{att}}([s_j; c_1]), \dots, \text{MLP}^{\text{att}}([s_j; c_n])$$

$$\alpha^j = \text{softmax}(\bar{\alpha}_{[1]}^j, \dots, \bar{\alpha}_{[n]}^j)$$

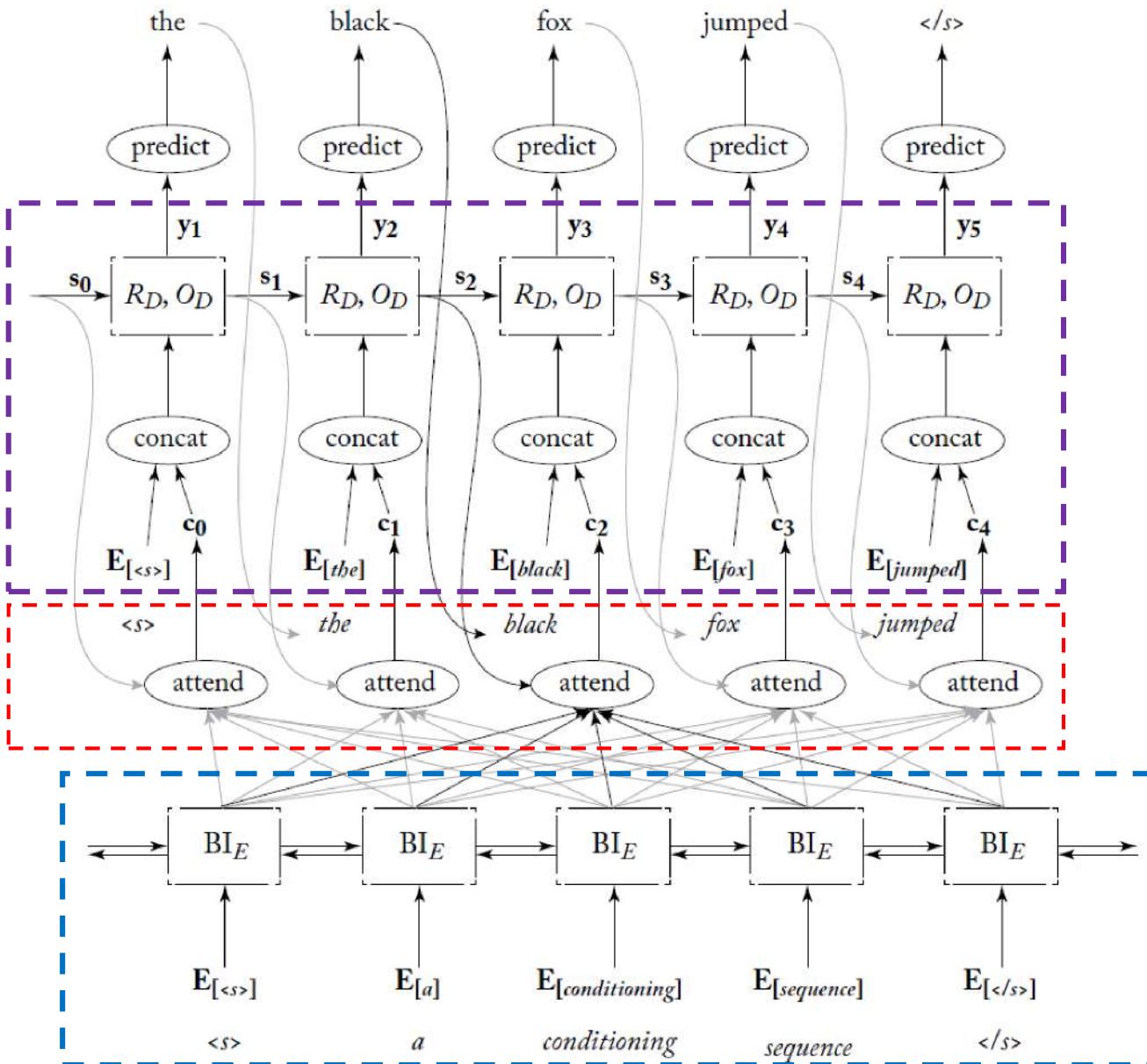
$$\text{attend}(c_{1:n}, \hat{t}_{1:j}) = c^j$$

$$c^j = \sum_{i=1}^n \alpha_{[i]}^j \cdot c_i$$

$$\alpha^j = \text{softmax}(\bar{\alpha}_{[1]}^j, \dots, \bar{\alpha}_{[n]}^j)$$

$$\bar{\alpha}_{[i]}^j = \text{MLP}^{\text{att}}([s_j; c_i]),$$

Sequence-to-Sequence Generation with Attention



$$p(t_{j+1} = k \mid \hat{t}_{1:j}, \mathbf{x}_{1:n}) = f(O_{\text{dec}}(s_{j+1}))$$

$$s_{j+1} = R_{\text{dec}}(s_j, [\hat{t}_j; c^j])$$

$$c^j = \sum_{i=1}^n \alpha_{[i]}^j \cdot c_i$$

$$\mathbf{c}_{1:n} = \text{biRNN}_{\text{enc}}^*(\mathbf{x}_{1:n})$$

$$\alpha^j = \text{softmax}(\bar{\alpha}_{[1]}^j, \dots, \bar{\alpha}_{[n]}^j)$$

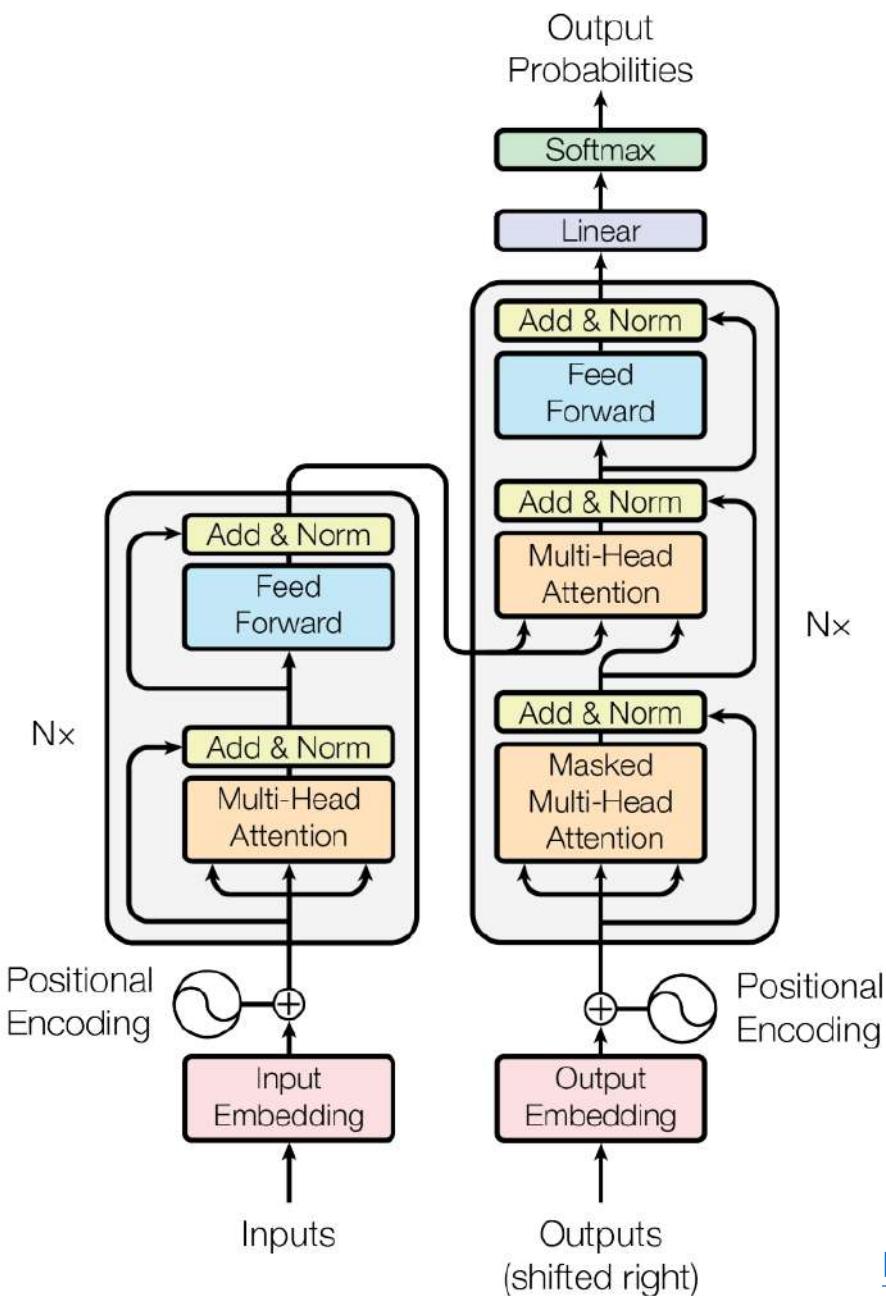
$$\bar{\alpha}_{[i]}^j = \text{MLP}^{\text{att}}([s_j; c_i])$$

$$\hat{t}_j \sim p(t_j \mid \hat{t}_{1:j-1}, \mathbf{x}_{1:n})$$

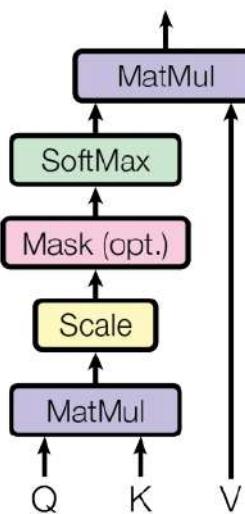
$$f(z) = \text{softmax}(\text{MLP}^{\text{out}}(z))$$

$$\text{MLP}^{\text{att}}([s_j; c_i]) = \mathbf{v} \tanh([s_j; c_i]U + \mathbf{b}).$$

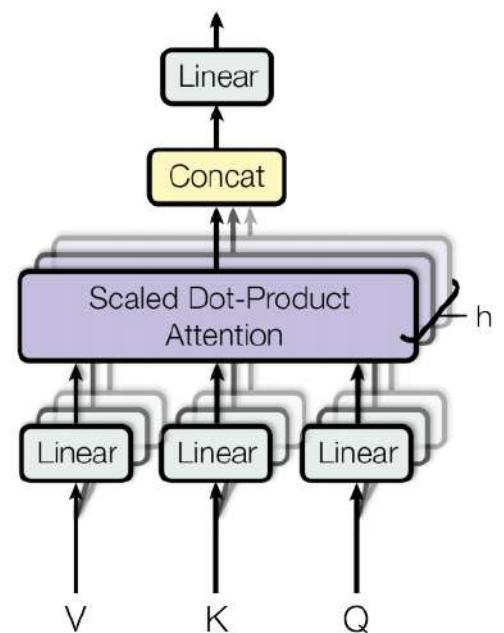
The Transformer



Scaled Dot-Product Attention



Multi-Head Attention



Revisit Language Models

- Predict each word within its sentence context in a large corpus
- Two factors for downstream NLP tasks
 - Is the representation of a word capturing the full context of that word in a sentence?
 - Full context is the entire sequence of words that precede and follow a word in a sentence
 - Is the model multilayered to output multiple representations for a word, one from each layer?
 - Representations of a word from the different layers of a model have been found to be useful for downstream syntactic and semantic tasks

Language Modeling Approaches

- Autoregressive approach
 - Left to right prediction
 - Right to left prediction
- Masked language approach
 - Prediction of a word using all other words in a sentence
 - Mask one or more of words in a sentence and model predicts those masked words given the other words in sentence
 - Bert
- The permutation approach
 - Decide what tokens are involved in the computation of a vector
 - Pick a permutation and compute the layer vector for that position by only using all word vectors that precede it in the sequence
 - words 6, 1 ,2, and 9 for computing 3 in 612934578
 - XLNet

*Order (or position) of words
in the sentence is taken into account*

Autoregressive Approach

Alaska

Alaska is

Alaska is about

Alaska is about twelve

Alaska is about twelve times

Alaska is about twelve times larger

Alaska is about twelve times larger than

Alaska is about twelve times larger than New

Alaska is about twelve times larger than New York

Left-to-right prediction

Word prediction using context from only one side

York

New York

than New York

larger than New York

times larger than New York

twelve times larger than New York

about twelve times larger than New York

is about twelve times larger than New York

Alaska is about twelve times larger than New York

Right-to-left prediction

Masked Language Approach

Word prediction using context from both sides (e.g. BERT)

The words in red in BERT is masked out
Replaced with a special token [MASK]
BERT masks about 15% of words in a sentence and
use the context words to predict it.

Alaska is about twelve times larger than New York

Alaska **is** about twelve times larger than New York

Alaska is **about** twelve times larger than New York

Alaska is about **twelve** times larger than New York

Alaska is about twelve **times** larger than New York

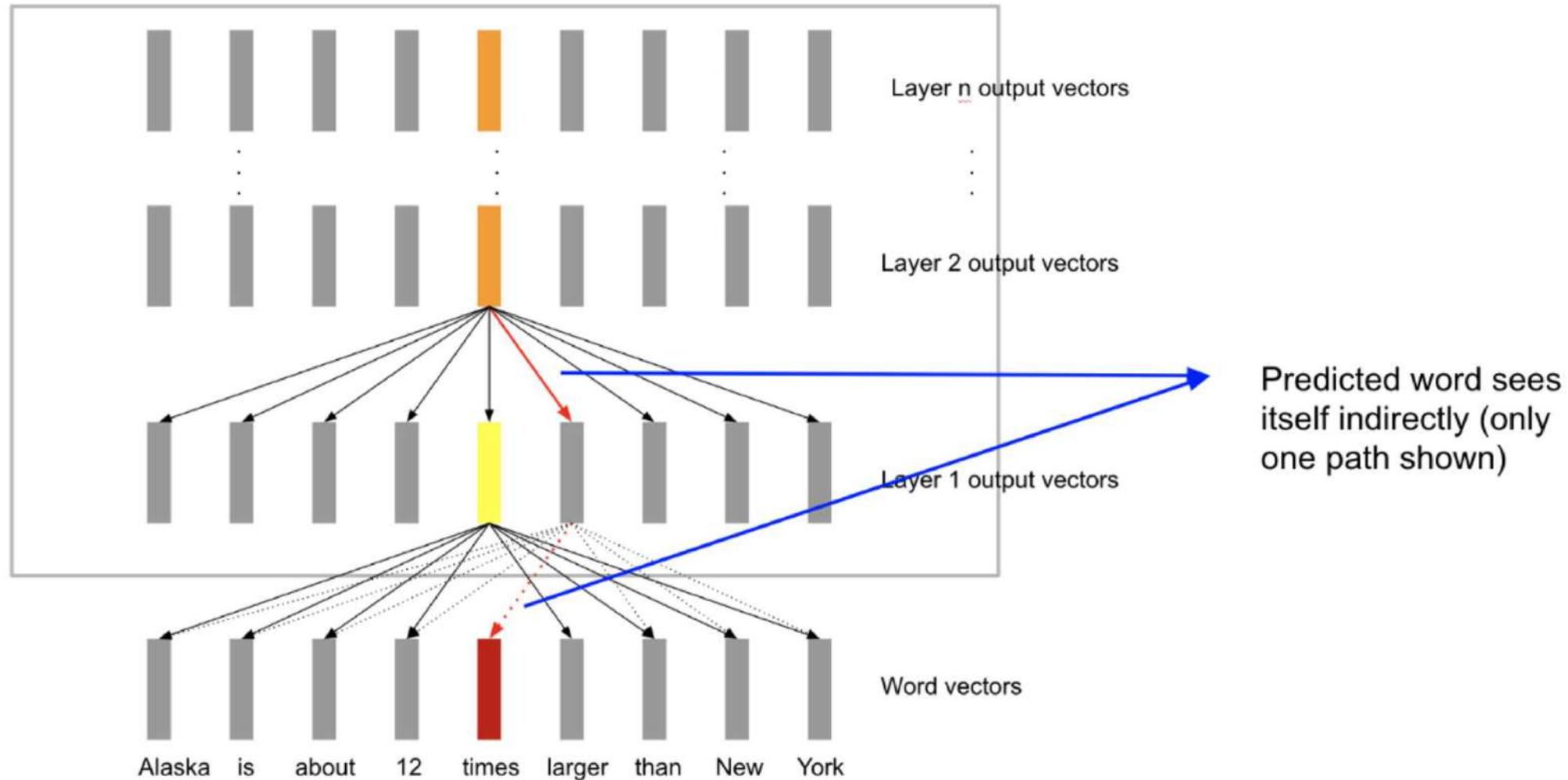
Alaska is about twelve times **larger** than New York

Alaska is about twelve times larger **than** New York

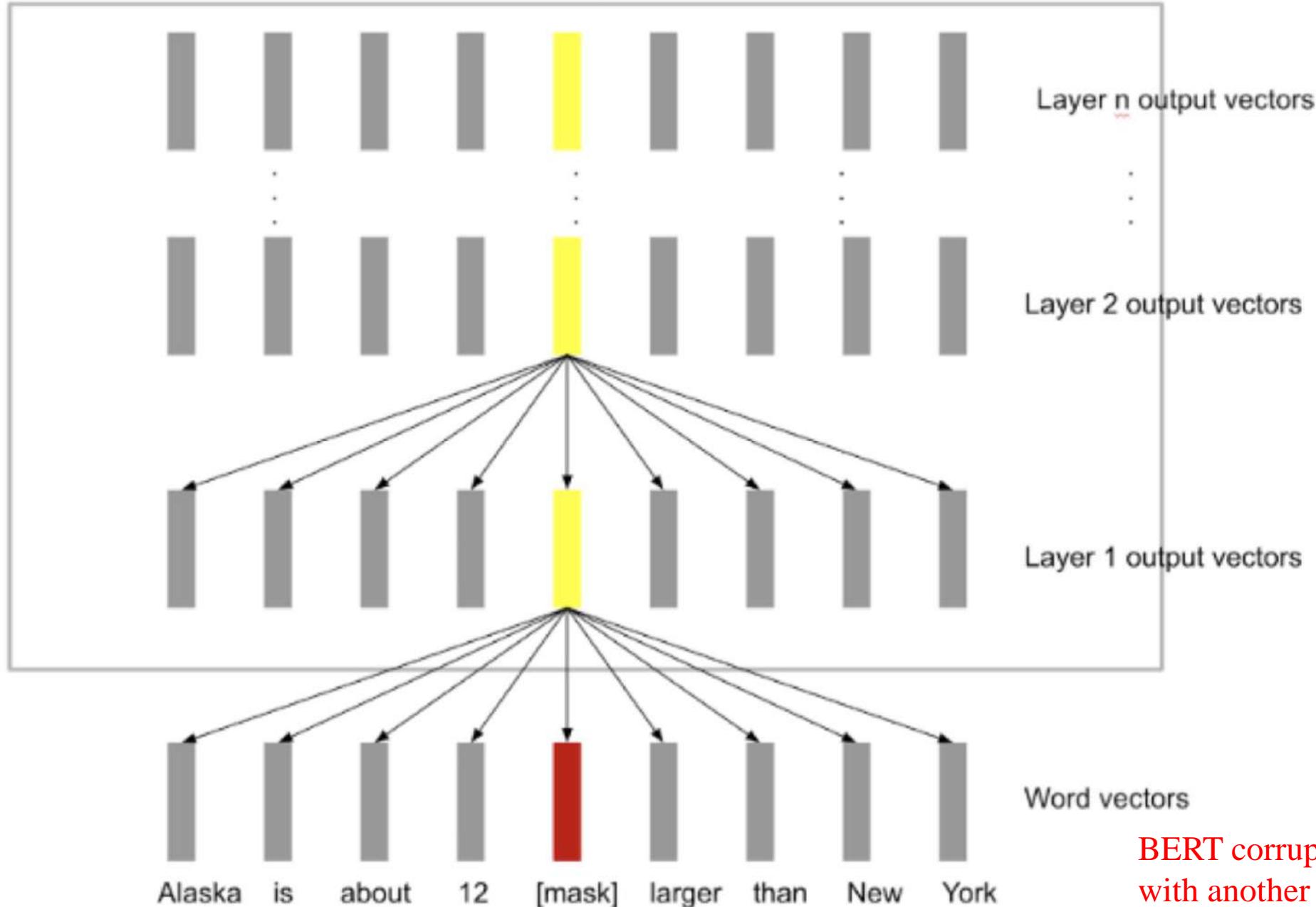
Alaska is about twelve times larger than **New** York

Alaska is about twelve times larger than **New York**

How to use context from both sides of a word for prediction in a multilayered model?

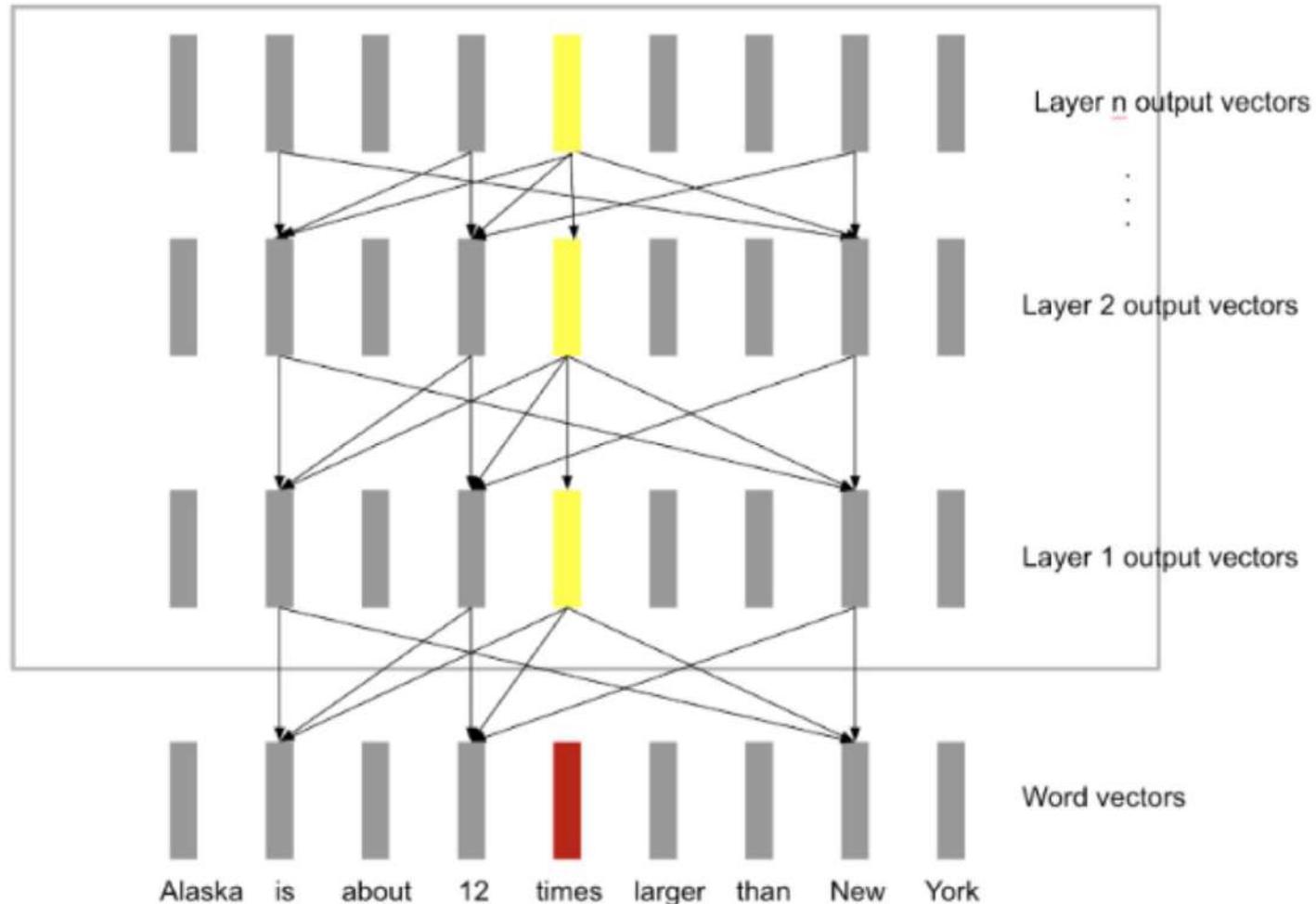


BERT solution:- Corrupt/replace the predicted town



BERT corrupts/replaces ~15% of tokens with another token [mask]

XLNet solution:- Predict using contexts that do not expose currently predicted token (even indirectly)



$n!$ Prediction contexts available for a sentence with n tokens

9! permutations—362,880

words 6,1,2,9,3, and 4 for computing the vector for “times” in a permutation, say 612934578

Predict a word at a position in a sentence by using a subset of its bidirectional context
Take into account the word is itself not seen using the permutation scan rule

Learn Deep Representations

- Learning multiple representations of a word using a deep model
 - Lower layer representations of certain models for syntactic tasks
 - Higher layer representations for semantic tasks

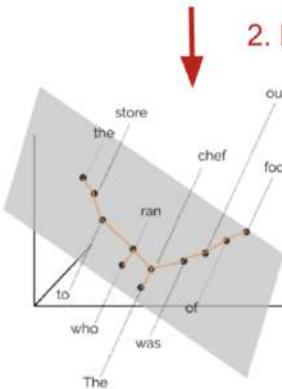
Finding Syntax with Structural Probes

The chef who ran to the store was out of food



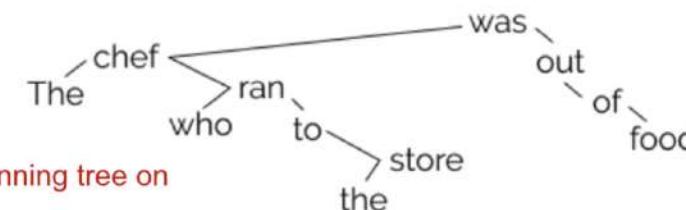
1. BERT transforms words to vectors

$$\begin{bmatrix} .4 \\ -.2 \\ .3 \end{bmatrix} \begin{bmatrix} .1 \\ .9 \\ -.2 \end{bmatrix} \begin{bmatrix} .3 \\ -.4 \\ .2 \end{bmatrix} \begin{bmatrix} .7 \\ -.4 \\ 0 \end{bmatrix} \begin{bmatrix} .4 \\ 0 \\ -.5 \end{bmatrix} \begin{bmatrix} .1 \\ -.6 \\ .2 \end{bmatrix} \begin{bmatrix} .3 \\ .1 \\ -.6 \end{bmatrix} \begin{bmatrix} .1 \\ .9 \\ -.8 \end{bmatrix} \begin{bmatrix} .3 \\ .1 \\ .8 \end{bmatrix} \begin{bmatrix} -.8 \\ .3 \\ -.6 \end{bmatrix} \begin{bmatrix} 0 \\ .7 \\ -.9 \end{bmatrix}$$



2. Linear Transform of word vectors

3. Minimum spanning tree on distances



Recovered Syntax tree from transformed BERT vectors

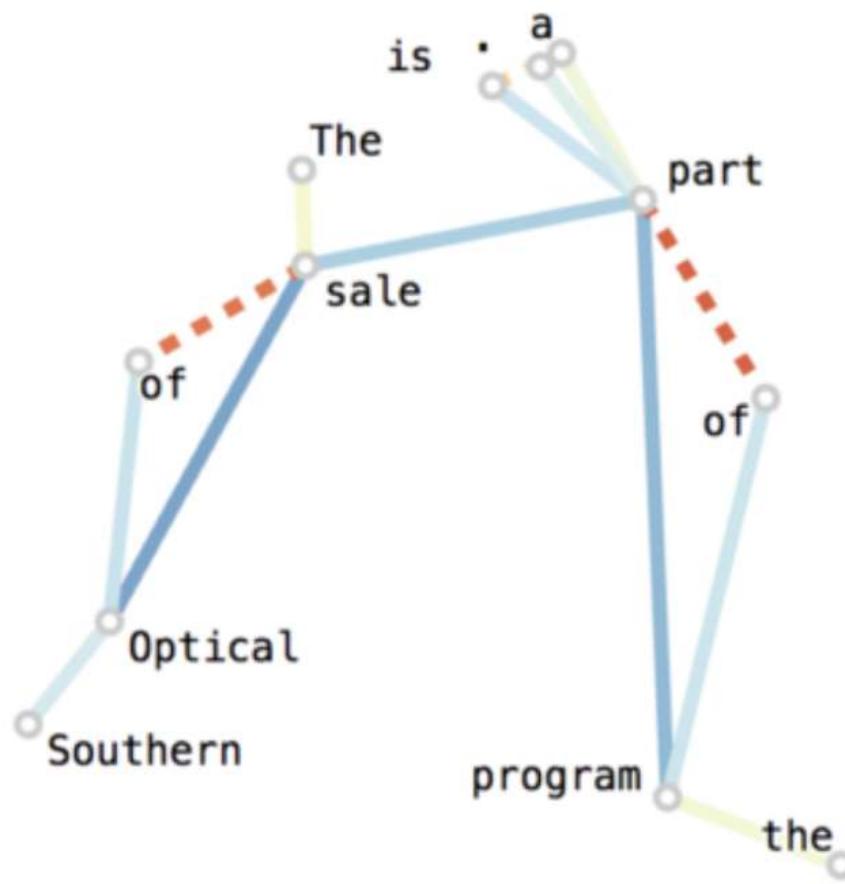
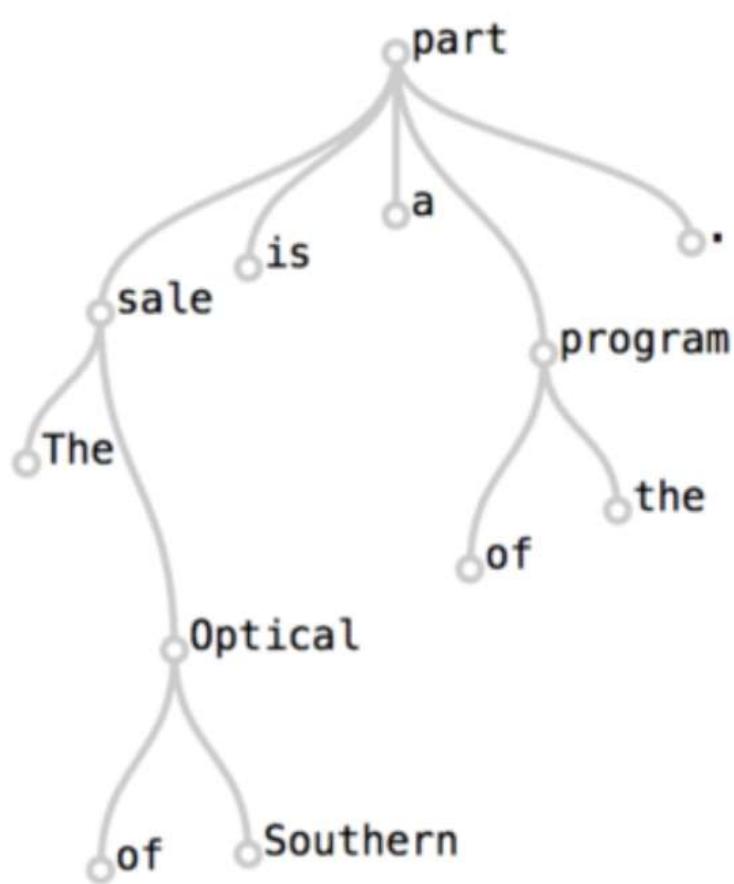


Same as

The chef who ran to the stores was out of food

Syntax tree of sentence

"The sale of Southern Optical is a part of the program."



Ratio between d^2 and tree distance



— Ground truth dependency
---- No ground truth dependency, $d^2 < 1.5$

German article “die”



Was der Fall ist, **die** Tatsache,
ist das Bestehen von Sachverhalten.

über **die** Verhandlungen
der Königl.

single person dies

multiple people die



Chernenko became the first Soviet leader to **die** in less than three years

Vaughan's ultimate fantasy was to **die** in a head-on collision with movie star Elizabeth Taylor

Over 60 people **die** and over 100 are unaccounted for.

Many more **die** from radiation sickness, starvation and cold.

a playing die



Players must always move a token according to the **die** value

The faces of a **die** may be placed clockwise or counterclockwise

References of This Lecture

- Y. Bengio et al., “A neural probabilistic language model.” J. Mach. Learn. Res. 3, pp. 1137-1155, 2003.
- T. Mikolov, and et al., ”Efficient estimation of word representations in vector space,” In Proc. of Workshop at ICLR, 2013.
- <http://blog.csdn.net/itplus/article/details/37969979>
- Yoav Goldverg, Neural Network Methods for Natural Language Processing, Chapters 9 and 14

References

- [XLNet: Generalized Autoregressive Pretraining for Language Understanding, Zhilin Yang et al, June 2019](#)
(<https://arxiv.org/pdf/1906.08237.pdf>)
- [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Devlin et al, Oct 2018](#)
(<https://arxiv.org/abs/1810.04805>)

Lecture 6: Part-of-Speech Tagging

Word Classes: Parts of Speech

- traditional parts of speech
 - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc.
 - Called: parts-of-speech, lexical categories, word classes, morphological classes, lexical tags, etc.

POS examples

- N noun *chair, bandwidth, pacing*
- V verb *study, debate, munch*
- ADJ adjective *purple, tall, ridiculous*
- ADV adverb *unfortunately, slowly*
- P preposition *of, by, to*
- PRO pronoun *I, me, mine*
- DET determiner *the, a, that, those*

Words' Syntactic Functions

- **Nouns** refer to entities in the world like people, animals and things
- **Determiners** describe the particular reference of a noun
- **Adjectives** describe the properties of nouns
- **Verbs** are used to describe actions, activities and states
- **Adverbs** modify a verb in the same way as adjectives modify nouns

Words' Syntactic Functions

- **Prepositions** are typically small words that express spatial or time relationships.
- **Conjunctions** and **complementizers** link two words, phrases or clauses
 - A complementizer is a conjunction which marks a complement clause.
 - I know *that* he is here.

Open and Closed Classes

- Closed class: a small fixed membership
 - usually **function words**
 - prepositions, particles, determiners, pronouns, conjunctions, numerals
- Open class: new ones can be created all the time
 - English has 4: Nouns, Verbs, Adjectives, Adverbs

POS Tagging

- Words often have more than one POS: *back*
 - The *back* door = JJ (後面)
 - On my *back* = NN (背部)
 - Win the voters *back* = RB (回)
 - Promised to *back* the bill = VB (背書)
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

POS Tagging

- The process of assigning a part-of-speech or lexical class marker to each word in a collection.
- | WORD | tag |
|------|-----|
|------|-----|

the	DET
koala	N
put	V
the	DET
keys	N
on	P
the	DET
table	N

Why POS Tagging is Useful

refuse (n: 垃圾 , v: 拒絕)

lead (n: 鉛 , v: 領路)

He will refuse to lead.

There is lead in the refuse.

Why is POS Tagging Useful?

- First step of a vast number of practical tasks
 - Speech synthesis
 - How to pronounce “lead”?
 - INsult inSULT
 - OObject obJECT
 - OVERflow overFLOW
 - DIScount disCOUNT
 - CONtent conTENT
 - Parsing
 - Helpful to know parts of speech before you start parsing
 - Information extraction
 - Finding names, relations, etc.
 - Machine Translation

POS Tagging: Choosing a Tagset

- To do POS tagging, we need to choose a standard set of tags to work with
- Could pick very coarse tagsets
 - N, V, Adj, Adv.
- More commonly used set is the finer grained, “Penn TreeBank tagset”, 45 tags
 - PRP\$, WRB, WP\$, VBG
- Even more fine-grained tagsets exist

The Penn Treebank Part-of-Speech Tagset

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	<i>and, but, or</i>	PDT	predeterminer	<i>all, both</i>	VBP	verb non-3sg present	<i>eat</i>
CD	cardinal number	<i>one, two</i>	POS	possessive ending	's	VBZ	verb 3sg pres	<i>eats</i>
DT	determiner	<i>a, the</i>	PRP	personal pronoun	<i>I, you, he</i>	WDT	wh-determ.	<i>which, that</i>
EX	existential ‘there’	<i>there</i>	PRP\$	possess. pronoun	<i>your, one's</i>	WP	wh-pronoun	<i>what, who</i>
FW	foreign word	<i>mea culpa</i>	RB	adverb	<i>quickly</i>	WP\$	wh-possess.	<i>whose</i>
IN	preposition/ subordin-conj	<i>of, in, by</i>	RBR	comparative adverb	<i>faster</i>	WRB	wh-adverb	<i>how, where</i>
JJ	adjective	<i>yellow</i>	RBS	superlatv. adverb	<i>fastest</i>	\$	dollar sign	\$
JJR	comparative adj	<i>bigger</i>	RP	particle	<i>up, off</i>	#	pound sign	#
JJS	superlative adj	<i>wildest</i>	SYM	symbol	<i>+, %, &</i>	"	left quote	‘ or “
LS	list item marker	<i>1, 2, One</i>	TO	“to”	<i>to</i>	"	right quote	’ or ”
MD	modal	<i>can, should</i>	UH	interjection	<i>ah, oops</i>	(left paren	[, (, {, <
NN	sing or mass noun	<i>llama</i>	VB	verb base form	<i>eat</i>)	right paren],), }, >
NNS	noun, plural	<i>llamas</i>	VBD	verb past tense	<i>ate</i>	,	comma	,
NNP	proper noun, sing.	<i>IBM</i>	VBG	verb gerund	<i>eating</i>	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	<i>Carolinas</i>	VBN	verb past part.	<i>eaten</i>	:	sent-mid punc	: ; ... - -

How Hard is POS Tagging?

Measuring Ambiguity

	87-tag Original Brown	45-tag Treebank Brown
Unambiguous (1 tag)	44,019	38,857
Ambiguous (2–7 tags)	5,490	8844
Details:		
2 tags	4,967	6,731
3 tags	411	1621
4 tags	91	357
5 tags	17	90
6 tags	2 (<i>well, beat</i>)	32
7 tags	2 (<i>still, down</i>)	6 (<i>well, set, round, open, fit, down</i>)
8 tags		4 (<i>'s, half, back, a</i>)
9 tags		3 (<i>that, more, in</i>)

Methods for POS Tagging

1. Rule-based tagging
 - (ENGTWOL; ENGlish TWO Level analysis)
2. Stochastic
 - Probabilistic sequence models
 - HMM (Hidden Markov Model) tagging
 - MEMMs (Maximum Entropy Markov Models)
3. Bidirectional LSTM
4. Transformation-Based Tagger
5. More on

[http://aclweb.org/aclwiki/index.php?title=POS_Tagging_\(State_of_the_art\)#WSJ](http://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)#WSJ)

System name	Short description	Main publication	Software	Extra Data?***	All tokens	Unknown words	License
TnT*	Hidden Markov model	Brants (2000)	TnT	No	96.46%	85.86%	Academic/research use only (license)
MElt	Maximum entropy Markov model with external lexical information	Denis and Sagot (2009)	Alpage linguistic workbench	No	96.96%	91.29%	CeCILL-C
GENIA Tagger**	Maximum entropy cyclic dependency network	Tsuruoka, et al (2005)	GENIA	No	97.05%	Not available	Gratis for non-commercial usage
Averaged Perceptron	Averaged perceptron	Collins (2002)	Not available	No	97.11%	Not available	Unknown
Maxent easiest-first	Maximum entropy bidirectional easiest-first inference	Tsuruoka and Tsujii (2005)	Easiest-first	No	97.15%	Not available	Unknown
SVMTool	SVM-based tagger and tagger generator	Giménez and Márquez (2004)	SVMTool	No	97.16%	89.01%	LGPL 2.1
LAPOS	Perceptron based training with lookahead	Tsuruoka, Miyao, and Kazama (2011)	LAPOS	No	97.22%	Not available	MIT
Morče/COMPOST	Averaged perceptron	Spouštová et al. (2009)	COMPOST	No	97.23%	Not available	Non-free (academic-only)
Morče/COMPOST	Averaged perceptron	Spouštová et al. (2009)	COMPOST	Yes	97.44%	Not available	Unknown
Stanford Tagger 1.0	Maximum entropy cyclic dependency network	Toutanova et al. (2003)	Stanford Tagger	No	97.24%	89.04%	GPL v2+
Stanford Tagger 2.0	Maximum entropy cyclic dependency network	Manning (2011)	Stanford Tagger	No	97.29%	89.70%	GPL v2+
Stanford Tagger 2.0	Maximum entropy cyclic dependency network	Manning (2011)	Stanford Tagger	Yes	97.32%	90.79%	GPL v2+
LTAG-spinal	Bidirectional perceptron learning	Shen et al. (2007)	LTAG-spinal	No	97.33%	Not available	Unknown
SCCN	Semi-supervised condensed nearest neighbor	Søgaard (2011)	SCCN	Yes	97.50%	Not available	Unknown
CharWNN	MLP with neural character embeddings	dos Santos and Zadrozny (2014)	Not available	No	97.32%	89.86%	Unknown
structReg	CRF with structure regularization	Sun (2014)	Not available	No	97.36%	Not available	Unknown
Bi-LSTM-CRF	Bidirectional LSTM-CRF	Huang et al. (2015)	Not available	No	97.55%	Not available	Unknown
NLP4J	Dynamic feature induction	Choi (2016)	NLP4J	Yes	97.64%	92.03%	Apache 2
Flair	Bidirectional LSTM-CRF with contextual string embeddings	Akbik et al. (2018)	Flair	Yes	97.85%	Not available	MIT

[https://aclweb.org/aclwiki/POS_Tagging_\(State_of_the_art\)#WSJ](https://aclweb.org/aclwiki/POS_Tagging_(State_of_the_art)#WSJ)

HMM Part-of-Speech Tagging

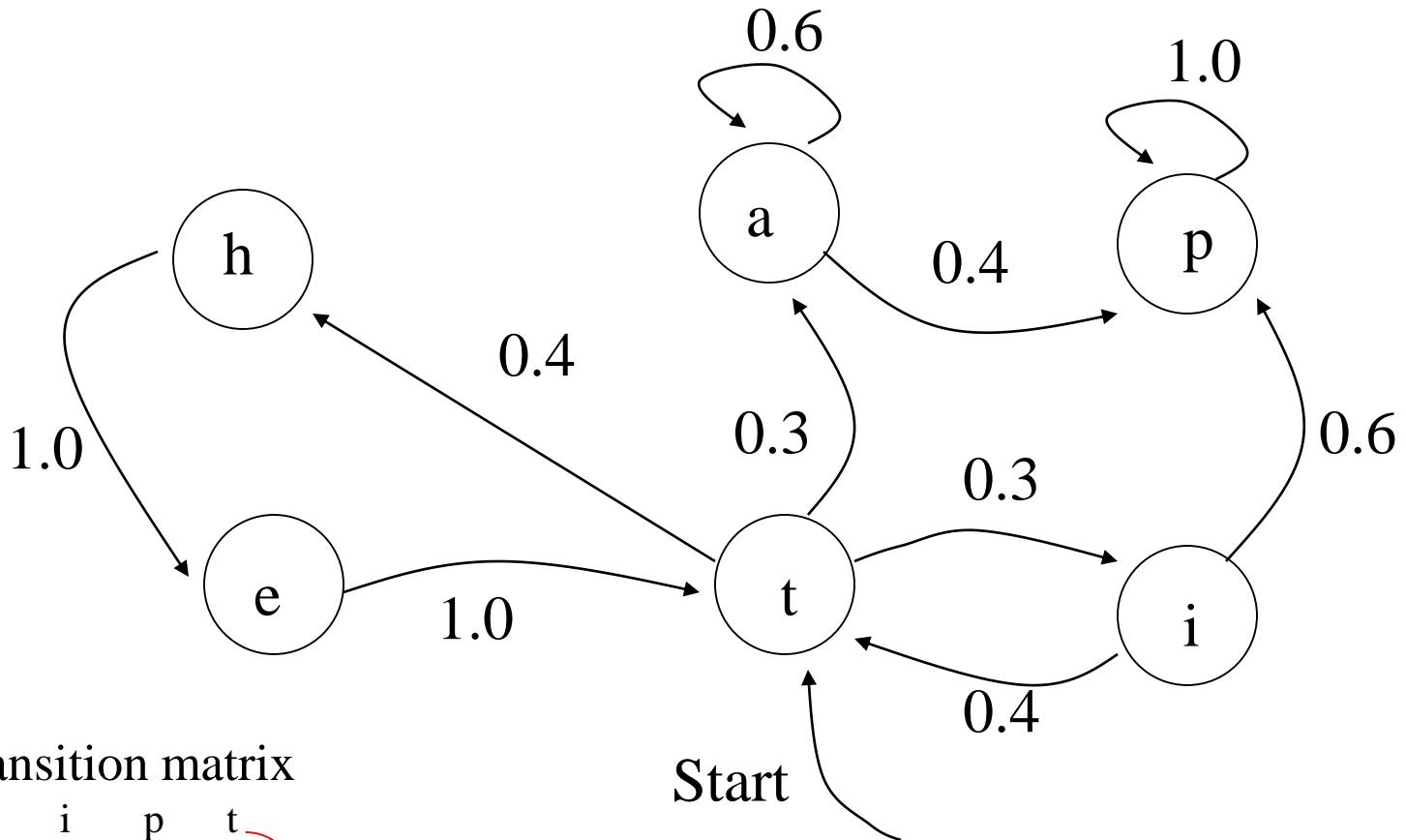
Markov Models

- Markov models are statistical tools useful for many NLP applications.
- They model a sequence of random variables that are not necessarily independent.
- They rely on two assumptions
 - *Limited Horizon*
 - *Time Invariant*

Markov Assumptions

- Let $X=(X_1, \dots, X_T)$ be a sequence of random variables taking values in some finite set $S=\{s_1, \dots, s_N\}$, the state space, the Markov properties are:
 - $X=(我_1, 上_2, 一堂_3, 有趣的_4, 課程_5)$
- **Limited Horizon**: $P(X_{t+1} = s_k | X_1, \dots, X_t) = P(X_{t+1} = s_k | X_t)$
i.e., a word's tag only depends on the previous tag.
 - $P(X_5=課程|X_1=我, X_2=上, X_3=一堂, X_4=有趣的)=P(X_5=課程|X_4=有趣的)$
- **Time Invariant**: $P(X_{t+1} = s_k | X_t) = P(X_2 = s_k | X_1)$
i.e., the dependency does not change over time.
 - 有趣的₁ 課程₂ 很₃ 多₄
 - 我₁ 上₂ 一堂₃ 有趣的₄ 課程₅
- If X possesses these properties, then X is said to be a **Markov Chain**.

Example of a Markov Chain



stochastic transition matrix

	a	e	h	i	p	t
a	0.6	0	0	0	0.4	0
e	0	0	0	0	0	1.0
h	0	1.0	0	0	0	0
i	0	0	0	0	0.6	0.4
p	0	0	0	0	1.0	0
t	0.3	0	0.4	0.3	0	0

initial state: $\pi_t=1$

Markov chain can be described by a stochastic transition matrix A:

$$a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i)$$

Here, $a_{ij} \geq 0, \forall i, j$ and $\sum_{j=1}^N a_{ij} = 1, \forall i$.

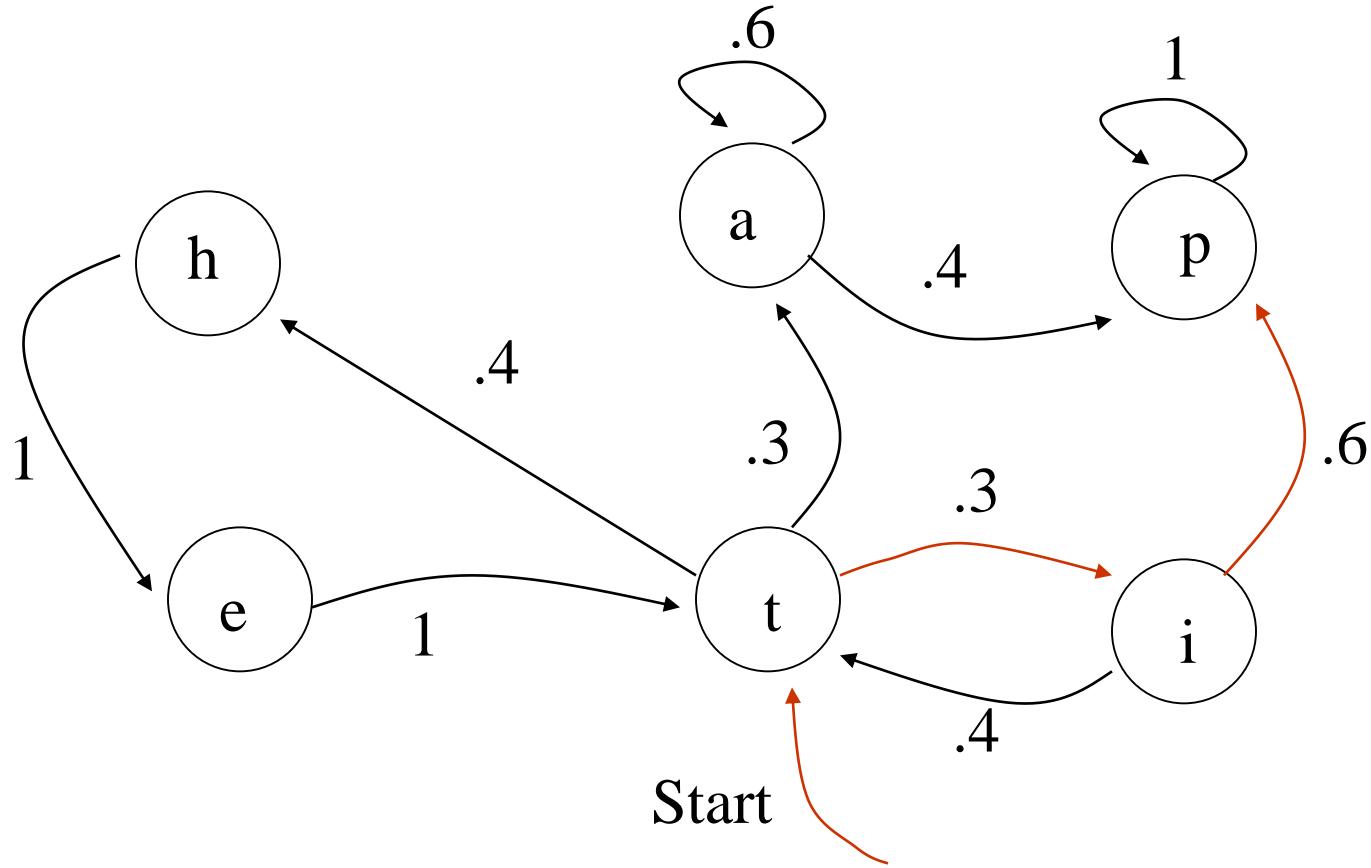
The probabilities of different initial states for the Markov chain:

$$\pi_i = P(X_1 = s_i)$$

Here, $\sum_{i=1}^N \pi_i = 1$.

The probability of a sequence of states X_1, \dots, X_T :

$$\begin{aligned} P(X_1, \dots, X_T) &= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1, X_2)\dots P(X_T \mid X_1, \dots, X_{T-1}) \\ &= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_2)\dots P(X_T \mid X_{T-1}) \\ &= \pi_{X_1} \prod_{t=1}^{T-1} a_{X_t X_{t+1}} \quad \text{bigram model} \end{aligned}$$



$$\begin{aligned}
 P(t, i, p) &= P(X_1 = t)P(X_2 = i | X_1 = t)P(X_3 = p | X_2 = i) \\
 &= 1.0 \times 0.3 \times 0.6 \\
 &= 0.18
 \end{aligned}$$

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states.
- In part-of-speech tagging (and many other tasks) the symbols do not uniquely determine the states.
 - The output symbols are **words**
 - But the hidden states are **part-of-speech tags**
- A Hidden Markov Model is an extension of a Markov chain in which the input symbols are not the same as the states.

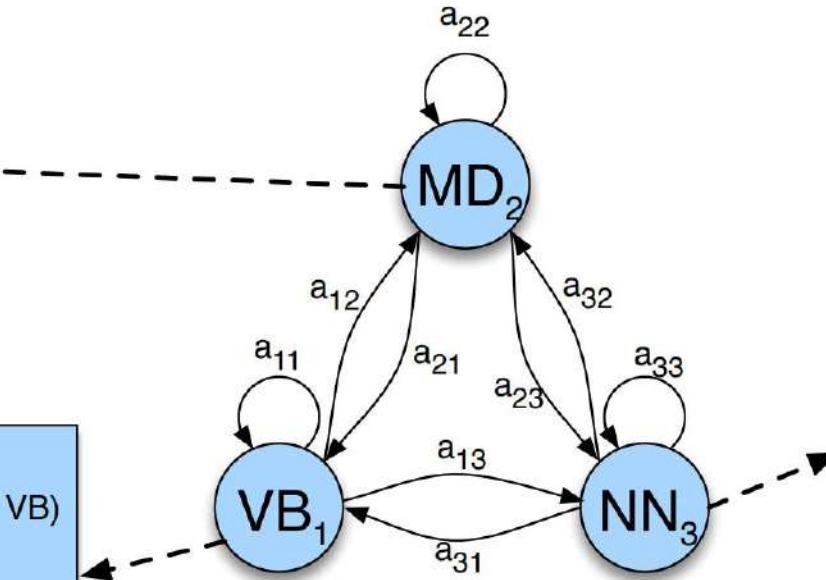
Hidden Markov Models

- States $Q = q_1, q_2 \dots q_N$;
- Observations $O = o_1, o_2 \dots o_N$;
 - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, \dots v_{|V|}\}$
- Transition probabilities
 - Transition probability matrix $A = \{a_{ij}\}$
$$a_{ij} = P(q_t = j | q_{t-1} = i) \quad 1 \leq i, j \leq N$$
- Observation likelihoods
 - Output probability matrix $B = \{b_i(k)\}$
$$b_i(k) = P(X_t = o_k | q_t = i)$$
- Special initial probability vector π
$$\pi_i = P(q_1 = i) \quad 1 \leq i \leq N$$

B₂
 $P(\text{"aardvark"} \mid \text{MD})$
...
 $P(\text{"will"} \mid \text{MD})$
...
 $P(\text{"the"} \mid \text{MD})$
...
 $P(\text{"back"} \mid \text{MD})$
...
 $P(\text{"zebra"} \mid \text{MD})$

B₁
 $P(\text{"aardvark"} \mid \text{VB})$
...
 $P(\text{"will"} \mid \text{VB})$
...
 $P(\text{"the"} \mid \text{VB})$
...
 $P(\text{"back"} \mid \text{VB})$
...
 $P(\text{"zebra"} \mid \text{VB})$

B₃
 $P(\text{"aardvark"} \mid \text{NN})$
...
 $P(\text{"will"} \mid \text{NN})$
...
 $P(\text{"the"} \mid \text{NN})$
...
 $P(\text{"back"} \mid \text{NN})$
...
 $P(\text{"zebra"} \mid \text{NN})$



3 Problems

- Given this framework there are 3 problems that we can pose to an HMM
 - Given an observation sequence, what is the probability of that sequence given a model?
 - Given an observation sequence and a model, what is the most likely state sequence?
 - Given an observation sequence, infer the best model parameters for a skeletal model

Problem 1

- The probability of a sequence given a model.

Computing Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence O , determine the likelihood $P(O|\lambda)$.

- In classification tasks
 - Word spotting in ASR, language identification, speaker identification, author identification, etc.
 - Train one HMM model per class.
 - Given an observation, pass it to each model and compute $P(\text{seq}|\text{model})$.
 - Select the class of the highest probability.

Problem 2

- Find the most probable state sequence, given a model and an observation sequence

Decoding: Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, \dots, o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \dots q_T$.

- Used in tagging problems, where the tags correspond to hidden states
 - Almost any problem can be cast as a sequence labeling problem
- Viterbi solves problem 2

Problem 3

- Infer the best model parameters, given a skeletal model and an observation sequence.
- Fill in the A and B tables with the right numbers.
 - The numbers that make the observation sequence most likely.

Solutions

- Problem 1: Forward
- Problem 2: Viterbi
- Problem 3: EM
 - Baum-Welch (or forward-backward)

Problem 2. HMM Tagging as Decoding

- For any model, such as an HMM, that contains hidden variables, the task of determining which sequence of variables is the underlying source of some sequence of observations is called the **decoding** task.
- The task of the **decoder** is to find the best hidden sequence.

POS Tagging as Sequence Classification

- We are given a sentence (an “observation” or “a sequence of observations”)
 - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags that corresponds to this sequence of observations?
- Probabilistic view
 - Consider all possible sequences of tags
 - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words $w_1 \dots w_n$.

Getting to HMMs

- We want, out of all sequences of n tags $t_1 \dots t_n$ the single tag sequence such that

$$P(t_1 \dots t_n | w_1 \dots w_n) \text{ is highest.}$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

- Hat $\hat{}$ means “our estimate of the best one”
- $\operatorname{argmax}_x f(x)$ means “the x such that $f(x)$ is maximized”

Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

Know this.

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n)$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} \frac{P(w_1^n | t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(w_1^n | t_1^n)P(t_1^n)$$

Likelihood and Prior



$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} \frac{\overbrace{P(w_1^n | t_1^n)}^{\text{likelihood}}}{\overbrace{P(t_1^n)}^{\text{prior}}}$$

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^n P(w_i | t_i)$$



$$P(t_1^n) \approx \prod_{i=1}^n P(t_i | t_{i-1})$$

$$\hat{t}_1^n = \operatorname{argmax}_{t_1^n} P(t_1^n | w_1^n) \approx \operatorname{argmax}_{t_1^n} \prod_{i=1}^n \overbrace{P(w_i | t_i)}^{\text{emission transition}} \overbrace{P(t_i | t_{i-1})}^{\text{transition}}$$

Two Kinds of Probabilities

- Tag transition probabilities $P(t_i|t_{i-1})$
 - Determiners likely to precede adjs and nouns
 - That/DT flight/NN
 - The/DT yellow/JJ hat/NN
 - So we expect $P(NN|DT)$ and $P(JJ|DT)$ to be high
 - But $P(DT|JJ)$ to be low
 - Compute $P(NN|DT)$ by counting in a labeled corpus:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Two Kinds of Probabilities

- Word likelihood probabilities $P(w_i|t_i)$
 - VBZ (3sg Pres verb) likely to be “is”
 - Compute $P(\text{is}|VBZ)$ by counting in a labeled corpus:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(\text{is}|VBZ) = \frac{C(VBZ, \text{is})}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

Question

- If there are 30 or so tags in the Penn set
- And the average sentence is around 20 words...
- How many tag sequences do we have to enumerate to argmax over?

$$30^{20}$$

- To enumerate all paths given the input and use the model to assign probabilities to each (?)
 - dynamic programming

The Viterbi Algorithm

function VITERBI(*observations* of len T ,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix $viterbi[N,T]$

for each state s **from** 1 **to** N **do** ; initialization step

$viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$

$backpointer[s,1] \leftarrow 0$

for each time step t **from** 2 **to** T **do** ; recursion step

for each state s **from** 1 **to** N **do**

$viterbi[s,t] \leftarrow \max_{s'=1}^N viterbi[s',t-1] * a_{s',s} * b_s(o_t)$

$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^N viterbi[s',t-1] * a_{s',s} * b_s(o_t)$

$bestpathprob \leftarrow \max_{s=1}^N viterbi[s,T]$; termination step

$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^N viterbi[s,T]$; termination step

$bestpath \leftarrow$ the path starting at state $bestpathpointer$, that follows $backpointer[]$ to states back in time

return *bestpath*, *bestpathprob*

path prob. matrix $viterbi$

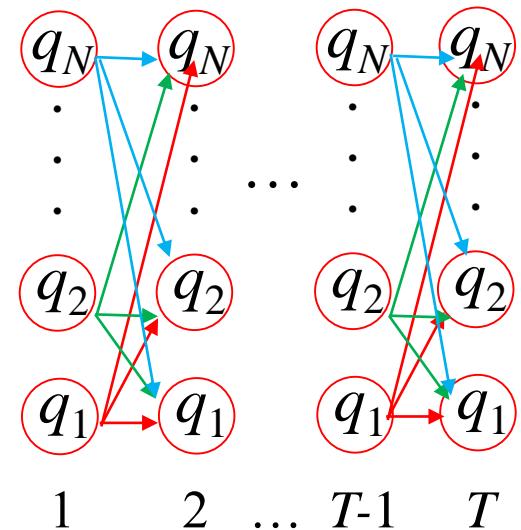


Figure 8.5 Viterbi algorithm for finding the optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A, B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.

The Viterbi Algorithm

```

function VITERBI(observations of len  $T$ ,state-graph of len  $N$ ) returns best-path, path-prob
    create a path probability matrix viterbi[ $N,T$ ]
    for each state  $s$  from 1 to  $N$  do
        viterbi[ $s,1$ ]  $\leftarrow \pi_s * b_s(o_1)$ 
        backpointer[ $s,1$ ]  $\leftarrow 0$ 
    for each time step  $t$  from 2 to  $T$  do
        for each state  $s$  from 1 to  $N$  do
            viterbi[ $s,t$ ]  $\leftarrow \max_{s'=1}^N viterbi[s',t-1] * a_{s',s} * b_s(o_t)$ 
            backpointer[ $s,t$ ]  $\leftarrow \operatorname{argmax}_{s'=1}^N viterbi[s',t-1] * a_{s',s} * b_s(o_t)$ 
        bestpathprob  $\leftarrow \max_{s=1}^N viterbi[s,T]$  ; termination step
        bestpathpointer  $\leftarrow \operatorname{argmax}_{s=1}^N viterbi[s,T]$  ; termination step
        bestpath  $\leftarrow$  the path starting at state bestpathpointer, that follows backpointer[] to states back in time
    return bestpath, bestpathprob

```

initialization step

; initialization step

; recursion step

; termination step

; termination step

1

Figure 8.5 Viterbi algorithm for finding the optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A, B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.

The Viterbi Algorithm

```

function VITERBI(observations of len  $T$ ,state-graph of len  $N$ ) returns best-path, path-prob
    create a path probability matrix viterbi[ $N,T$ ]
    for each state  $s$  from 1 to  $N$  do ; initialization step
        viterbi[ $s,1$ ]  $\leftarrow \pi_s * b_s(o_1)$ 
        backpointer[ $s,1$ ]  $\leftarrow 0$ 
    for each time step  $t$  from 2 to  $T$  do ; recursion step
        for each state  $s$  from 1 to  $N$  do
            
$$\text{viterbi}[s,t] \leftarrow \max_{s'=1}^N \text{viterbi}[s',t-1] * a_{s',s} * b_s(o_t)$$

            
$$\text{backpointer}[s,t] \leftarrow \operatorname{argmax}_{s'=1}^N \text{viterbi}[s',t-1] * a_{s',s} * b_s(o_t)$$

    
$$\text{bestpathprob} \leftarrow \max_{s=1}^N \text{viterbi}[s,T]$$
 ; termination step
    
$$\text{bestpathpointer} \leftarrow \operatorname{argmax}_{s=1}^N \text{viterbi}[s,T]$$
 ; termination step
    bestpath  $\leftarrow$  the path starting at state bestpathpointer, that follows backpointer[] to states back in time
    return bestpath, bestpathprob

```

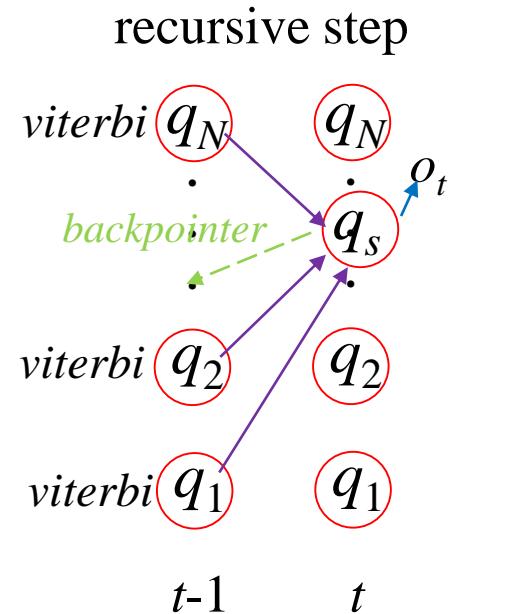


Figure 8.5 Viterbi algorithm for finding the optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A, B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.

The Viterbi Algorithm

function VITERBI(*observations* of len T ,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix $viterbi[N,T]$

for each state s from 1 to N do

$$viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$$

backpointer[s,1] ← 0

for each time step t **from** 2 **to** T **do**

; initialization step

; recursion step

for each state s from 1 to N do

$$viterbi[s,t] \leftarrow \max_{s'=1}^N viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^N viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$

$$bestpathprob \leftarrow \max_{s=1}^N viterbi[s, T]$$

; termination step

$$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^N viterbi[s, T]$$

; termination step

bestpath \leftarrow the path starting at state **bestpathpointer**, that follows backpointer[] to states back in time
return **bestpath**, **bestpathprob**

termination step

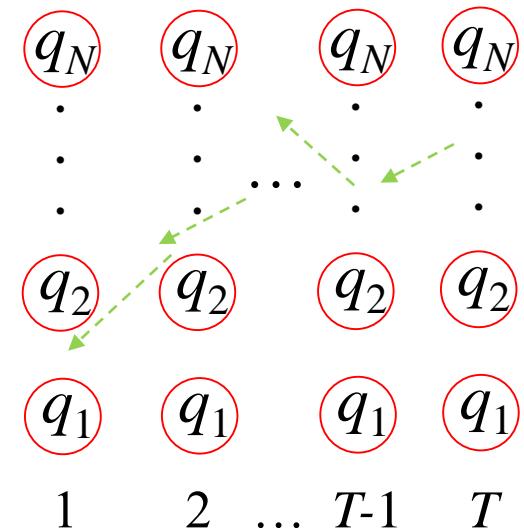
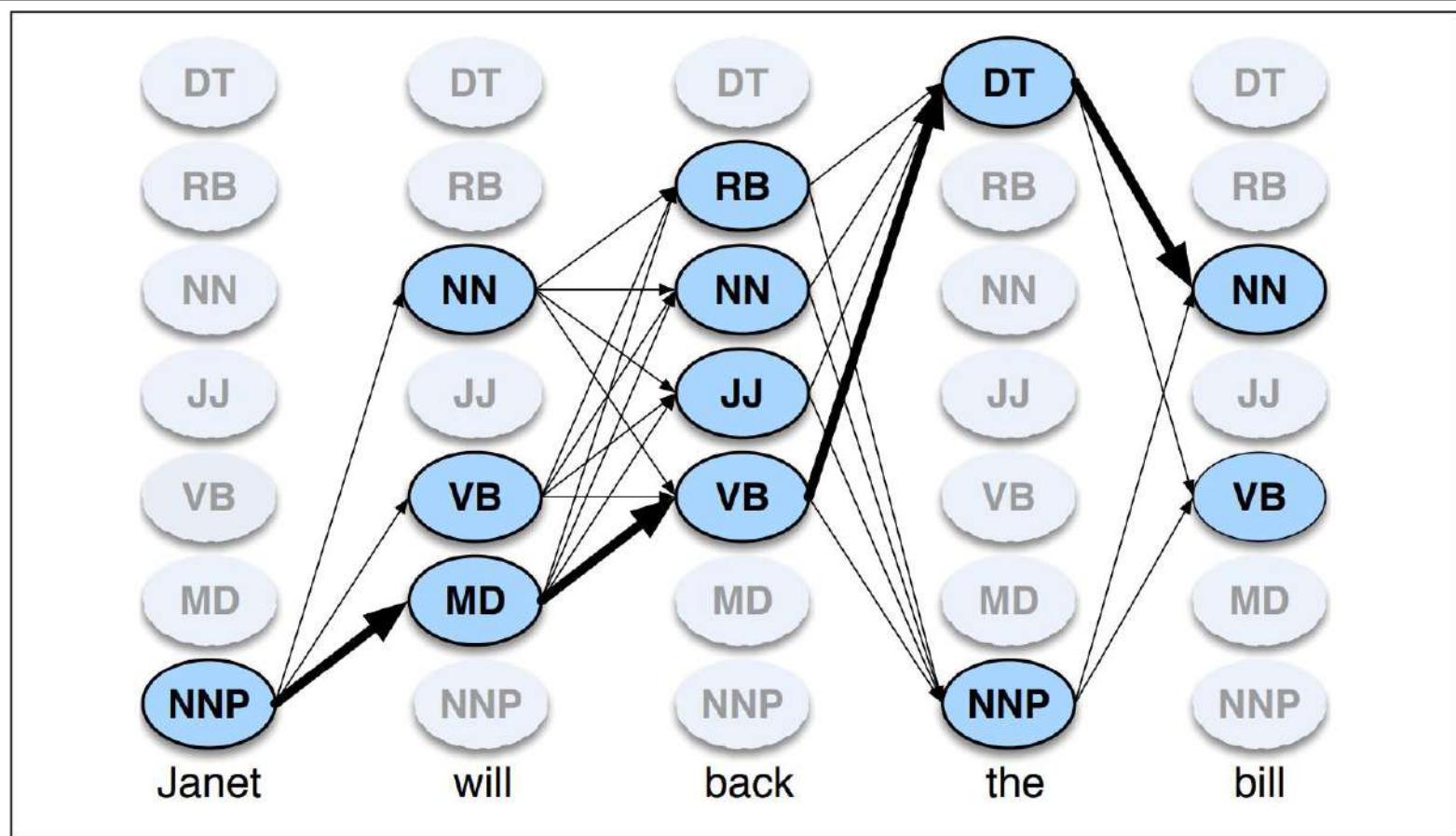


Figure 8.5 Viterbi algorithm for finding the optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A, B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.



$$v_t(j) = \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t)$$

$v_{t-1}(i)$ the **previous Viterbi path probability** from the previous time step

a_{ij} the **transition probability** from previous state q_i to current state q_j

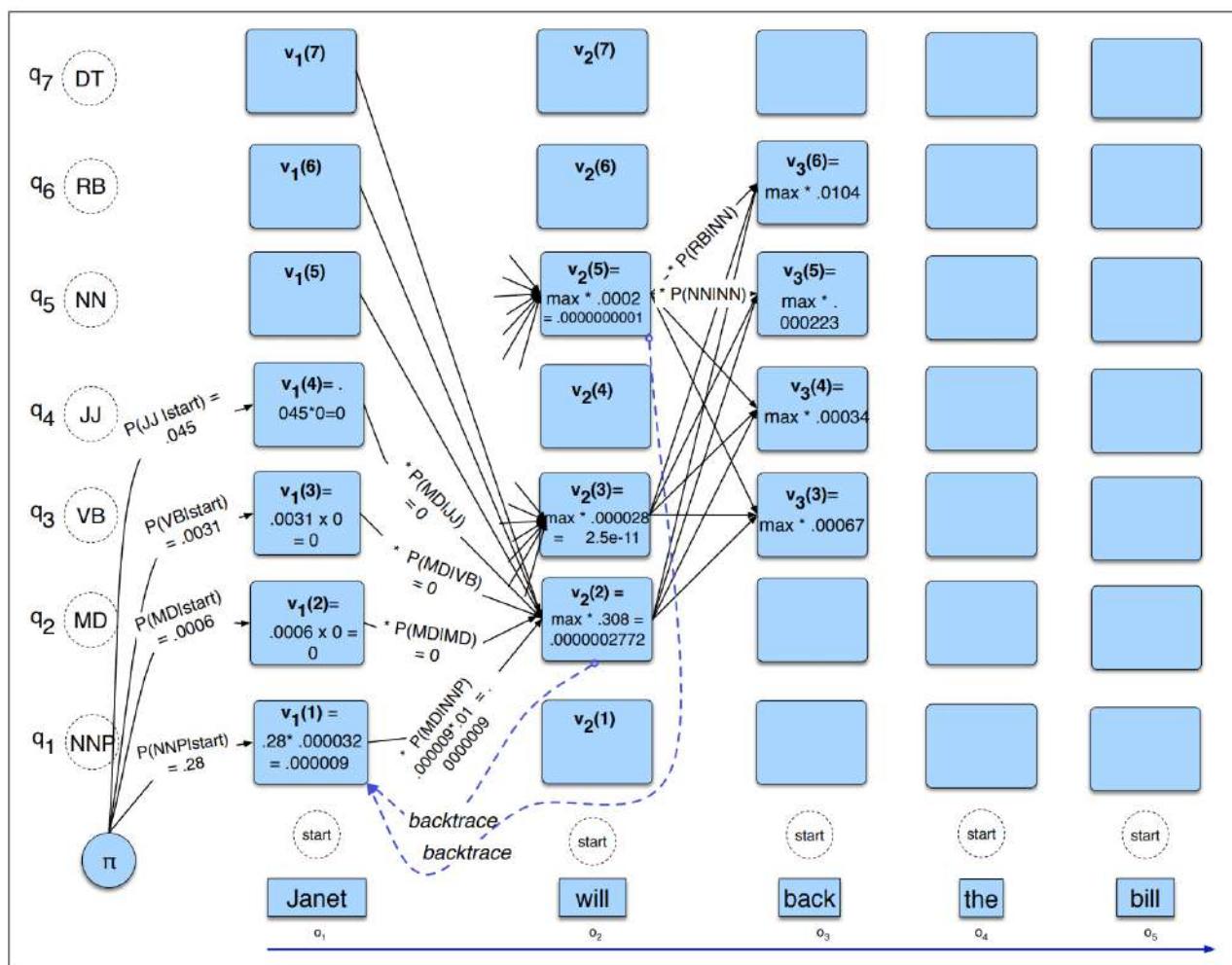
$b_j(o_t)$ the **state observation likelihood** of the observation symbol o_t given the current state j

	NNP	MD	VB	JJ	NN	RB	DT
< s >	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing.



Viterbi Summary

- Create an array
 - With columns corresponding to inputs
 - Rows corresponding to possible states
- Sweep through the array in one pass filling the columns left to right using our transition probs and observations probs
- Dynamic programming key is that we need only store the MAX prob path to each cell, (not all paths).

Beam Search

- Keep a fixed number of states instead of beam width all N current states

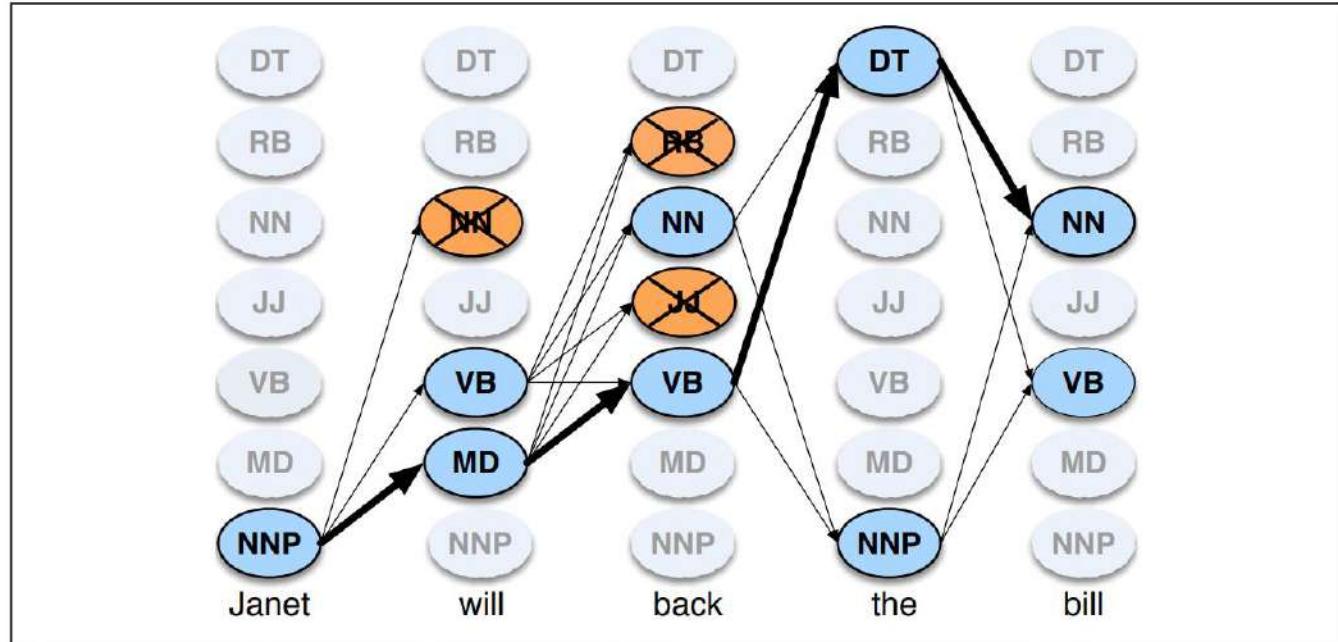
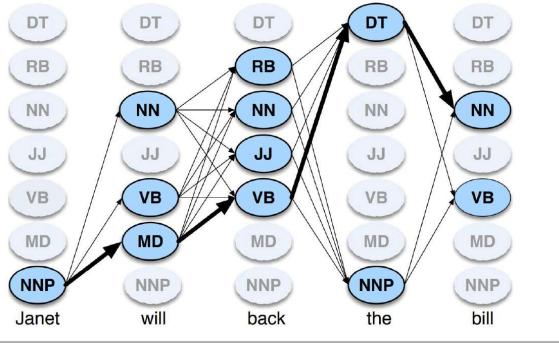


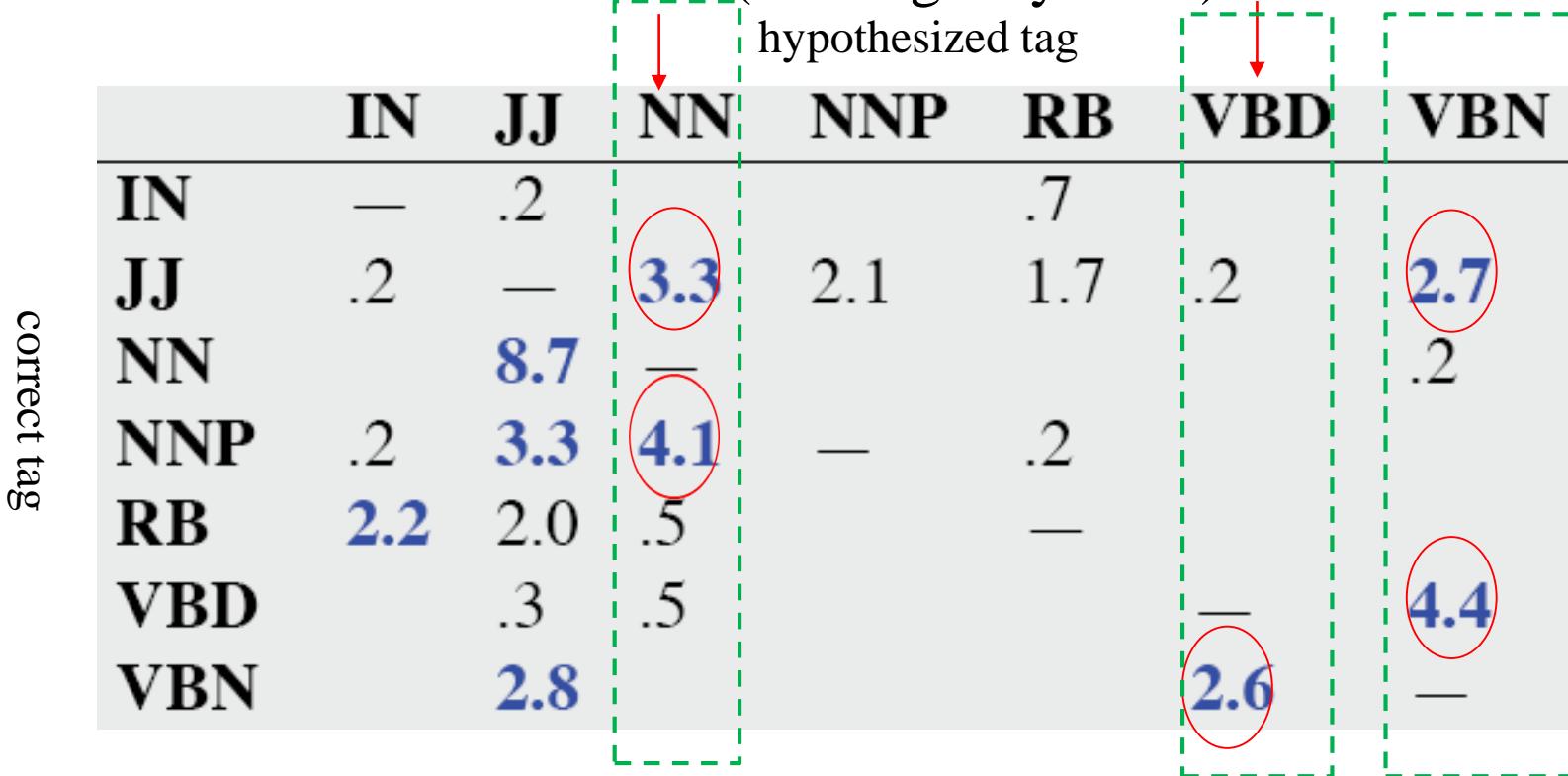
Figure 8.11 A beam search version of Fig. 8.6, showing a beam width of 2. At each time

Evaluation

- How do you evaluate POS tagger?
 - Overall error rate with respect to a gold-standard test set.
 - Error rates on particular tags
 - Error rates on particular words
 - Tag confusions, ...

Error Analysis

- Look at a confusion matrix (contingency table)



	IN	JJ	NN	NNP	RB	VBD	VBN
IN	—	.2			.7		
JJ	.2	—			1.7	.2	
NN			8.7				.2
NNP	.2	3.3		4.1			
RB	2.2	2.0	.5				
VBD		.3	.5				
VBN			2.8				

- See what errors are causing problems
 - Noun (NN, hypothesized) vs ProperNoun (NNP, correct) vs Adj (JJ, correct)
 - Preterite (VBD, hypothesized) vs Participle (VBN, correct) vs Adjective (JJ, correct)

Evaluation

- The result is compared with a manually coded “Gold Standard”
 - Typically accuracy reaches 96-97%
 - This may be compared with result for a baseline tagger
- Important: 100% is impossible even for human annotators.

Part-of-Speech Tagging from 97% to 100% (Manning, 2011)

Class	Frequency
1. Lexicon gap	4.5%
2. Unknown word	4.5%
3. Could plausibly get right	16.0%
4. Difficult linguistics	19.5%
5. Underspecified/unclear	12.0%
6. Inconsistent/no standard	28.0%
7. Gold standard wrong	15.5%

5. it is unclear whether *discontinued* should be regarded as an adjective or verbal participle.

it will take a \$ 10 million fourth-quarter charge against/IN discontinued/JJ operations/NNS

6. The 1930s, the treebank is inconsistent in sometimes tagging them as CD and at other times as NNS.

Orson Welles 's Mercury Theater in/IN the/DT '30s/NNS ./.

1. *slash* is clearly a noun, but in the training set, it occurs only but several times as a verb

a/DT 60/CD %/NN slash/NN in/IN the common stock dividend

2. *substandard* is a word which does not appear in the training data

blaming the disaster on/IN substandard/JJ construction/NN

3. *overnight* is here functioning as an adverb rather than an adjective (the tag it chose), since it is here a verb modifier not pre-modifying a noun

market/NN players/NNS overnight/RB in/IN Tokyo/NNP began bidding up oil prices

4. a tagger just cannot correctly choose between the present (VBP) and past (VBD) tag for set without an understanding of a multisentence discourse context

They/PRP set/VBP up/RP absurd/JJ situations/NNS , detached from reality

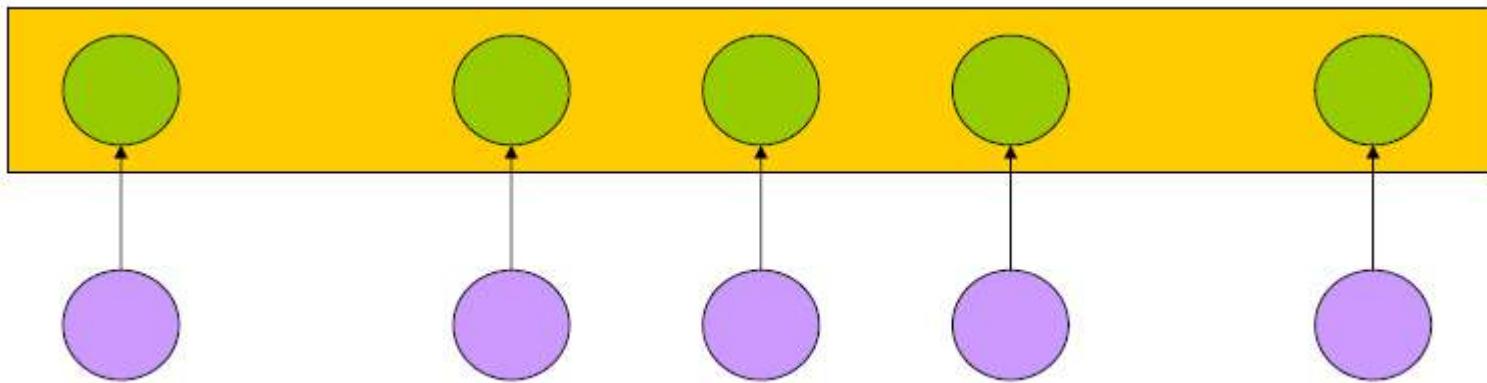
Unknown Words

- Simplest model
 - Unknown words can be of any part of speech
 - Or only any open class part of speech, i.e., nouns, verbs, and so on
- Morphological and other cues
 - *-ed*: past tense forms or past participles

Maximum Entropy Models

- MaxEnt: multinomial logistic regression
- sequence classification/sequence labeling
 - Assign a label to each element in a sequence
 - Application: assign a part-of-speech tag to a word
 - Maximum Entropy Markov Model (MEMM):
a common MaxEnt classifier

What is MaxEnt?



- Green nodes are *states*
- Purple nodes are *observations*
- States are mutually independent

Max-Ent

- Exponential/log-linear classifiers
 - Extract some set of features from the input
 - Combine them linearly, i.e., each feature is multiplied a weight and then added up
 - Use the sum as an exponent

$$p(c|x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i\right)$$

POS classification

- An example

Secretariat/NNP is/BEZ expected/VBN to/TO
race/?? tomorrow/

$$f_1(c, x) = \begin{cases} 1 & \text{if } word_i = \text{"race"} \text{ & } c = \text{NN} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(c, x) = \begin{cases} 1 & \text{if } t_{i-1} = \text{TO} \text{ & } c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(c, x) = \begin{cases} 1 & \text{if } \text{suffix}(word_i) = \text{"ing"} \text{ & } c = \text{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$f_1(c, x) = \begin{cases} 1 & \text{if } word_i = \text{"race"} \& c = \text{NN} \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(c, x) = \begin{cases} 1 & \text{if } t_{i-1} = \text{TO} \& c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(c, x) = \begin{cases} 1 & \text{if } \text{suffix}(word_i) = \text{"ing"} \& c = \text{VBG} \\ 0 & \text{otherwise} \end{cases}$$

$$f_4(c, x) = \begin{cases} 1 & \text{if } \text{is_lower_case}(word_i) \& c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$f_5(c, x) = \begin{cases} 1 & \text{if } word_i = \text{"race"} \& c = \text{VB} \\ 0 & \text{otherwise} \end{cases}$$

$$f_6(c, x) = \begin{cases} 1 & \text{if } t_{i-1} = \text{TO} \& c = \text{NN} \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|x)$$

		f1	f2	f3	f4	f5	f6
VB	f	0	1	0	1	1	0
VB	w		.8		.01	.1	
NN	f	1	0	0	0	0	1
NN	w	.8					-1.3

Secretariat/NNP is/BEZ
expected/VBN to/TO race/??
tomorrow/

$$p(c|x) = \frac{\exp\left(\sum_{i=0}^N w_{ci} f_i(c, x)\right)}{\sum_{c' \in C} \exp\left(\sum_{i=0}^N w_{c'i} f_i(c', x)\right)}$$

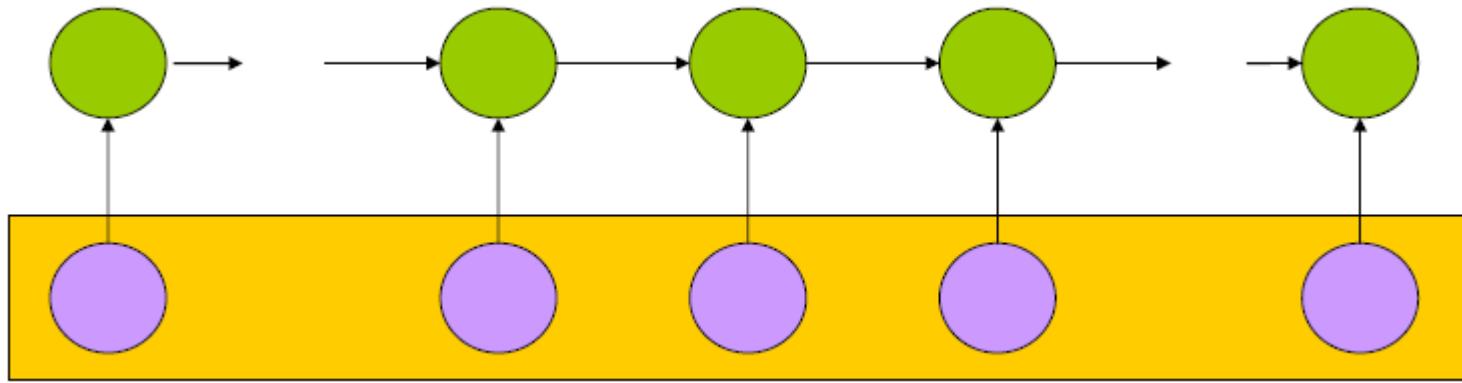
$$P(NN|x) = \frac{e^{.8} e^{-1.3}}{e^{.8} e^{-1.3} + e^{.8} e^{.01} e^{.1}} = .20$$

$$P(VB|x) = \frac{e^{.8} e^{.01} e^{.1}}{e^{.8} e^{-1.3} + e^{.8} e^{.01} e^{.1}} = .80$$

Maximum Entropy Markov Model (MEMM)

- The basic MaxEnt model classifies a single observation into one of a set of discrete classes
- MEMM: an augmentation of the basic MaxEnt classifier so that it can be applied to assign a class to each element in a sequence

What is MEMM?



- Purple nodes are *observations*
- Observations (features of observations)
determine states

POS tagging from MaxExt to MEMM

- How to turn a single local classifier into a general sequence classifier
 - features from the current word
 - features from surrounding words
 - the output of the classifier from previous words
- The simplest approach
 - run the local classifier left-to-right
 - when classifying each word, we can rely on the output of the classifier from the previous word as a feature.

- Flaw of the simplest approach
 - It makes a hard decision on each word before moving on to the next word
 - The classifier cannot use information from the later words to inform its earlier decisions

MEMM

- Compute the posterior $P(T|W)$ directly

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T \prod_i P(t_i|w_i, t_{i-1})\end{aligned}$$

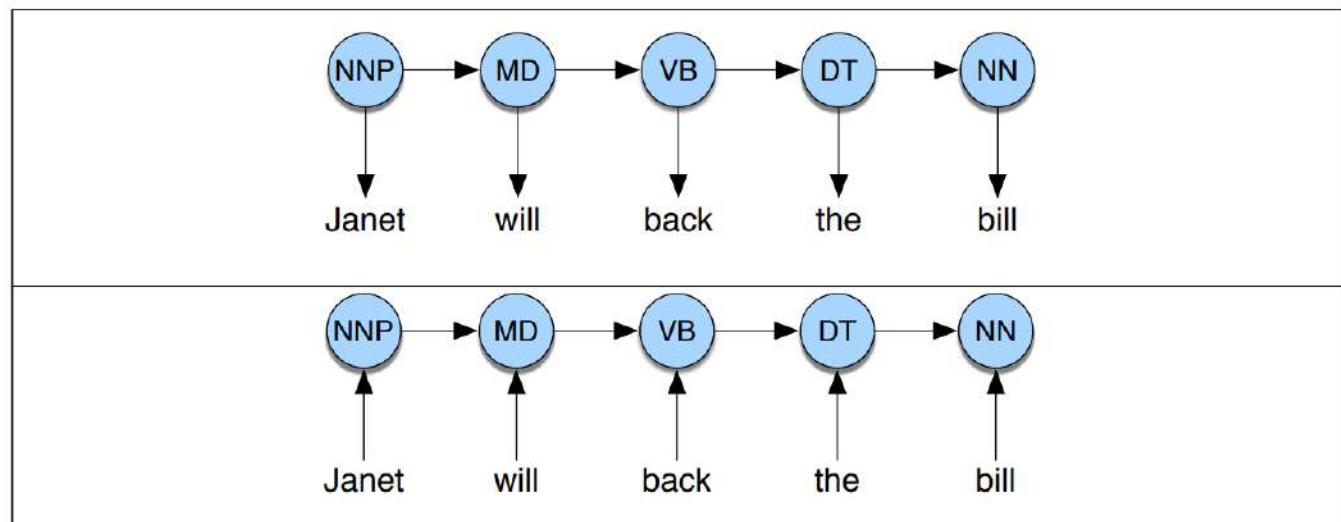
- **discriminative model:** train the model directly to discriminate among the possible tag sequences

HMM vs. MEMM

- Each arc would be associated with a probability

two separate probabilities

a single probability function



HMM vs. MEMM

- Unlike the HMM, the MEMM can condition on any useful feature of the input observation.
 - HMM computes the probability of the state sequence given the observations as

$$P(Q|O) = \prod_{i=1}^n P(o_i|q_i) \times \prod_{i=1}^n P(q_i|q_{i-1})$$

- MEMM computes the probability of the state sequence given the observations as

$$P(Q|O) = \prod_{i=1}^n P(q_i|q_{i-1}, o_i)$$

HMM vs. MEMM

- In general, an MEMM can condition on many more features than HMM
- To estimate the individual probability of a transition from a state q' to a state q producing an observation

$$P(q|q', o) = \frac{1}{Z(o, q')} \exp\left(\sum_i w_i f_i(o, q)\right)$$

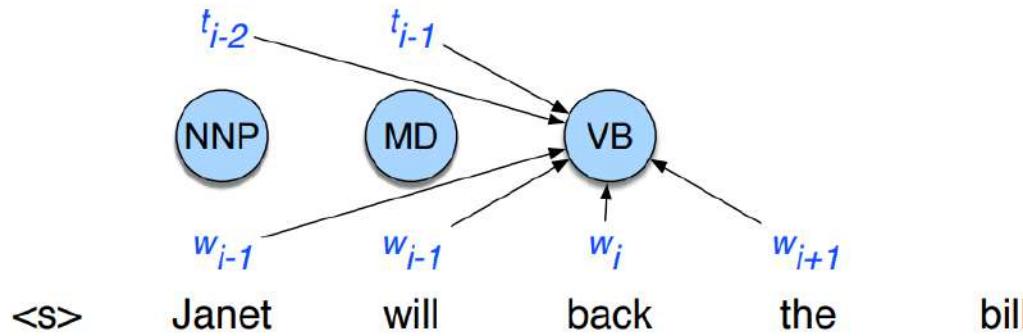


Figure 8.13 An MEMM for part-of-speech tagging showing the ability to condition on more features.

Feature Template

$$\begin{aligned} &\langle t_i, w_{i-2} \rangle, \langle t_i, w_{i-1} \rangle, \langle t_i, w_i \rangle, \langle t_i, w_{i+1} \rangle, \langle t_i, w_{i+2} \rangle \\ &\quad \langle t_i, t_{i-1} \rangle, \langle t_i, t_{i-2}, t_{i-1} \rangle, \\ &\quad \langle t_i, t_{i-1}, w_i \rangle, \langle t_i, w_{i-1}, w_i \rangle \langle t_i, w_i, w_{i+1} \rangle, \end{aligned}$$

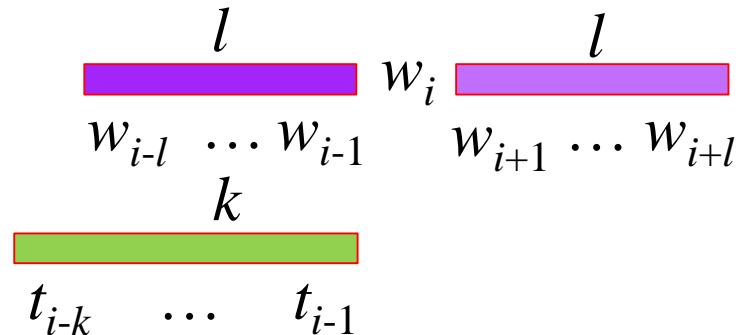
- $t_i = \text{VB}$ and $w_{i-2} = \text{Janet}$
- $t_i = \text{VB}$ and $w_{i-1} = \text{will}$
- $t_i = \text{VB}$ and $w_i = \text{back}$
- $t_i = \text{VB}$ and $w_{i+1} = \text{the}$
- $t_i = \text{VB}$ and $w_{i+2} = \text{bill}$
- $t_i = \text{VB}$ and $t_{i-1} = \text{MD}$
- $t_i = \text{VB}$ and $t_{i-1} = \text{MD}$ and $t_{i-2} = \text{NNP}$
- $t_i = \text{VB}$ and $w_i = \text{back}$ and $w_{i+1} = \text{the}$

- w_i contains a particular prefix (from all prefixes of length ≤ 4)
- w_i contains a particular suffix (from all suffixes of length ≤ 4)
- w_i contains a number
- w_i contains an upper-case letter
- w_i contains a hyphen
- w_i is all upper case
- w_i 's word shape
- w_i 's short word shape
- w_i is upper case and has a digit and a dash (like CFC-12)
- w_i is upper case and followed within 3 words by Co., Inc., etc.

Decoding and Training MEMMs

$$\begin{aligned}\hat{T} &= \underset{T}{\operatorname{argmax}} P(T|W) \\ &= \underset{T}{\operatorname{argmax}} \prod_i P(t_i | w_{i-l}^{i+l}, t_{i-k}^{i-1})\end{aligned}$$

$$= \underset{T}{\operatorname{argmax}} \prod_i \frac{\exp \left(\sum_j \theta_j f_j(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1}) \right)}{\sum_{t' \in \text{tagset}} \exp \left(\sum_j \theta_j f_j(t', w_{i-l}^{i+l}, t_{i-k}^{i-1}) \right)}$$



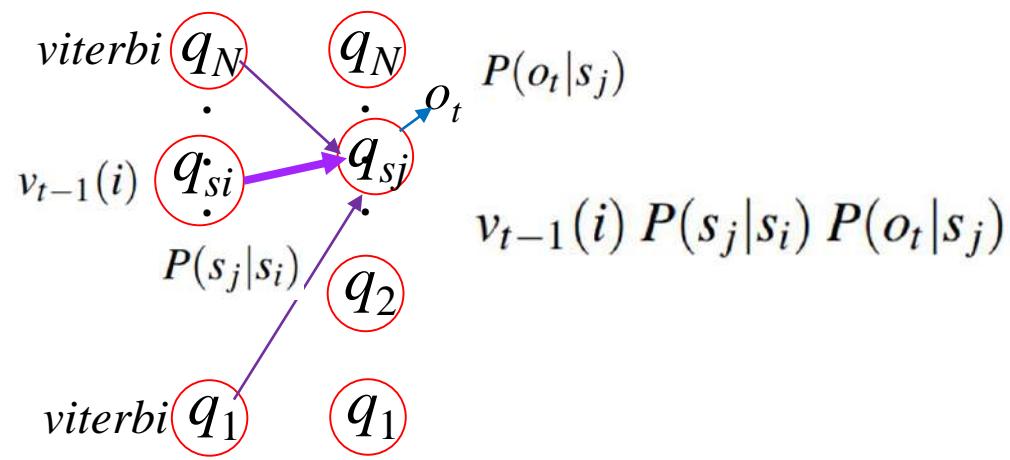
Decoding in MEMM

- Replace the Viterbi equation to compute the Viterbi value of time t for state j from

$$v_t(j) = \max_{i=1}^N v_{t-1}(i) P(s_j|s_i) P(o_t|s_j) \quad 1 \leq j \leq N, 1 < t \leq T$$

to

$$v_t(j) = \max_{i=1}^N v_{t-1}(i) P(s_j|s_i, o_t) \quad 1 \leq j \leq N, 1 < t \leq T$$

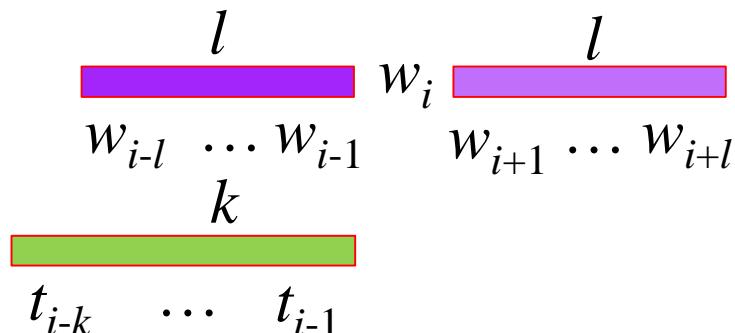


Greedy Decoding Algorithm

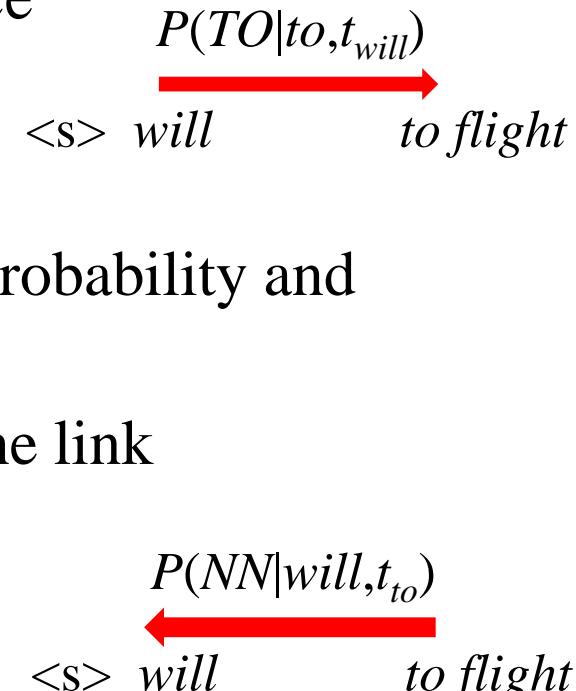
- Build a local classifier that classifies each word left to right, making a hard classification of the first word in the sentence, then a hard decision on the second word, and so on.

```
function GREEDY SEQUENCE DECODING(words W, model P) returns tag sequence T
```

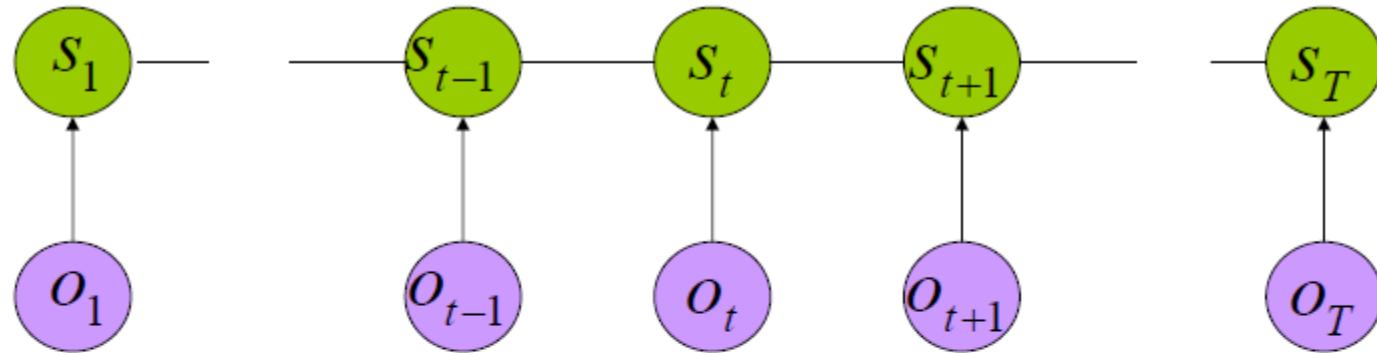
```
for  $i = 1$  to  $\text{length}(W)$   
     $\hat{t}_i = \underset{t' \in T}{\text{argmax}} P(t' | w_{i-l}^{i+l}, t_{i-k}^{i-1})$ 
```



Bidirectionality

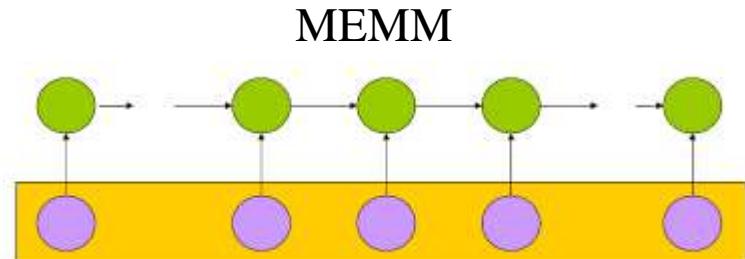
- MEMM and HMM are exclusively run left-to-right.
 - Label bias or observation bias problem
 - Situations when one source of information is ignored because it is explained away by another source
 - $\langle s \rangle \text{ will/NN to/TO fight/VB}$
 - $P(TO|to, t_{will})$ is so close to 1 regardless of t_{will}
 - the model cannot make use of the transition probability and incorrectly chooses *MD*
 - Bidirectionality helps the model by making the link between *TO* available when tagging the *NN*
 - conditional random field or CRF
- 

CRF Formalism



- $s: \{1, 2, \dots, N\}$ are values of states
- $o: \{1, 2, \dots, M\}$ are values of observations

$$P(S | O)$$



CRF Formalism

$$S \leftarrow Y, O \leftarrow X$$

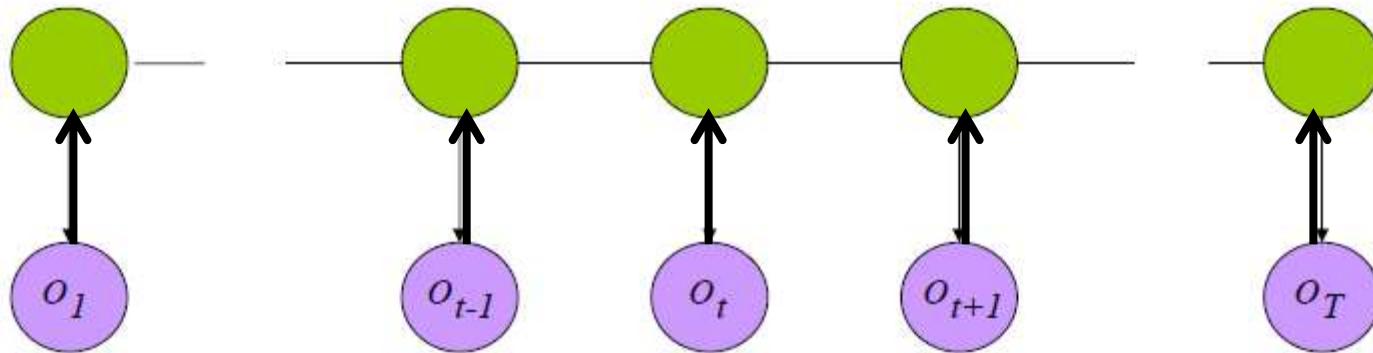
$$P(S | O) = P(Y | X)$$

$$P(Y | X) = \frac{\exp\left(\sum_k \lambda_k \cdot F_k(Y, X)\right)}{Z(X)}$$

$$Z(X) = \sum_Y \exp\left(\sum_k \lambda_k \cdot F_k(Y, X)\right)$$

$$F(Y, X) = \sum_{t=1}^T f(y_t, x_t)$$

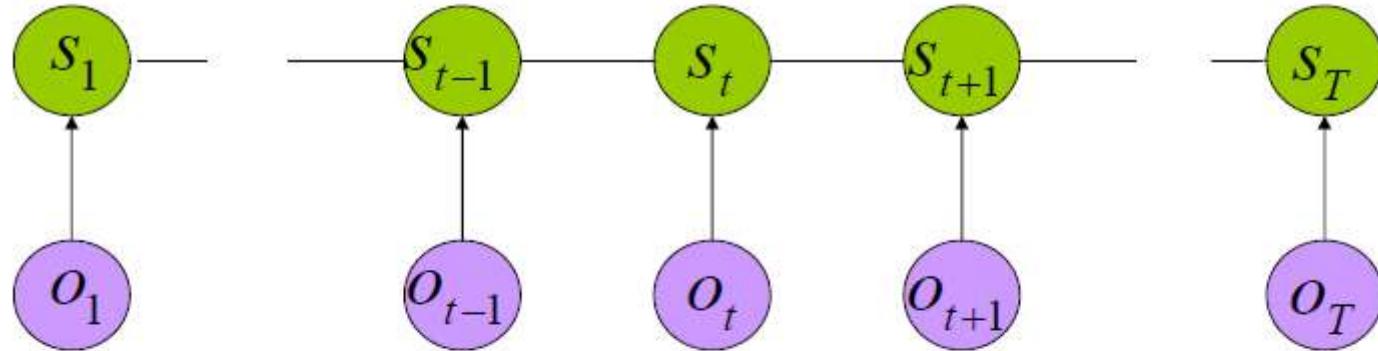
Tagging



- Viterbi algorithm

$$\arg \max_Y P(Y | X) = \arg \max_Y \log \frac{\exp \sum_k \lambda_k \cdot F_k(Y, X)}{Z(X)}$$
$$= \arg \max_Y \sum_k \lambda_k \cdot F_k(Y, X)$$

Learning



$(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$

$$P(Y | X) = \frac{\exp\left(\sum_k \lambda_k \cdot F_k(Y, X)\right)}{Z(X)}$$

$$Z(X) = \sum_Y \exp\left(\sum_k \lambda_k \cdot F_k(Y, X)\right)$$

$$\arg \max \sum_{i=1}^n \log \frac{\exp\left(\sum_k \lambda_k \cdot F_k(Y, X)\right)}{Z(X)}$$

Maximum Likelihood

- Turn a sequence model into a bidirectional model by using multiple passes, for example,
 - The tagger can be run twice, once left-to-right and once right-to-left.
 - In greedy decoding, for each word the classifier chooses the highest-scoring of the tag assigned by the left-to-right and right-to-left classifier.
 - In Viterbi decoding, the classifier chooses the higher scoring of the two sequences (left-to-right or right-to-left).
- bi-LSTM model

Tagging with bi-LSTMs

- An RNN is a function that reads in n vectors x_1, \dots, x_n and produces an output vector h_n , that depends on the entire sequence x_1, \dots, x_n .
- The vector h_n is then fed as an input to some classifier, or higher-level RNNs in stacked/hierarchical models.
- sequence bi-RNN

$$v = \text{bi-RNN}_{\text{seq}}(x_{1:n}) = \text{RNN}_f(x_{1:n}) \circ \text{RNN}_r(x_{n:1})$$

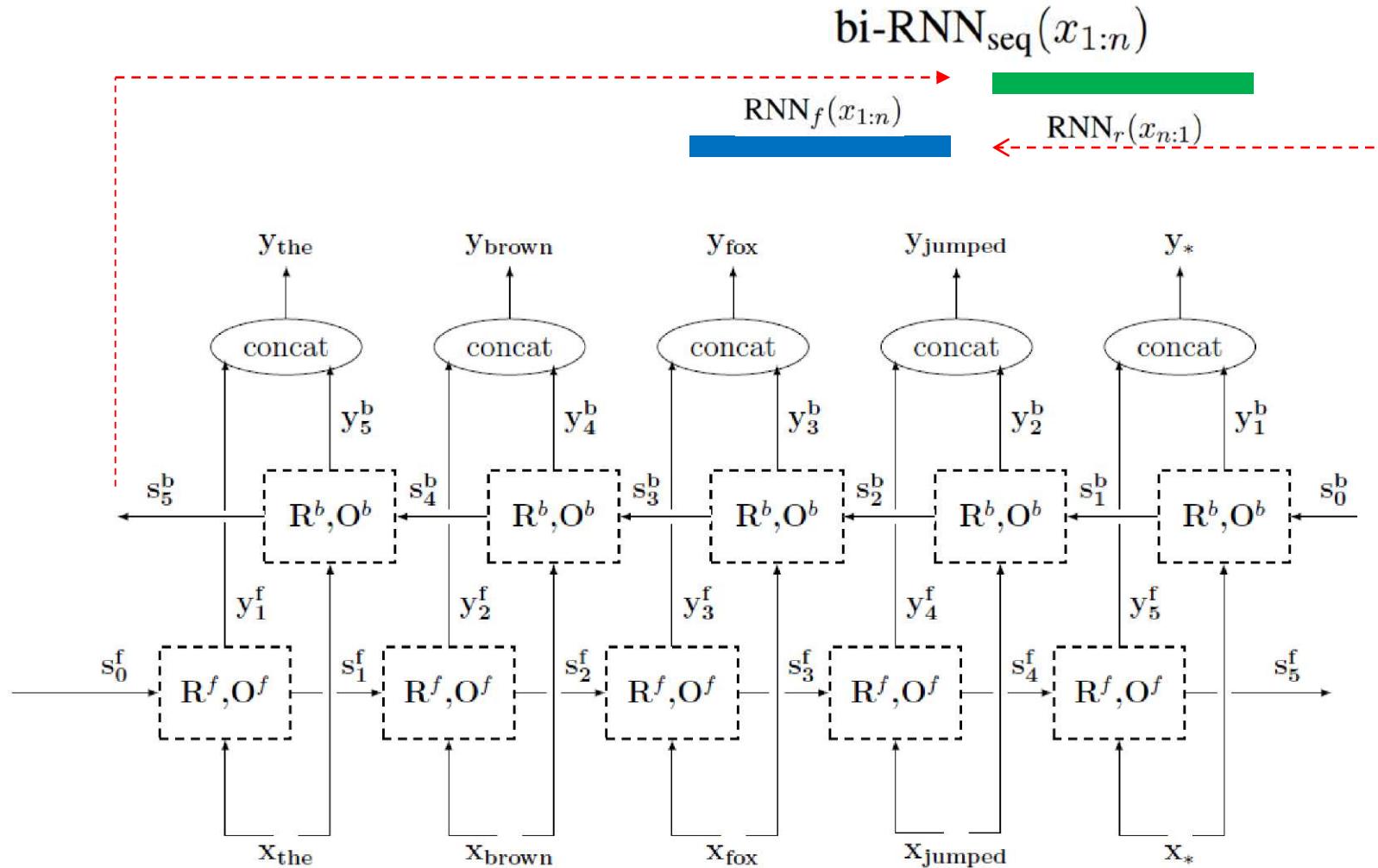
- context bi-RNN

$$v_i = \text{bi-RNN}_{\text{ctx}}(x_{1:n}, i) = \text{RNN}_f(x_{1:i}) \circ \text{RNN}_r(x_{n:i})$$

Computing the biRNN for the sentence “the brown fox jumped .”

sequence bi-RNN

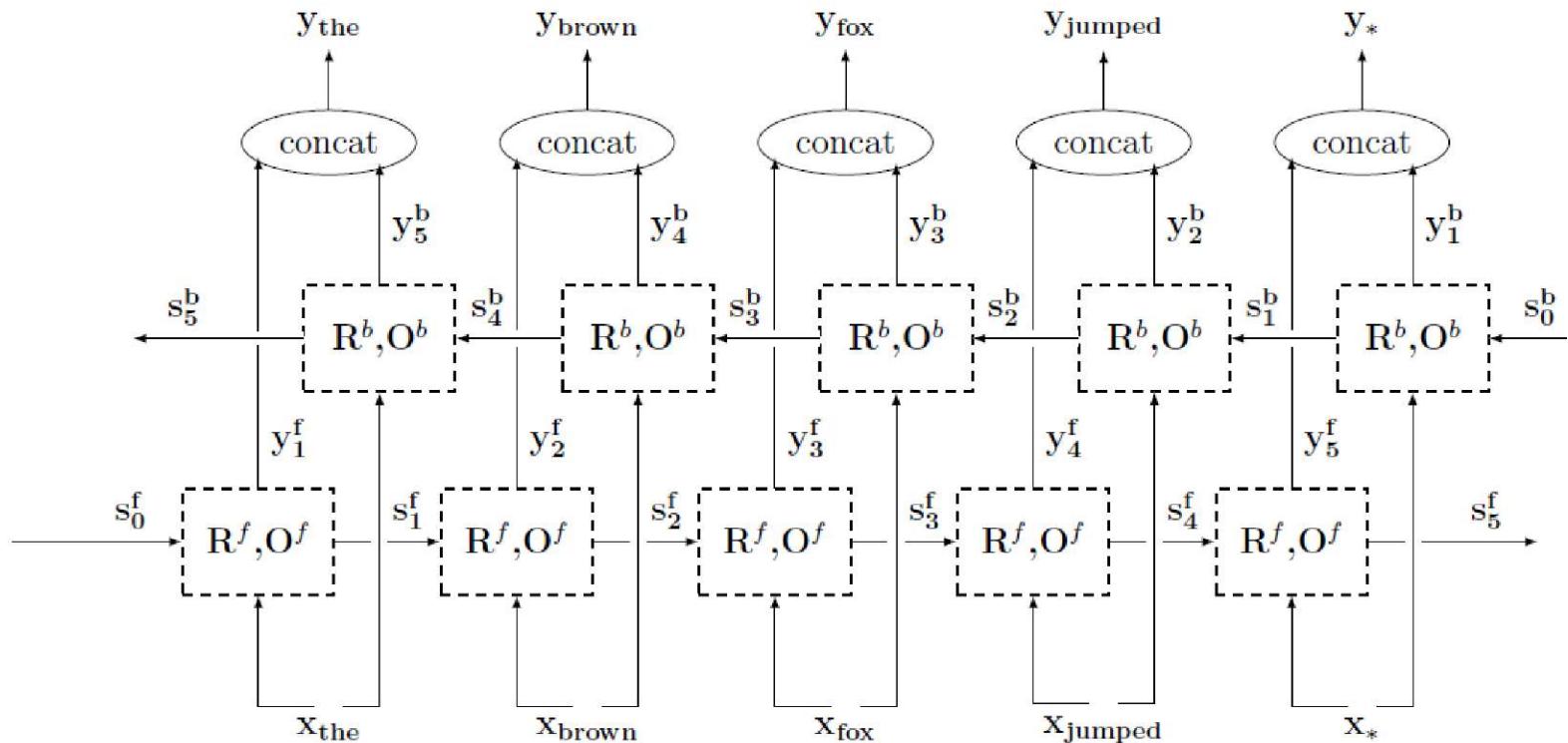
$$v = \text{bi-RNN}_{\text{seq}}(x_{1:n}) = \text{RNN}_f(x_{1:n}) \circ \text{RNN}_r(x_{n:1})$$



Computing the biRNN for the sentence “the brown fox jumped .”

context bi-RNN

$$v_i = \text{bi-RNN}_{\text{ctx}}(x_{1:n}, i) = \text{RNN}_f(x_{1:i}) \circ \text{RNN}_r(x_{n:i})$$



multi-task learning

tags of the sequence

the log frequency
of the next token

cross-entropy loss

$$L(\hat{y}_t, y_t) + L(\hat{y}_a, y_a)$$

t: POS tag

a: log frequency label

$$a = \text{int}(\log(freq_{train}(w)))$$

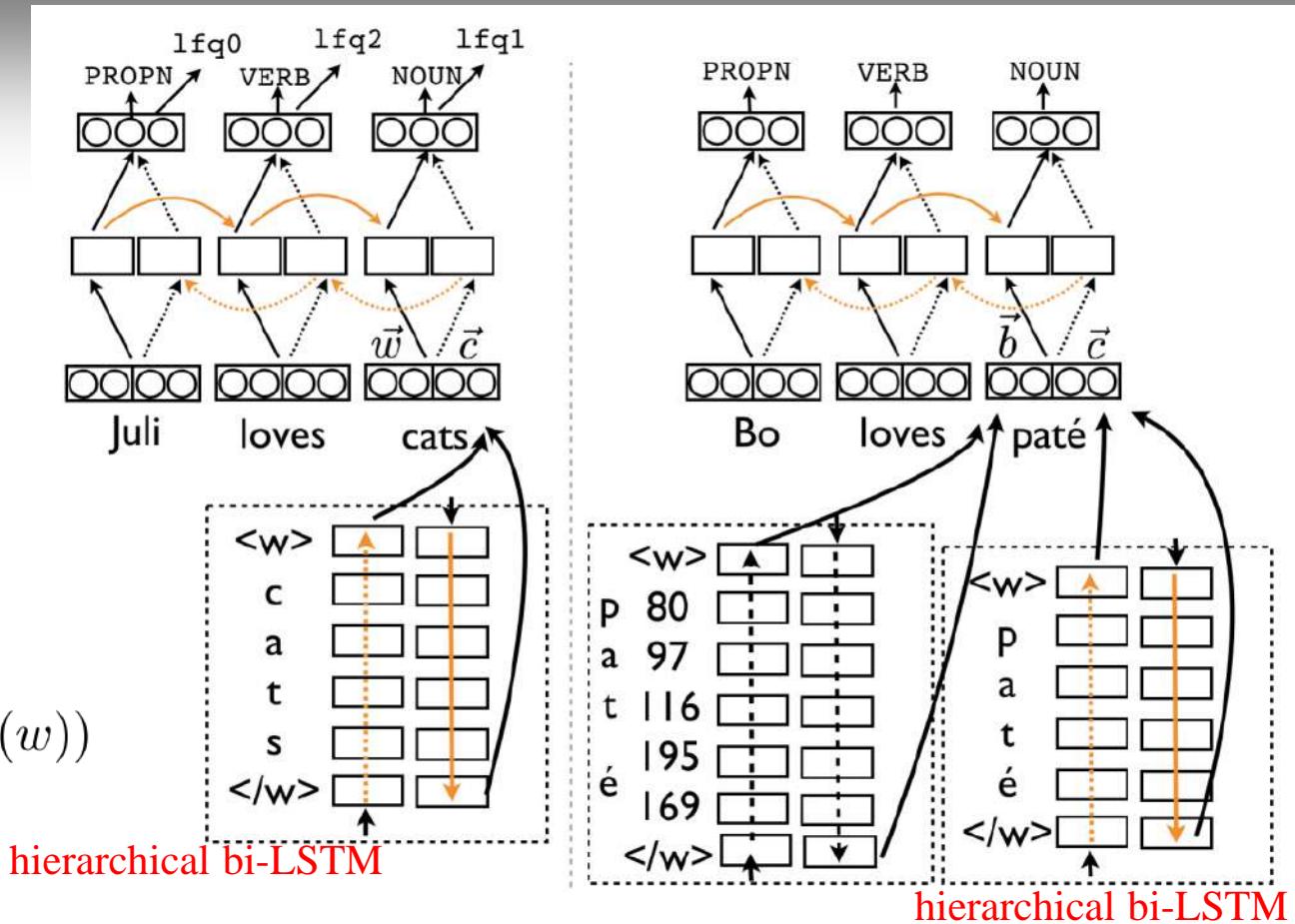


Figure 1: Right: bi-LSTM, illustrated with $\vec{b} + \vec{c}$ (bytes and characters), for $\vec{w} + \vec{c}$ replace \vec{b} with words \vec{w} . Left: FREQBIN, our multi-task bi-LSTM that predicts at every time step the tag and the frequency class for the next token.

TNT:
a second order HMM
with suffix trie
handling for OOVs

The overall best system is
the multi-task bi-
LSTM FREQBIN
(it uses $\sim w + \sim c$ and
POLYGLOT
initialization for $\sim w$)

Polyglot language models:
neural network language
models that are trained on
and applied to any
number of languages

	BASELINES		BI-LSTM using:				$\vec{w} + \vec{c}$ + POLYGLOT		OOV ACC		BTS
	TNT	CRF	\vec{w}	\vec{c}	$\vec{c} + \vec{b}$	$\vec{w} + \vec{c}$	bi-LSTM	FREQBIN	bi-LSTM	FREQBIN	
avg	94.61	94.27	92.37	94.29	94.01	96.08†	96.50	96.50	87.80	87.98	95.70
Indoeur.	94.70	94.58	92.72	94.58	94.28	96.24†	96.63	96.61	87.47	87.63	—
non-Indo.	94.57	93.62	91.97	93.51	93.16	95.70†	96.21	96.28	90.26	90.39	—
Germanic	93.27	93.21	91.18	92.89	92.59	94.97†	95.55	95.49	85.58	85.45	—
Romance	95.37	95.53	94.71	94.76	94.49	95.63†	96.93	96.93	85.84	86.07	—
Slavic	95.64	94.96	91.79	96.45	96.26	97.23†	97.42	97.43	91.48	91.69	—
ar	97.82	97.56	95.48	98.68	98.43	98.89	98.87	98.91	95.90	96.21	—
bg	96.84	96.36	95.12	97.89	97.78	98.25	98.23	90.06	90.06	90.56	97.84
cs	96.82	96.56	93.77	96.38	96.08	97.93	98.02	97.89	91.65	91.30	98.50
da	94.29	93.83	91.96	95.12	94.88	95.94	96.16	96.35	86.13	86.35	95.52
de	92.64	91.38	90.33	90.02	90.11	93.11	93.51	93.38	85.37	86.77	92.87
en	92.66	93.35	92.10	91.62	91.57	94.61	95.17	95.16	80.28	80.11	93.87
es	94.55	94.23	93.60	93.06	92.29	95.34	95.67	95.74	79.26	79.27	95.80
eu	93.35	91.63	88.00	92.48	92.72	94.91	95.38	95.51	83.55	84.30	—
fa	95.98	95.65	95.31	95.82	95.03	96.89	97.60	97.49	88.82	89.05	96.82
fi	93.59	90.32	87.95	90.25	89.15	95.18	95.74	95.85	88.35	88.85	95.48
fr	94.51	95.14	94.44	94.39	93.69	96.04	96.20	96.11	82.79	83.54	95.75
he	93.71	93.63	93.97	93.74	93.58	95.92	96.92	96.96	88.75	88.83	—
hi	94.53	96.00	95.99	93.40	92.99	96.64	96.97	97.10	83.98	85.27	—
hr	94.06	93.16	89.24	95.32	94.47	95.59	96.27	96.82	90.50	92.71	—
id	93.16	92.96	90.48	91.37	91.46	92.79	93.32	93.41	88.03	87.67	92.85
it	96.16	96.43	96.57	95.62	95.77	97.64	97.90	97.95	89.15	89.15	97.56
nl	88.54	90.03	84.96	89.11	87.74	92.07	92.82	93.30	78.61	75.95	—
no	96.31	96.21	94.39	95.87	95.75	97.77	98.06	98.03	93.56	93.75	—
pl	95.57	93.96	89.73	95.80	96.19	96.62	97.63	97.62	95.00	94.94	—
pt	96.27	96.32	94.24	95.96	96.2	97.48	97.94	97.90	92.16	92.33	—
sl	94.92	94.77	91.09	96.87	96.77	97.78	96.97	96.84	90.19	88.94	—
sv	95.19	94.45	93.32	95.57	95.5	96.30	96.60	96.69	89.53	89.80	95.57

Table 2: Tagging accuracies on UD 1.2 test sets. \vec{w} : words, \vec{c} : characters, \vec{b} : bytes. Bold/†: best accuracy/representation; +POLYGLOT: using pre-trained embeddings. FREQBIN: our multi-task model. OOV ACC: accuracies on OOVs. BTS: best results in Gillick et al. (2016) (not strictly comparable).

Sequential Labelling

- Penn TreeBank (PTB) POS tagging

Conv-CRF (Collobert et al., 2011)	97.29
LSTM	97.29
BI-LSTM	97.40
CRF	97.45
LSTM-CRF	97.54
BI-LSTM-CRF	97.55

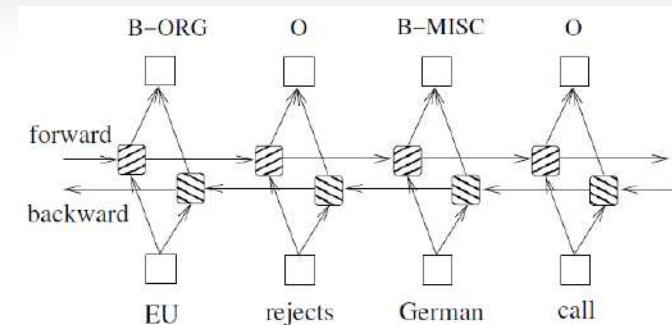
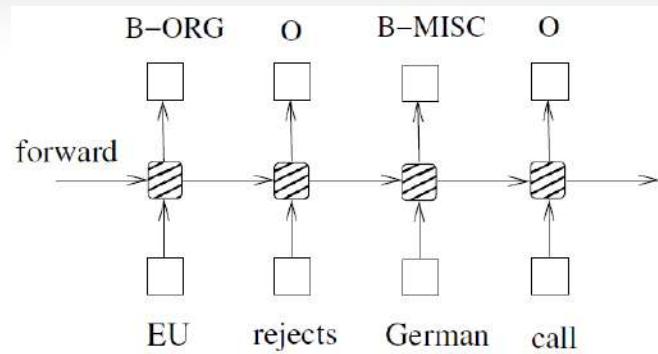
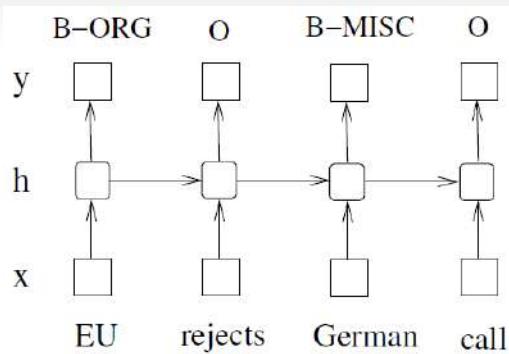


Figure 1: A simple RNN model.

Figure 3: A LSTM network.

Figure 4: A bidirectional LSTM network.

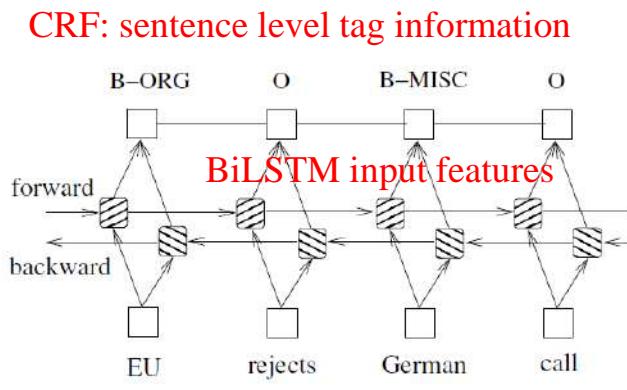
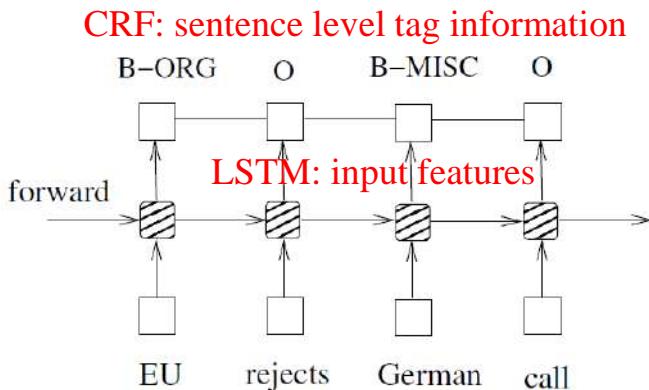
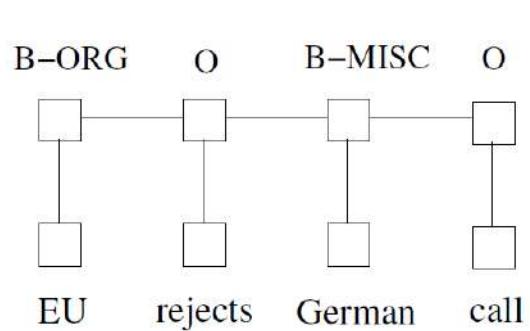


Figure 5: A CRF network.

Figure 6: An LSTM-CRF model.

Figure 7: A BI-LSTM-CRF model.

Tasks

- Penn TreeBank (PTB) POS tagging
 - Each word is assigned with a unique tag
- CoNLL 2000 chunking
 - Each word is tagged with its phrase type
 - B-NP: a word starting a noun phrase
- CoNLL 2003 named entity tagging
 - Each word is tagged with other or one of four entity types: Person, Location, Organization, or Miscellaneous.

Spelling Features

- whether start with a capital letter
- whether has all capital letters
- whether has all lower case letters
- whether has non initial capital letters
- whether mix with letters and digits
- whether has punctuation
- letter prefixes and suffixes (with window size of 2 to 5)
- whether has apostrophe end ('s)
- letters only, for example, I. B. M. to IBM
- non-letters only, for example, A. T. &T. to ..&
- word pattern feature, with capital letters, lower case letters, and digits

Features

- Context features
 - unigram features and bi-grams features
- Word embedding
 - 50-dimensional embedding vector

Features Connection Tricks

- Two alternatives
 - Input both word, spelling and context features to the network directly

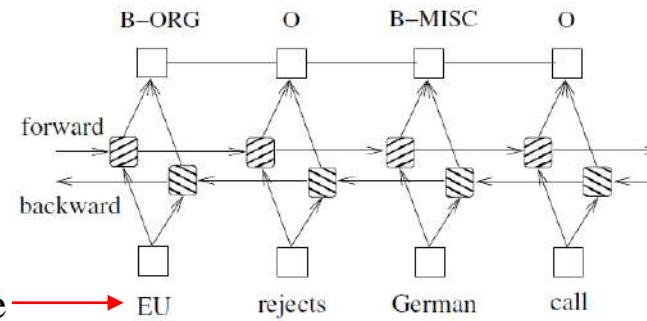
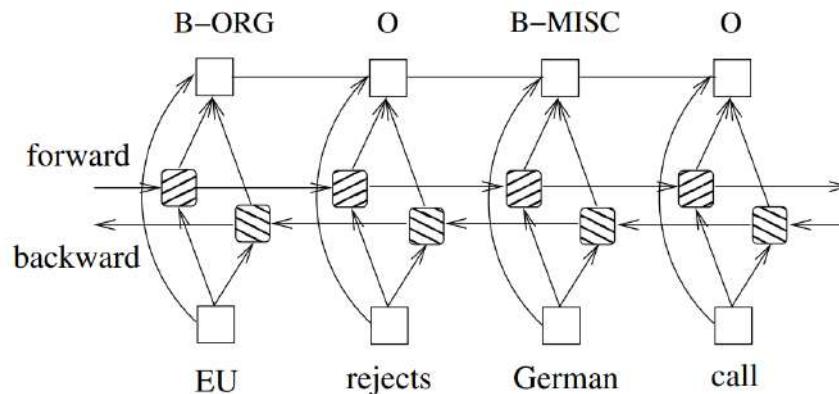


Figure 7: A BI-LSTM-CRF model.

- Connect features to outputs of networks directly



Experimental Results

Two ways to initialize word embedding

Chunking

NE tagging

		POS	CoNLL2000	CoNLL2003
Random	Conv-CRF (Collobert et al., 2011)	96.37	90.33	81.47
	LSTM	97.10	92.88	79.82
	BI-LSTM	97.30	93.64	81.11
	CRF	97.30	93.69	83.02
	LSTM-CRF	97.45	93.80	84.10
	BI-LSTM-CRF	97.43	94.13	84.26
Senna	Conv-CRF (Collobert et al., 2011)	97.29	94.32	88.67 (89.59)
	LSTM	97.29	92.99	83.74
	BI-LSTM	97.40	93.92	85.17
	CRF	97.45	93.83	86.13
	LSTM-CRF	97.54	94.27	88.36
	BI-LSTM-CRF	97.55	94.46	88.83 (90.10)

Tagging performance on POS, chunking and NER tasks with only word features.

		POS	CoNLL2000	CoNLL2003
Senna	LSTM	94.63 (-2.66)	90.11 (-2.88)	75.31 (-8.43)
	BI-LSTM	96.04 (-1.36)	93.80 (-0.12)	83.52 (-1.65)
	CRF	94.23 (-3.22)	85.34 (-8.49)	77.41 (-8.72)
	LSTM-CRF	95.62 (-1.92)	93.13 (-1.14)	81.45 (-6.91)
	BI-LSTM-CRF	96.11 (-1.44)	94.40 (-0.06)	84.74 (-4.09)

Tweets contain lot of **OOV words** compared to standard text

Each word from the input sequence is represented by a combination of two vectors of features, **character-level** and **word-level** embedding.

A word: a $v \times l$ dimensional matrix

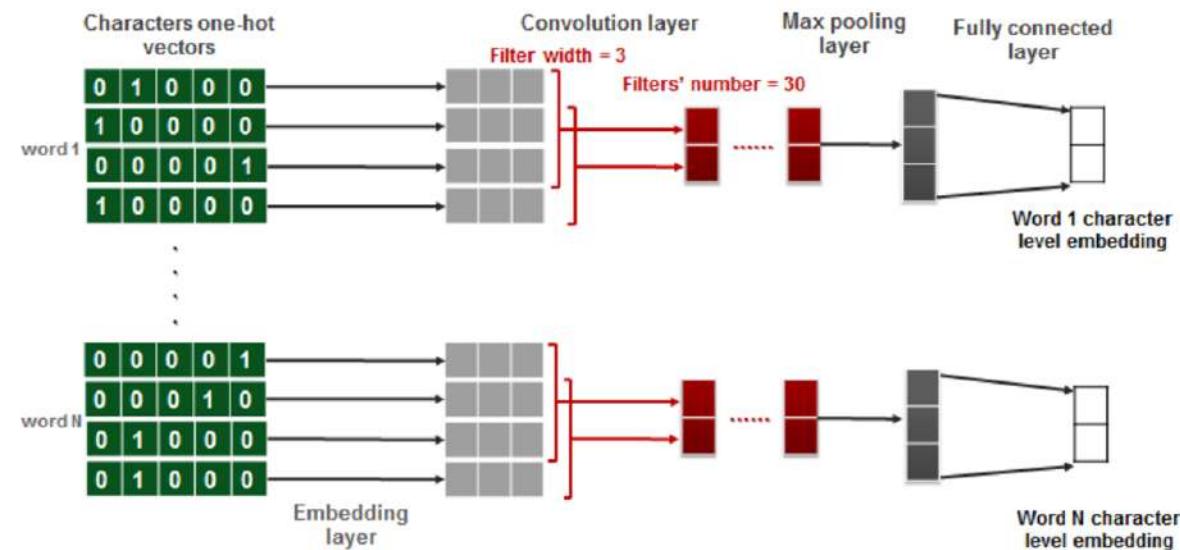
Word embedding: a $d \times l$ dimensional matrix

v : character's vocabulary size

l : maximal length of words

d : character embedding's dimension

Convolutional Neural Network architecture for character-level embedding



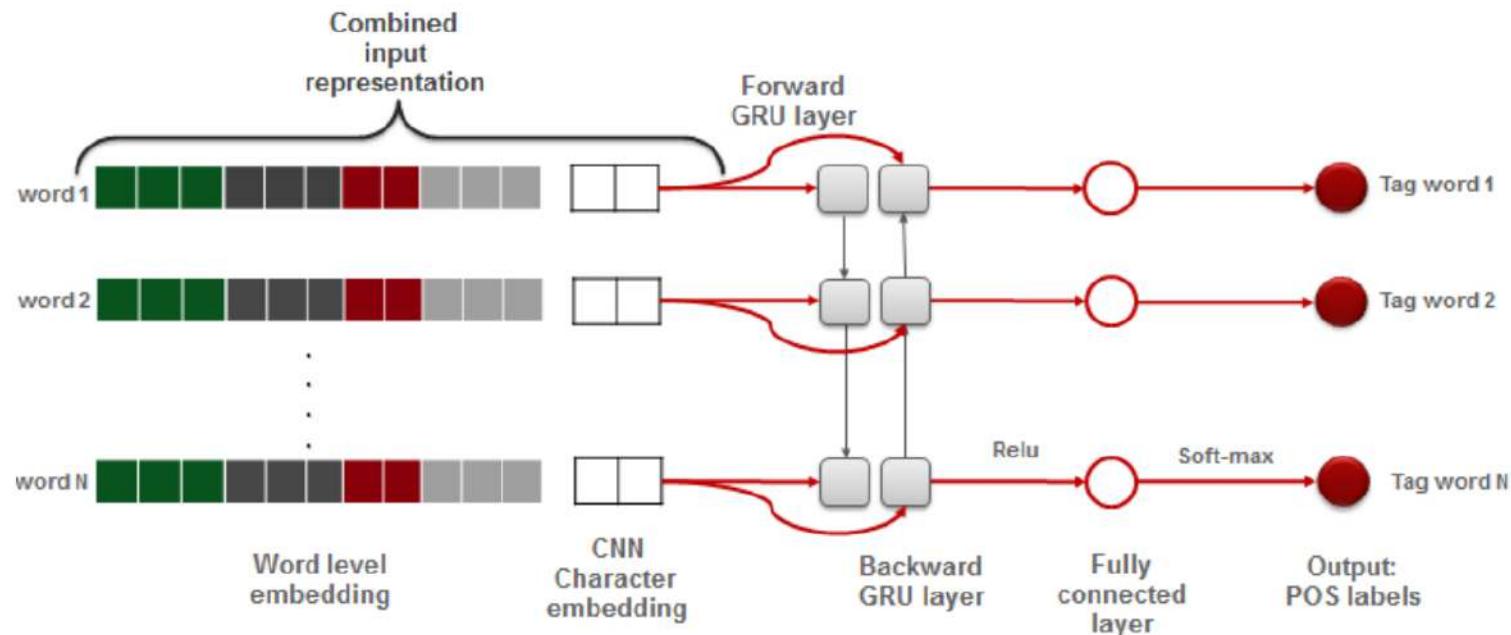


Figure 2: Overall system design. First, the system embeds each word of the current sentence into two representations: character level representation using a CNN network and a word level representation by combining different pre-trained models. Then, the two representations are combined and fed into a bidirectional GRU layer, the resulting vector is fed to a fully connected layer and finally a softmax layer to perform POS tagging.

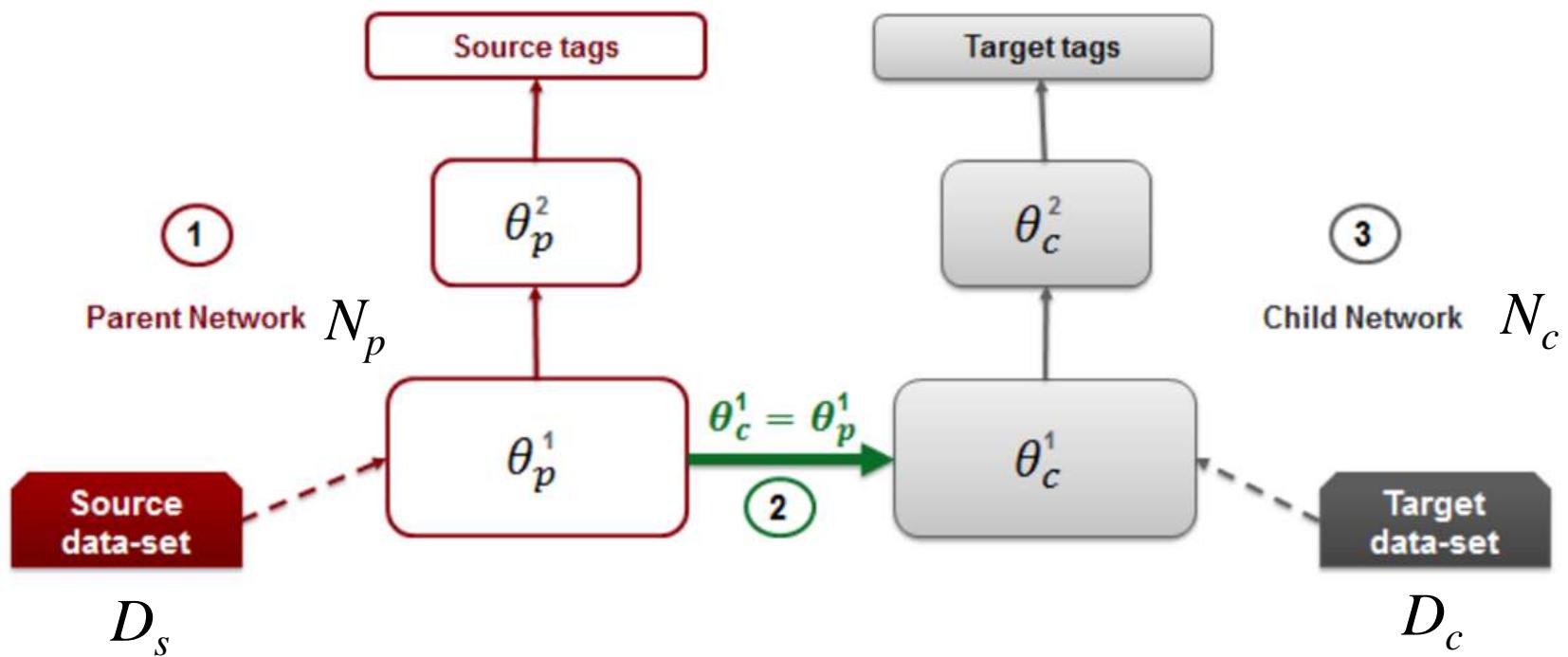
Transfer Learning Approach

- Learning a parent neural network on a source problem with enough data, then transferring a part of its weights to represent data of a target problem with few training examples.
- Cross-Domain Transfer
- Cross-Task Transfer

Cross-Domain Transfer

- Knowledge is transferred from a source domain to a target domain
- The source domain is the **standard form (well-established)** of a language
- The target domain is **the social media text of the same language**
- The source and the target problems are trained for the same task (POS tagging)
- A parent neural network N_p with a set of parameters θ_p splitted into two sets:
$$\theta_p = (\theta_p^1, \theta_p^2)$$
- A child network N_c with a set of parameters θ_c splitted into two sets:
$$\theta_c = (\theta_c^1, \theta_c^2).$$

- (1) Learn the parent network on annotated data from the source problem on a source dataset D_s .
- (2) Transfer weights of the first set of parameters of the parent network N_p to the child network N_c . $N_c: \theta_c^1 = \theta_p^1$
- (3) Fine-tune the child network to the target problem by training it on the target dataset D_c .



Cross-Task Transfer

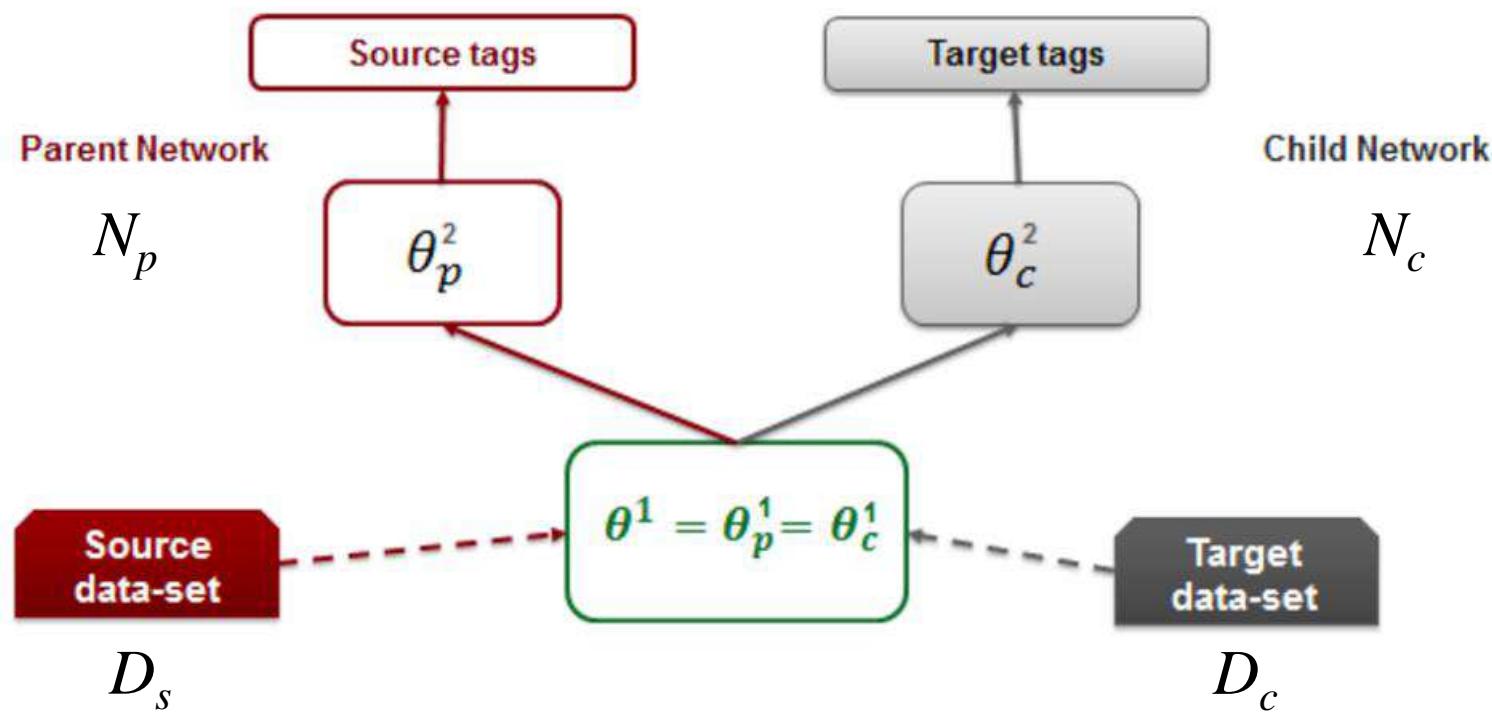
- The source and the target problems share the same domain and the same language
 - Social media text of the same language
- Tasks are different
 - The source problem's task is NER
 - The target's is POS tagging

The parent neural network N_p and the child network N_c share the same first set of parameters

$$\theta_p^1 = \theta_c^1 = \theta^1$$

θ^1 are jointly optimized by the two tasks

Task specific parameters θ_p^2 and θ_c^2 are trained for each task separately.



Data Sets

Language	Domain	Corpus	Task	# Sentences	# Tokens
English	Source	WSJ	POS	67,786	1,2M
	Target	NPS	POS	10,567	45,000
	Target	T-POS	POS	787	15,000
	Target	Ark dataset (Oct27 + Daily547)	POS	1,827 + 547	26,594 + 7,707
French	Source	FTB	POS	21,634	624,187
	Target	French Web 2.0	POS	1,700	20,557
	Target	ExtremeUGC	POS	974	8,099
Spanish	Source	xLime Spanish NER	NER	7,668	140,852
	Target	xLime Spanish POS	POS	7,668	140,852
German	Source	xLime German NER	NER	3,400	60,873
	Target	xLime German POS	POS	3,400	60,873
Italian	Source	xLime Italian NER	NER	8,601	162,269
	Target	xLime Italian POS	POS	8,601	162,269

Experimental Results

Language	English			French		Spanish	German	Italian
Dataset	T-Pos	ARK	NPS	Fr2.0	UGC	xlime		
Acc. without transfer Learning (%)	89.13	91.33	92.9	91.14	87.89	90.87	90.1	89.41
Acc. with transfer learning (%)	90.90	92.01	93.2	91.99	88.07	91.03	90.33	89.66

Source of This Lecture

- Chapter 8, Speech and Language Processing
- Chapter 9, Foundations of Statistical Natural Language Processing
- Chapter 16, Neural Network Methods for Natural Language Processing
- Papers

Lecture 7. Syntax and Parsing

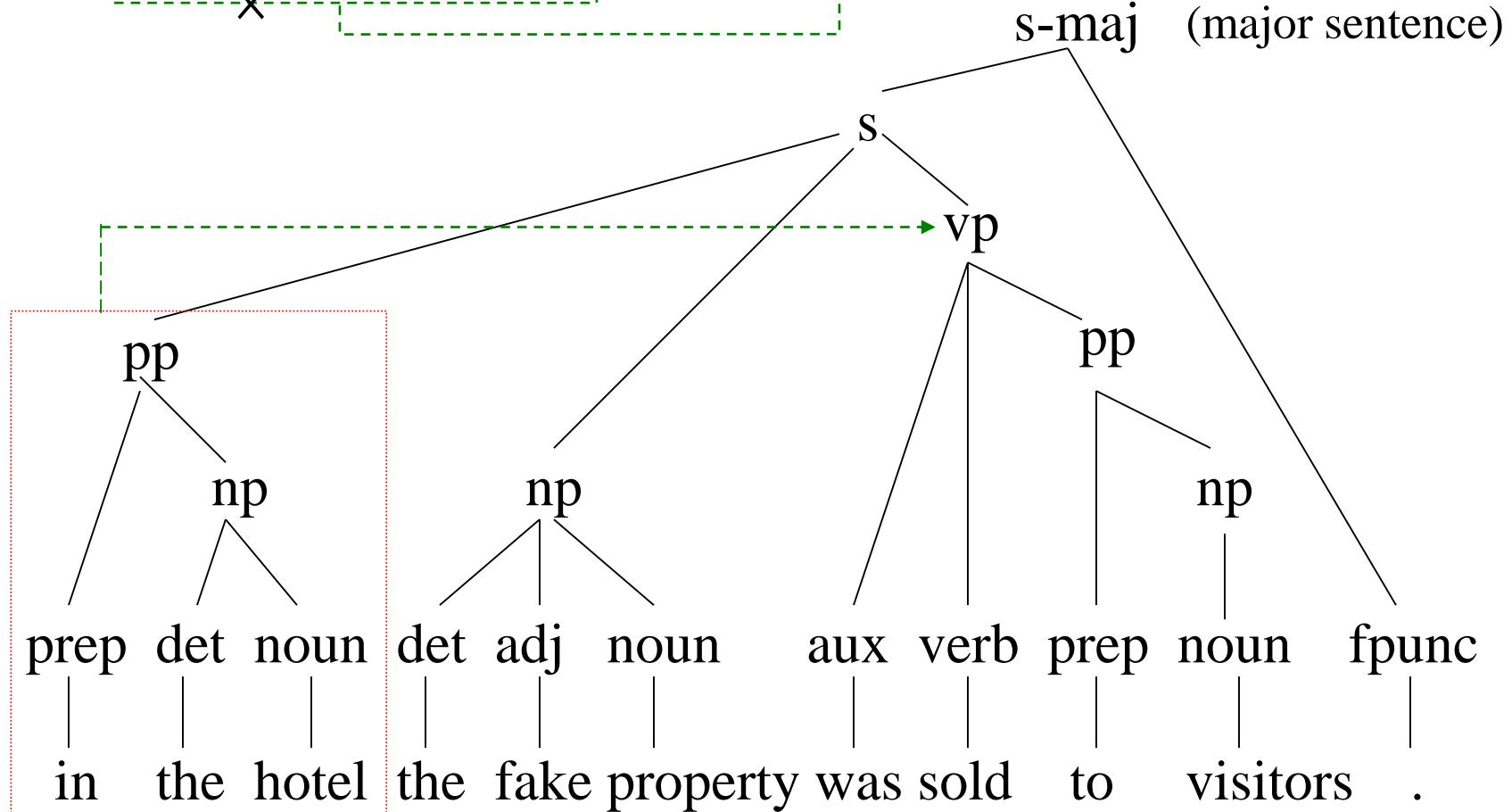
Issues in Syntactic Level

- From Word and MWE to Sentence
- From Sequence to Tree (Lecture 7)
- From Statistic Parsing to Neural Network Parsing (Lecture 8)
- Dependency Parsing (Lecture 9)

Syntax

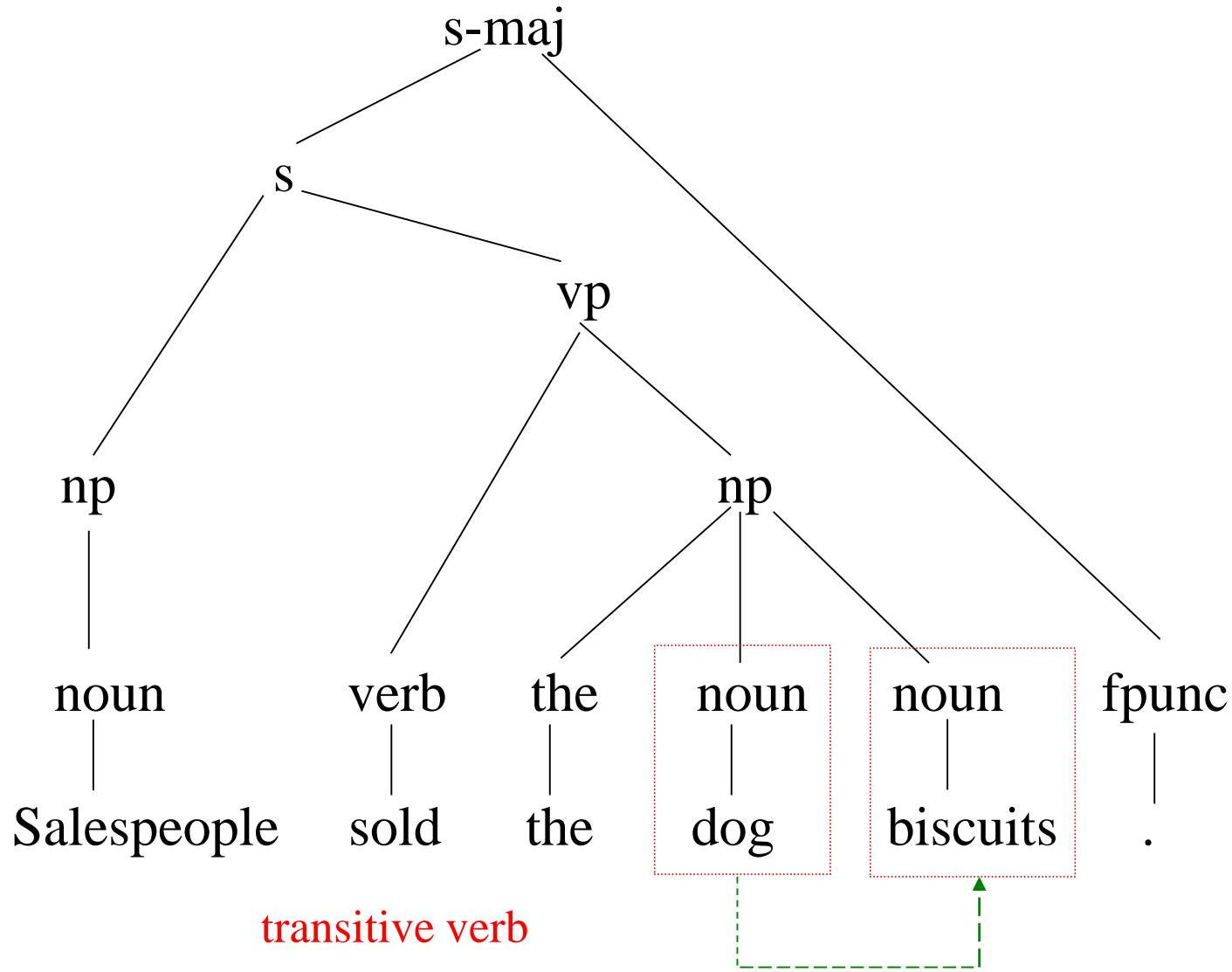
Syntactic Structure

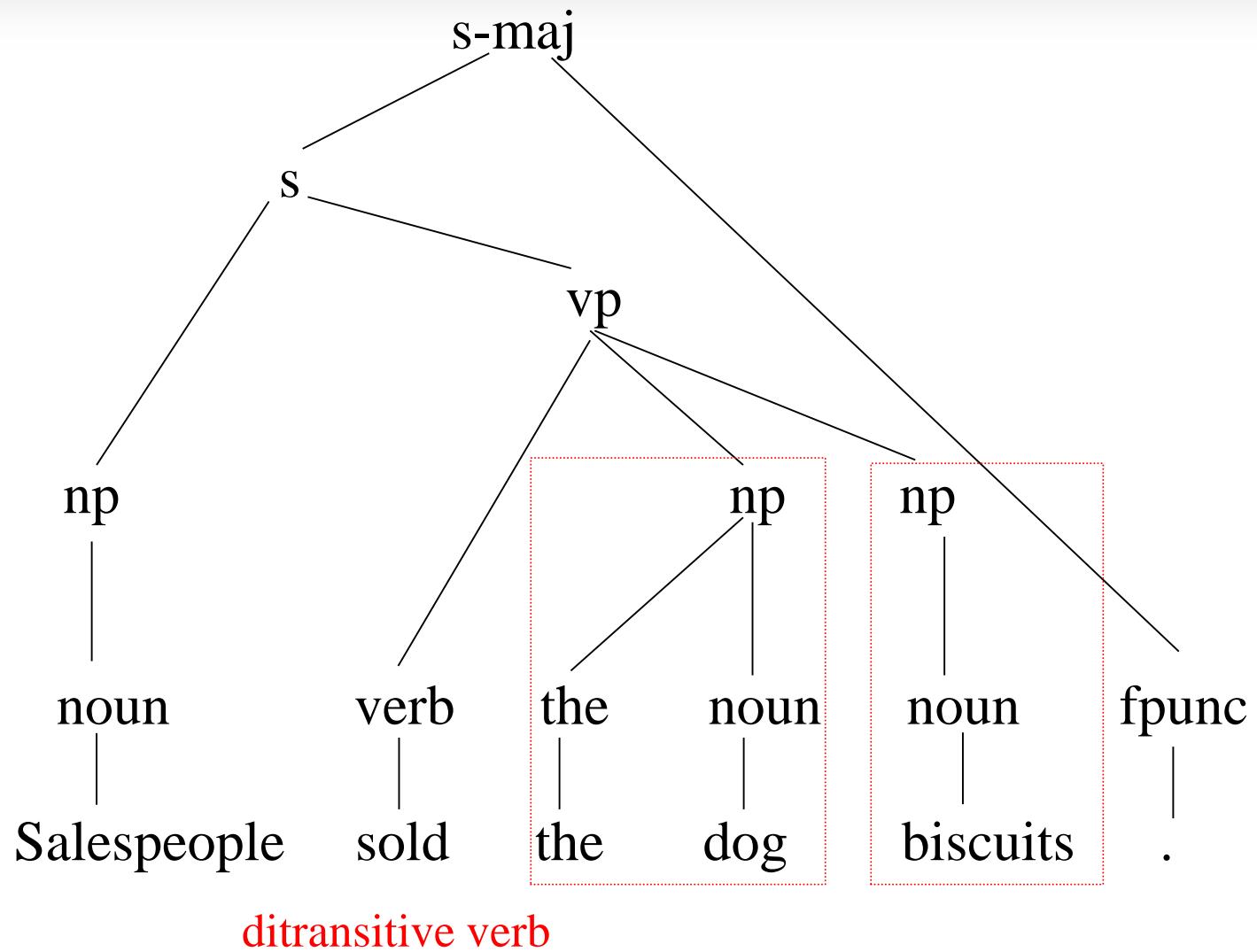
“In the hotel the fake property was sold to visitors.”



Ambiguity

“Salespeople sold the dog biscuits.”





Context-Free Grammars

- Context-free grammars (CFGs)
- Consist of
 - Rules
 - Terminals
 - Non-terminals

Context-Free Grammars

- Terminals
 - words
- Non-Terminals
 - The constituents in a language
 - noun phrase, verb phrase and sentence
 - Syntactic Category vs. Lexical Category (POS tag)
- Rules
 - Rules consist of a single non-terminal on the left and any number of terminals and non-terminals on the right.

Definition

- Formally, a CFG consists of

N a set of **non-terminal symbols** (or **variables**)

Σ a set of **terminal symbols** (disjoint from N)

R a set of **rules** or productions, each of the form $A \rightarrow \beta$,
where A is a non-terminal,

β is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$

S a designated **start symbol**

L0 Lexicon

Noun → *flights | breeze | trip | morning*

Verb → *is | prefer | like | need | want | fly*

Adjective → *cheapest | non-stop | first | latest
| other | direct*

Pronoun → *me | I | you | it*

Proper-Noun → *Alaska | Baltimore | Los Angeles
| Chicago | United | American*

Determiner → *the | a | an | this | these | that*

Preposition → *from | to | on | near*

Conjunction → *and | or | but*

L0 Grammar

Grammar Rules

Examples

$S \rightarrow NP VP$

I + want a morning flight

$NP \rightarrow Pronoun$

I

| $Proper-Noun$

Los Angeles

| $Det Nominal$

a + flight

$Nominal \rightarrow Nominal Noun$

morning + flight

| $Noun$

flights

$VP \rightarrow Verb$

do

| $Verb NP$

want + a flight

| $Verb NP PP$

leave + Boston + in the morning

| $Verb PP$

leaving + on Thursday

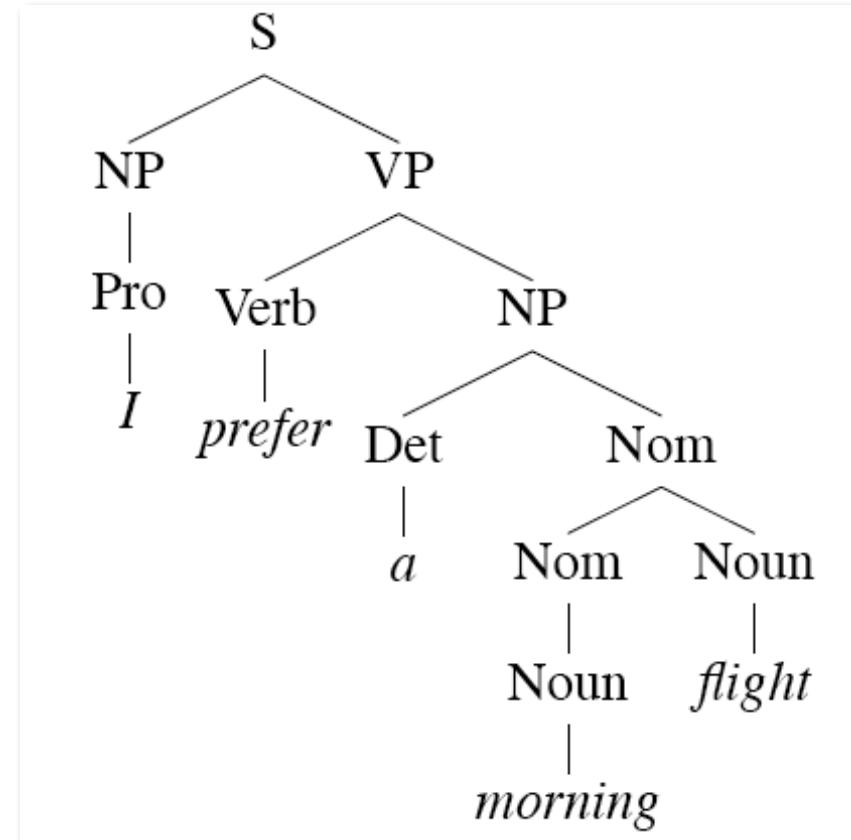
$PP \rightarrow Preposition NP$

from + Los Angeles

Derivations

- A *derivation* is a sequence of rules applied to a string that *accounts* for that string

- Covers all the elements in the string
- Covers only the elements in the string



Let $\alpha_1, \alpha_2, \dots, \alpha_m$ be strings in $(\Sigma \cup N)^*$, $m \geq 1$, such that

$$\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \dots, \alpha_{m-1} \Rightarrow \alpha_m$$

$$\mathcal{L}_G = \{w \mid w \text{ is in } \Sigma^* \text{ and } S \xrightarrow{*} w\}$$

We say that α_1 **derives** α_m , or $\alpha_1 \xrightarrow{*} \alpha_m$.

Syntactic Parsing

- Parsing is the process of taking a string and a grammar and returning parse tree(s) for that string
- Mapping from a string of words to its parse tree

An English Grammar Fragment

- Sentences
- Noun phrases
- Verb phrases
 - Subcategorization

Sentence Types

- Declaratives: *A plane left.*

$$S \rightarrow NP\ VP$$

- Imperatives: *Leave!*

$$S \rightarrow VP$$

- Yes-No Questions: *Did the plane leave?*

$$S \rightarrow Aux\ NP\ VP$$

- WH Questions: *When did the plane leave?*

$$S \rightarrow WH\text{-}NP\ Aux\ NP\ VP$$

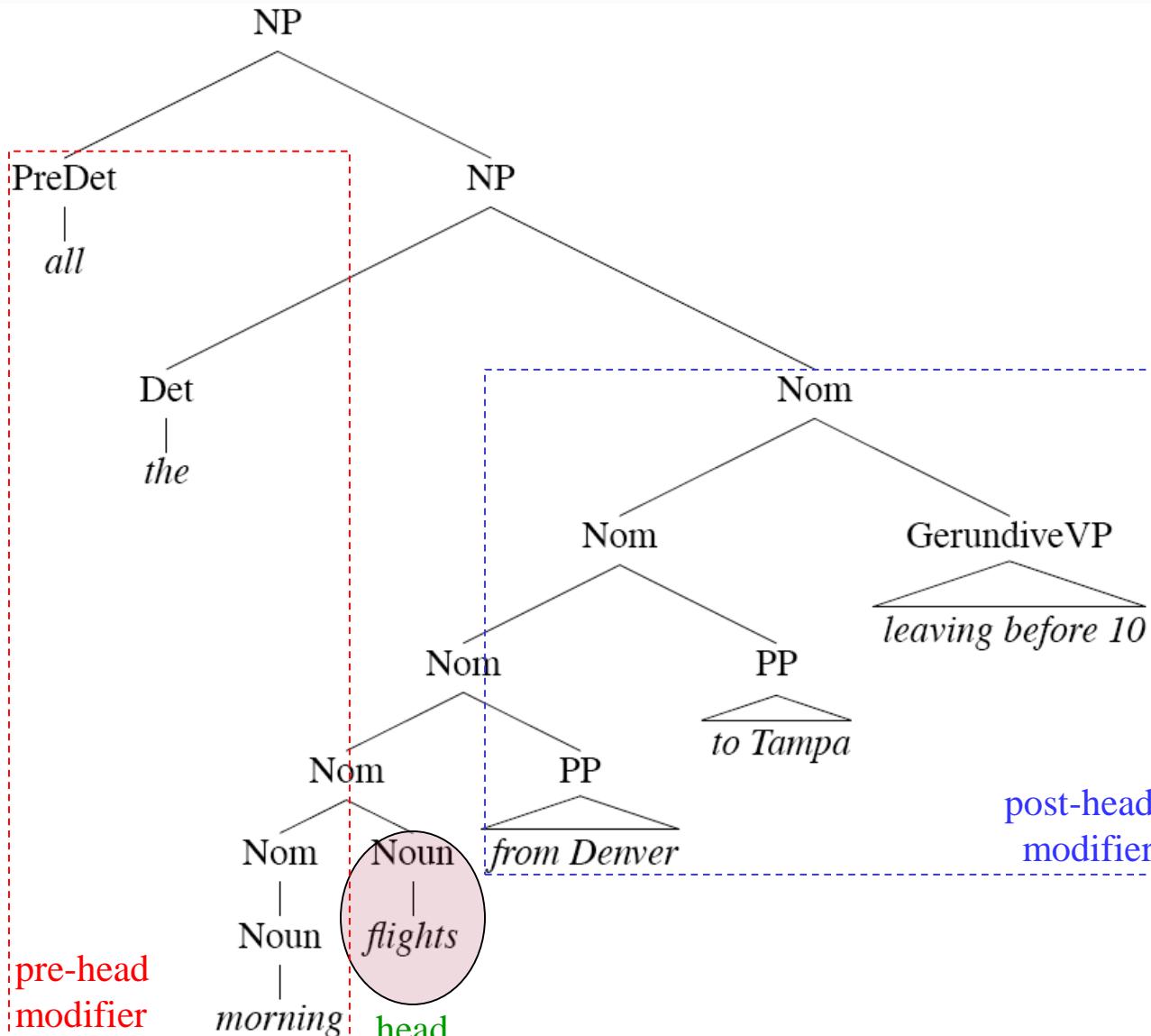
Noun Phrases

- Consider the following rule in more detail.

$NP \rightarrow Det\ Nominal$

- The nominal construction follows the determiner and contains any pre- and post-**head noun** modifiers.
- Consider the derivation for the following example
 - *All the morning flights from Denver to Tampa leaving before 10*

Noun Phrases



Verb Phrases

- English VPs consist of a **head verb** along with 0 or **more following constituents** which we'll call *arguments*.

$VP \rightarrow Verb$ disappear

$VP \rightarrow Verb\ NP$ prefer a morning flight

$VP \rightarrow Verb\ NP\ PP$ leave Boston in the morning

$VP \rightarrow Verb\ PP$ leaving on Thursday

Subcategorization

- We can subcategorize the verbs in a language according to the sets of VP rules that they participate in.
- Sneeze: John sneezed
- Find: Please find [a flight to NY]_{NP}
- Give: Give [me]_{NP}[a cheaper fare]_{NP}
- Help: Can you help [me]_{NP}[with a flight]_{PP}
- Prefer: I prefer [to leave earlier]_{TO-VP}
- Told: I was told [United has a flight]_S
- ...

Subcategorization Frame

- The possible sets of complements for a verb
- The relation between the verb and the other constituents

Frame	Verb	Example
\emptyset	eat, sleep	I ate
NP	prefer, find, leave	Find [NP the flight from Pittsburgh to Boston]
$NP\ NP$	show, give	Show [NP me] [NP airlines with flights from Pittsburgh]
$PP_{from}\ PP_{to}$	fly, travel	I would like to fly [PP from Boston] [PP to Philadelphia]
$NP\ PP_{with}$	help, load	Can you help [NP me] [PP with a flight]
VP_{to}	prefer, want, need	I would prefer [VP_{to} to go by United airlines]
VP_{brst}	can, would, might	I can [VP_{brst} go from Boston]
S	mean	Does this mean [S AA has a hub in Boston]

Academic Sinica Dictionary

```
<Word item = "打">
<WordFreq>1461</WordFreq>
<WordSense id="1">
    <English>generalized verb of doing with specific meaning determined by its object, strike, hit, fight, construct, forge, mix</English>
    <Phone>ㄉㄚˇ</Phone>
    <PinYin>da3</PinYin>
    <SyntacticFunction>
        <POS>VC2</POS>
        <Freq>1448</Freq>
    </SyntacticFunction>
    <TopLevelDefinition>{do|做}</TopLevelDefinition>
    <BottomLevelExpansion>{do|做}</BottomLevelExpansion>
</WordSense>
<WordSense id="2">
    <English>dozen</English>
    <Phone>ㄉㄚˇ</Phone>
    <PinYin>da3</PinYin>
    <SyntacticFunction>
        <POS>Nfg</POS>
        <Freq>13</Freq>
    </SyntacticFunction>
    <TopLevelDefinition>quantity={打}</TopLevelDefinition>
    <BottomLevelExpansion>quantity=(打)</BottomLevelExpansion>
</WordSense>
```

單賓 「打手飾」、「打毛衣」

標準量詞 「一打毛巾」

```
<WordSense id="3">
    <English>beat</English>
    <Phone>ㄉㄚˋ</Phone>
    <PinYin>da3</PinYin>
    <SyntacticFunction>
        <POS>VC2</POS>
        <Freq>0</Freq>
    </SyntacticFunction>
    <TopLevelDefinition>{beat|打}</TopLevelDefinition>
    <BottomLevelExpansion>{beat|打}</BottomLevelExpansion>
</WordSense>
<WordSense id="4">
    <English>from</English>
    <Phone>ㄉㄚˋ</Phone>
    <PinYin>da3</PinYin>
    <SyntacticFunction>
        <POS>P17</POS>
        <Freq>0</Freq>
    </SyntacticFunction>
    <TopLevelDefinition>LocationIni={}</TopLevelDefinition>
    <BottomLevelExpansion>LocationIni={}</BottomLevelExpansion>
</WordSense>
<WordSense id="5">
    <English>since</English>
    <Phone>ㄉㄚˋ</Phone>
    <PinYin>da3</PinYin>
    <SyntacticFunction>
        <POS>P17</POS>
        <Freq>0</Freq>
    </SyntacticFunction>
    <TopLevelDefinition>TimeIni={}</TopLevelDefinition>
    <BottomLevelExpansion>TimeIni={}</BottomLevelExpansion>
</WordSense>
```

單賓

「打壞人」

介詞

「您打那裡來？」

介詞 「打現在起，我要發奮用功。」

Chinese PropBank ([→](#))

- [arg0 我][argm-adv 已經][rel 打][arg1 電話][arg2 紿斯恩特]
- [arg1 這些算盤][arg0 產業界自己][rel 打]的[argm-ext 最精]
- [arg1 鮑薩]被[arg0 泰森的鐵拳][rel 打]得[arg2 爬不起來]
- [arg0 他][argm-tmp 晚上]則到體育場[rel 打][arg1 籃球]
- http://verbs.colorado.edu/chinese/cpb/html_frames/0614-da.html
- More examples 1-4937 ([→](#))

ArgM-EXT indicates the amount of change occurring from an action

Chinese Proposition Bank 3.0

- Jul 15, 2013
- Predicate-argument annotation on 187,731 words from Chinese Treebank 7.0
 - 173,206 verb instances
 - 14,525 noun instances
- newswire, magazine articles, various broadcast news and broadcast conversation programming, web newsgroups and weblogs

FrameNet

<https://framenet.icsi.berkeley.edu/fndrupal/about>

- A dictionary of more than 10,000 word senses
- More than 170,000 manually annotated sentences provide a unique training dataset
 - semantic role labeling
 - information extraction
 - machine translation
 - event recognition
 - sentiment analysis
 - etc.

Treebanks

- Treebanks are corpora in which each sentence has been paired with a parse tree.
- These are generally created
 - By first parsing the collection with an automatic parser
 - And then having human annotators correct each parse as necessary.
- This generally requires detailed annotation guidelines that provide a POS tagset, a grammar and instructions for how to deal with particular grammatical constructions.

Penn Treebank

- Penn TreeBank is a widely used treebank.

Most well known part is the Wall Street Journal section of the Penn TreeBank.

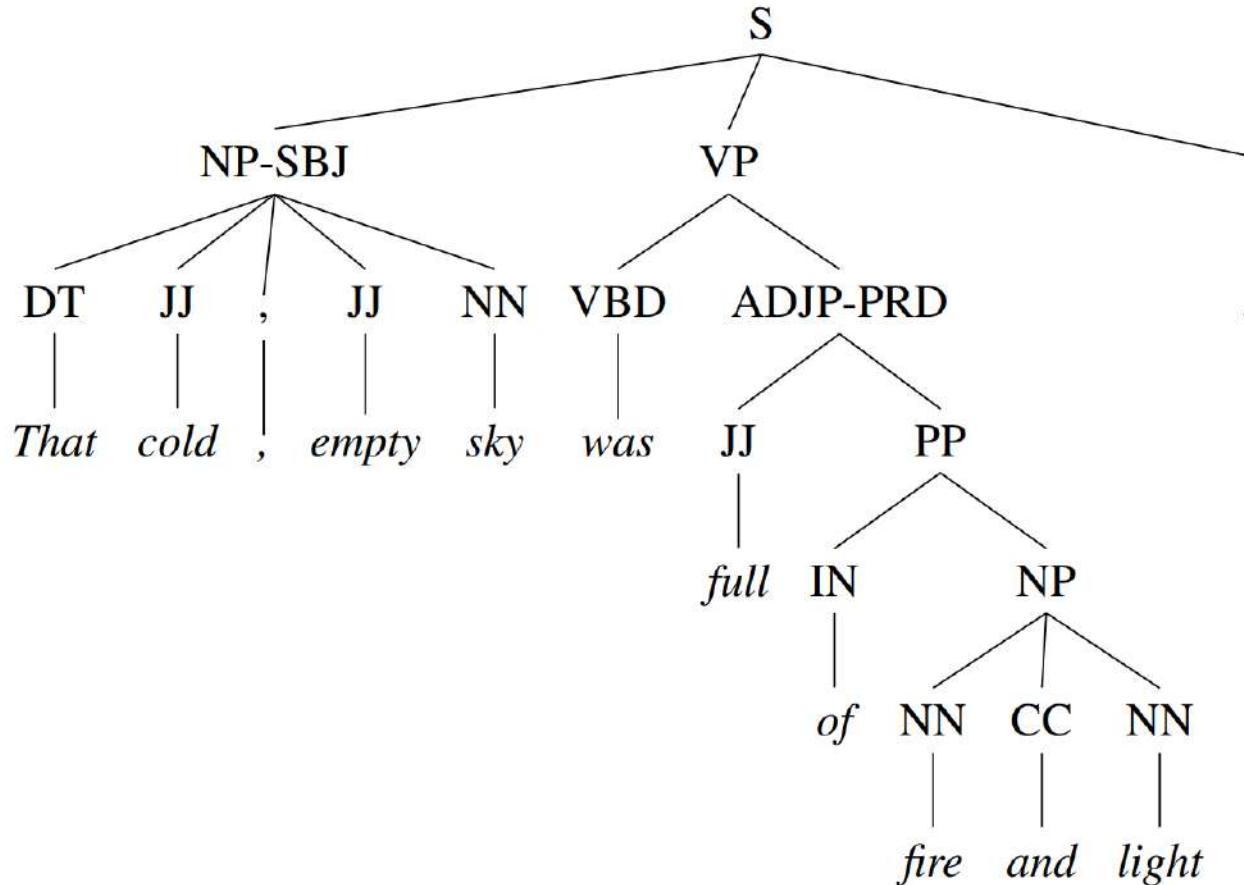
- 1 M words from the 1987-1989 Wall Street Journal.

```
( (S ( ' ' ' )
      (S-TPC-2
        (NP-SBJ-1 (PRP We) )
        (VP (MD would)
          (VP (VB have)
            (S
              (NP-SBJ (-NONE- *-1) )
              (VP (TO to)
                (VP (VB wait)
                  (SBAR-TMP (IN until)
                    (S
                      (NP-SBJ (PRP we) )
                      (VP (VBP have)
                        (VP (VBN collected)
                          (PP-CLR (IN on)
                            (NP (DT those)(NNS assets))))))))))))
            ( , , ) ( ' ' ')
            (NP-SBJ (PRP he) )
            (VP (VBD said)
              (S (-NONE- *T*-2) )))
            ( . . ) ))
```

Parsed sentences from the LDC Treebank3 version of the Brown

- Parenthesized notation
- Tree notation

```
((S
  (NP-SBJ (DT That)
    (JJ cold) (, ,)
    (JJ empty) (NN sky) )
  (VP (VBD was)
    (ADJP-PRD (JJ full)
      (PP (IN of)
        (NP (NN fire)
          (CC and)
          (NN light) )))))
  (. .) ))
```



The Use of the Empty -NONE- Nodes

We would have to wait until we have collected on those assets, he said.

```
( (S (‘ ‘))
  (S-TPC-2
    (NP-SBJ-1 (PRP We) )
    (VP (MD would)
      (VP (VB have)
        (S
          (NP-SBJ (-NONE-*1) )
          (VP (TO to)
            (VP (VB wait)
              (SBAR-TMP (IN until)
                (S
                  (NP-SBJ (PRP we) )
                  (VP (VBP have)
                    (VP (VBN collected)
                      (PP-CLR (IN on)
                        (NP (DT those)(NNS assets))))))))))))
      (, ,) (‘ ‘)
      (NP-SBJ (PRP he) )
      (VP (VBD said)
        (S (-NONE-*T*-2) )))
    (. .) ))
```

the subject is the earlier NP *We*

Treebank Grammars

- Treebanks implicitly define a grammar for the language covered in the treebank.
- Simply take the local rules that make up the sub-trees in all the trees in the collection and you have a grammar
 - The WSJ section gives us about 12k rules if you do this
- Not complete, but if you have decent size corpus, you'll have a grammar with decent coverage.

```

((S
  (NP-SBJ (DT That)
    (JJ cold) (, ,)
    (JJ empty) (NN sky) )
  (VP (VBD was)
    (ADJP-PRD (JJ full)
      (PP (IN of)
        (NP (NN fire)
          (CC and)
          (NN light) )))))
  (. .) )

( (S (‘ ‘ ‘)
  (S-TPC-2
    (NP-SBJ-1 (PRP We) )
    (VP (MD would)
      (VP (VB have)
        (S
          (NP-SBJ (-NONE- *-1) )
          (VP (TO to)
            (VP (VB wait)
              (SBAR-TMP (IN until)
                (S
                  (NP-SBJ (PRP we) )
                  (VP (VBP have)
                    (VP (VBN collected)
                      (PP-CLR (IN on)
                        (NP (DT those)(NNS assets))))))))))))
  (, ,) (‘ ‘ ‘)
  (NP-SBJ (PRP he) )
  (VP (VBD said)
    (S (-NONE- *T*-2) ))
  (. .) )

```

Grammar	Lexicon
$S \rightarrow NP\ VP.$	$PRP \rightarrow we he$
$S \rightarrow NP\ VP$	$DT \rightarrow the that those$
$S \rightarrow "S",\ NP\ VP.$	$JJ \rightarrow cold empty full$
$S \rightarrow -NONE-$	$NN \rightarrow sky fire light flight tomorrow$
$NP \rightarrow DT\ NN$	$NNS \rightarrow assets$
$NP \rightarrow DT\ NNS$	$CC \rightarrow and$
$NP \rightarrow NN\ CC\ NN$	$IN \rightarrow of at until on$
$NP \rightarrow CD\ RB$	$CD \rightarrow eleven$
$NP \rightarrow DT\ JJ,\ JJ\ NN$	$RB \rightarrow a.m.$
$NP \rightarrow PRP$	$VB \rightarrow arrive have wait$
$NP \rightarrow -NONE-$	$VBD \rightarrow was said$
$VP \rightarrow MD\ VP$	$VBP \rightarrow have$
$VP \rightarrow VBD\ ADJP$	$VBN \rightarrow collected$
$VP \rightarrow VBD\ S$	$MD \rightarrow should would$
$VP \rightarrow VBN\ PP$	$TO \rightarrow to$
$VP \rightarrow VB\ S$	
$VP \rightarrow VB\ SBAR$	
$VP \rightarrow VBP\ VP$	
$VP \rightarrow VBN\ PP$	
$VP \rightarrow TO\ VP$	
$SBAR \rightarrow IN\ S$	
$ADJP \rightarrow JJ\ PP$	
$PP \rightarrow IN\ NP$	

Treebank Grammars

- Such grammars tend to be very flat due to the fact that they tend to avoid recursion.
 - To ease the annotators burden, among things

VP → VBD PP

VP → VBD PP PP

VP → VBD PP PP PP

VP → VBD PP PP PP PP

Treebank Grammars

- 4,500 different rules for VPs
- many thousands of NP rules

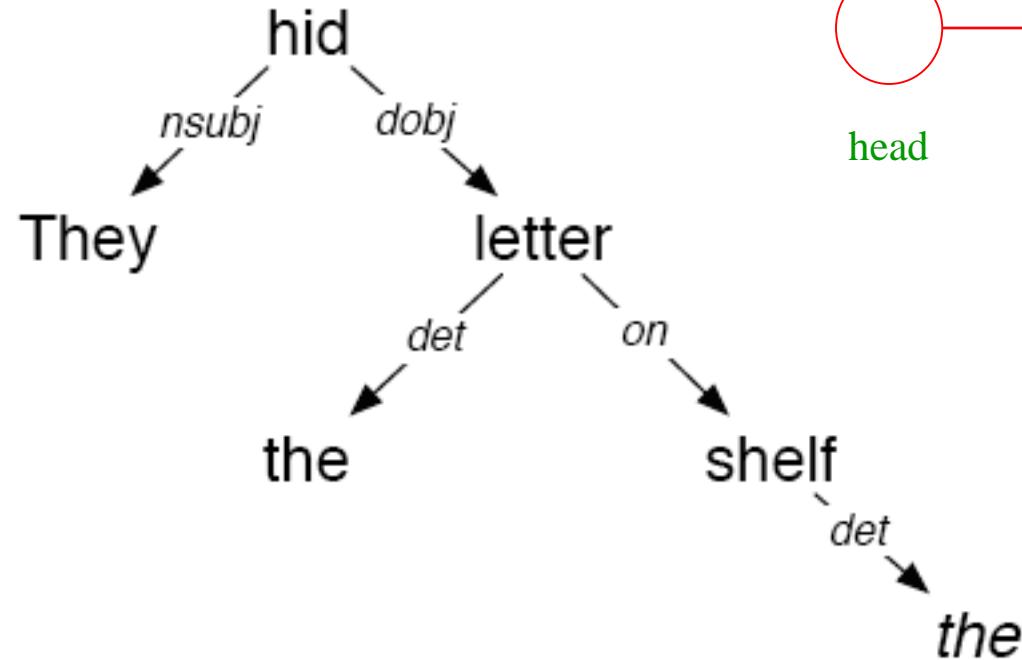
VP → VBD PP
VP → VBD PP PP
VP → VBD PP PP PP
VP → VBD PP PP PP PP
VP → VB ADVP PP
VP → VB PP ADVP
VP → ADVP VB PP

NP → DT JJ NN
NP → DT JJ NNS
NP → DT JJ NN NN
NP → DT JJ JJ NN
NP → DT JJ CD NNS
NP → RB DT JJ NN NN
NP → RB DT JJ JJ NNS
NP → DT JJ JJ NNP NNS
NP → DT NNP NNP NNP NNP JJ NN
NP → DT JJ NNP CC JJ JJ NN NNS
NP → RB DT JJS NN NN SBAR
NP → DT VBG JJ NNP NNP CC NNP
NP → DT JJ NNS , NNS CC NN NNS NN
NP → DT JJ JJ VBG NN NNP NNP FW NNP
NP → NP JJ , JJ ‘‘ SBAR ’’ NNS

Dependency Grammars

- In CFG-style phrase-structure grammars the main focus is on *constituents*.
- But it turns out you can get a lot done with just **binary relations among the words** in an utterance.
- In a dependency grammar framework, a parse is a tree where
 - the nodes stand for the words in an utterance
 - The links between the words represent dependency relations between pairs of words.
 - Relations may be typed (labeled), or not.

Dependency Parse



They hid the letter on the shelf

Dependency Relations

Argument Dependencies	Description
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition

Modifier Dependencies	Description
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier

Dependency Parsing

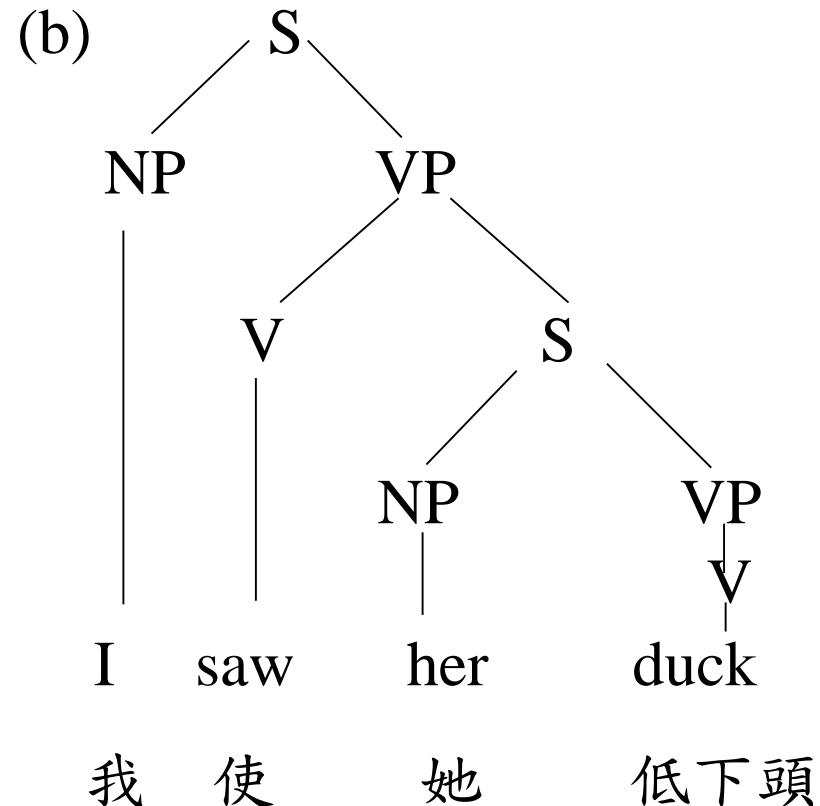
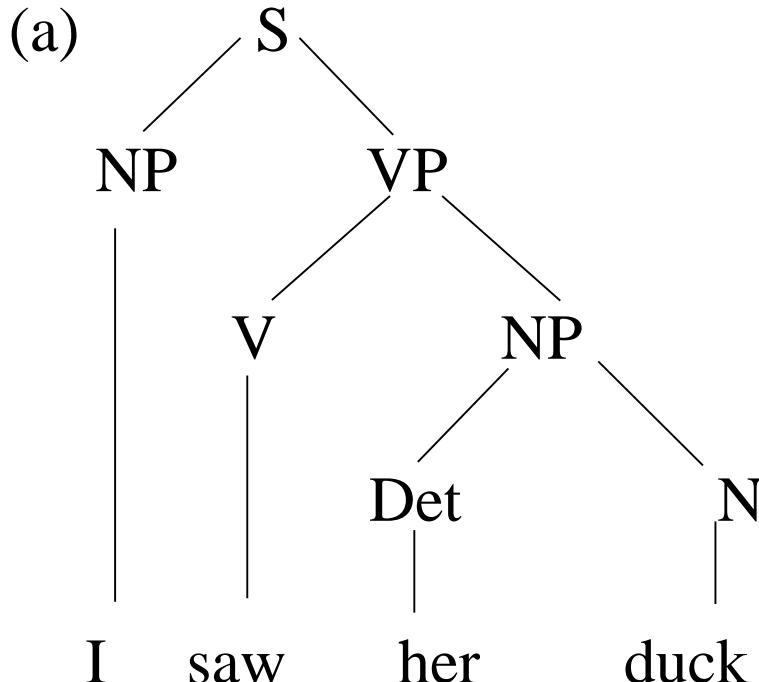
- The dependency approach has a number of advantages over full phrase-structure parsing.
 - Deals well with **free word order languages** where the constituent structure is quite fluid
 - Parsing is ***much faster*** than CFG-based parsers
 - Dependency structure often ***captures the syntactic relations*** needed by later applications
 - CFG-based approaches often extract this same information from trees anyway

Dependency Parsing

- There are two modern approaches to dependency parsing
 - **Optimization-based approaches** that search a space of trees for the tree that *best* matches some criteria
 - Think of spanning tree algorithms
 - **Shift-reduce approaches** that greedily take actions based on the current word and state.

Syntactic Parsing

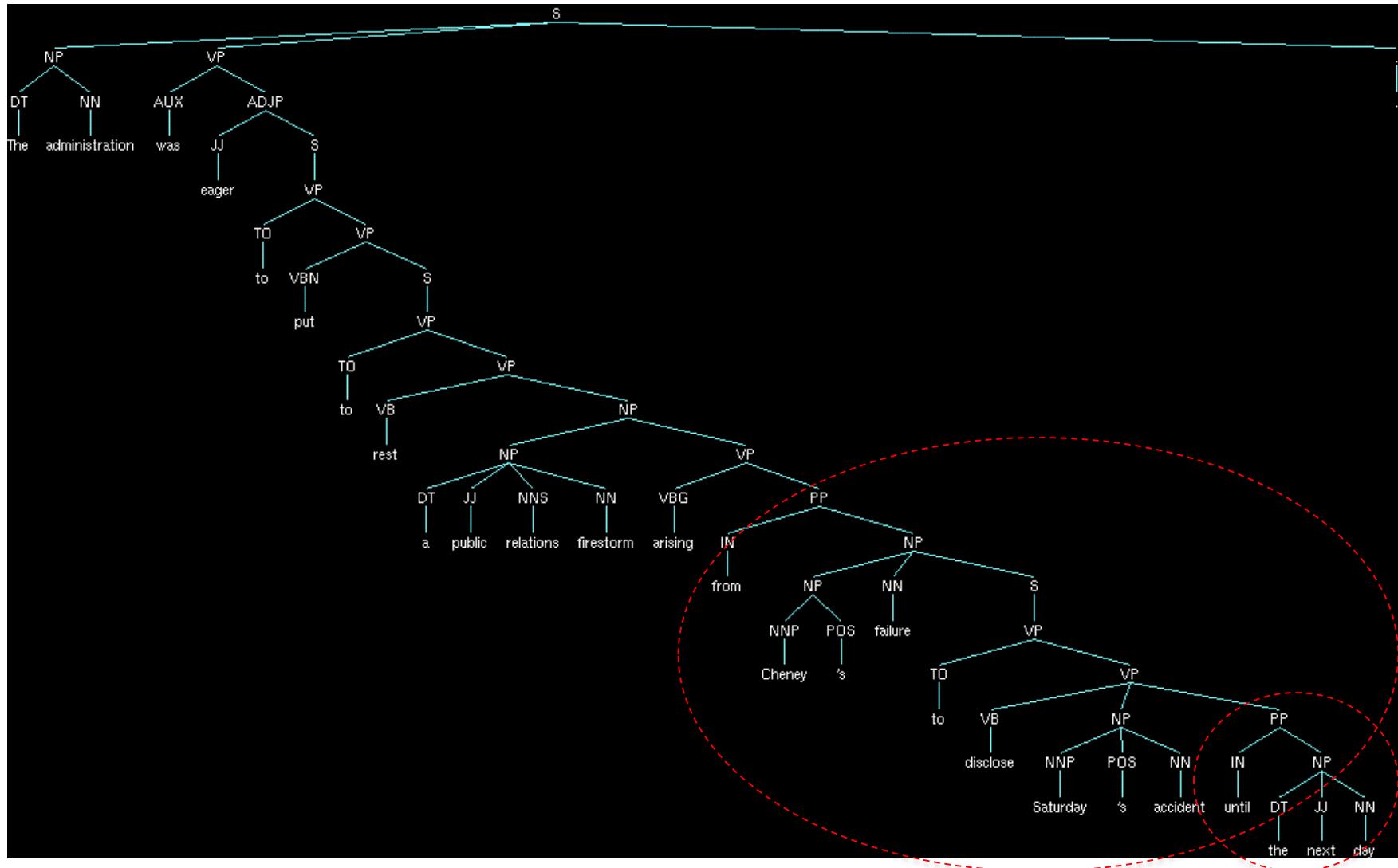
Deep vs. Shallow Parsing



$[I]_{np} [saw]_v [her\ duck]_{np}$

chunking

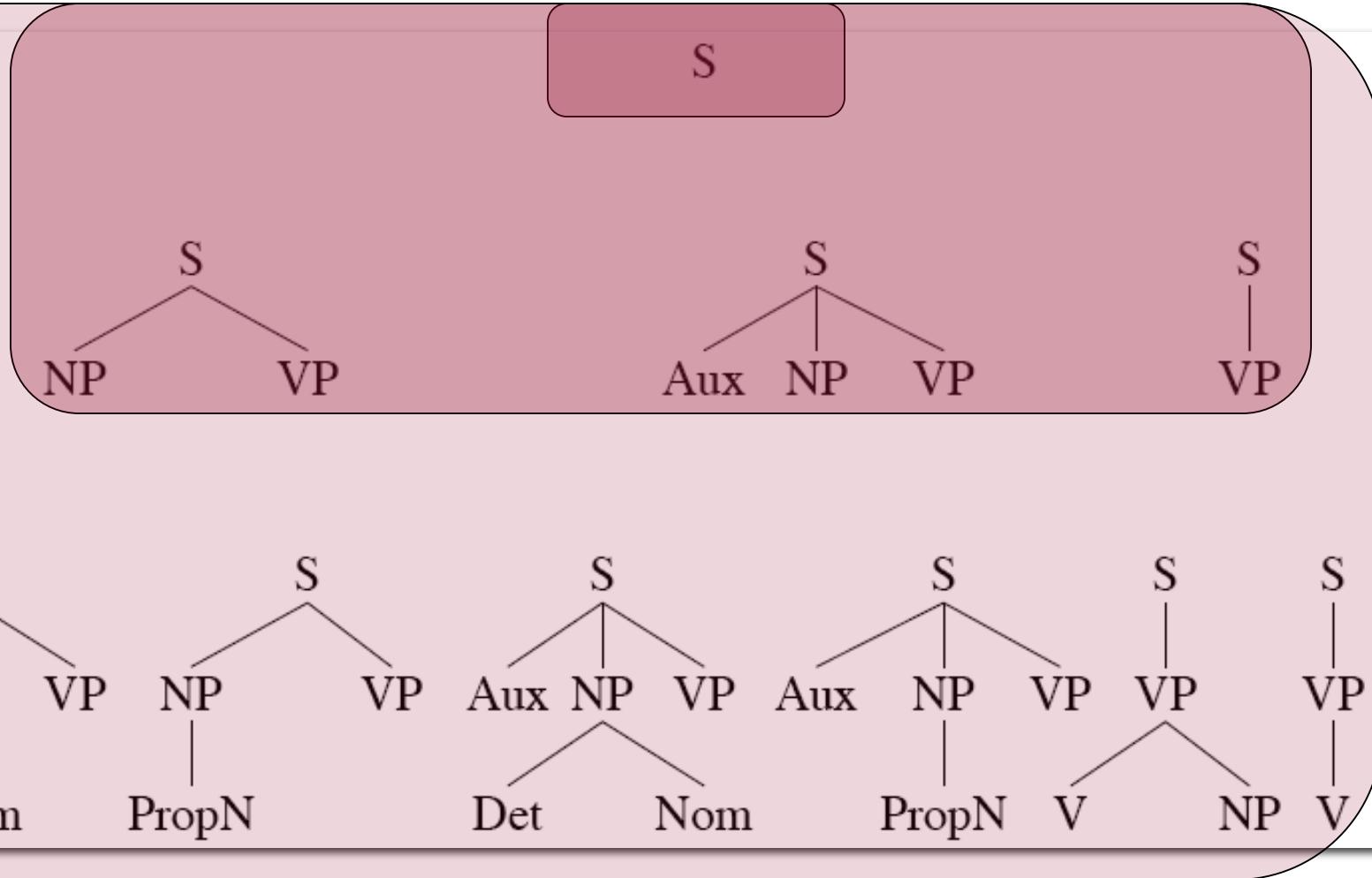
Automatic Syntactic Parse



Parsing

- Parsing with CFGs refers to the task of assigning proper trees to input strings
- Top-Down Search
 - Start with the rules that give us an S.
 - Then we can work our way down from there to the words.
- Bottom-Up Parsing
 - Start with trees that link up with the words in the right way.
 - Then work your way up from there to larger and larger trees.

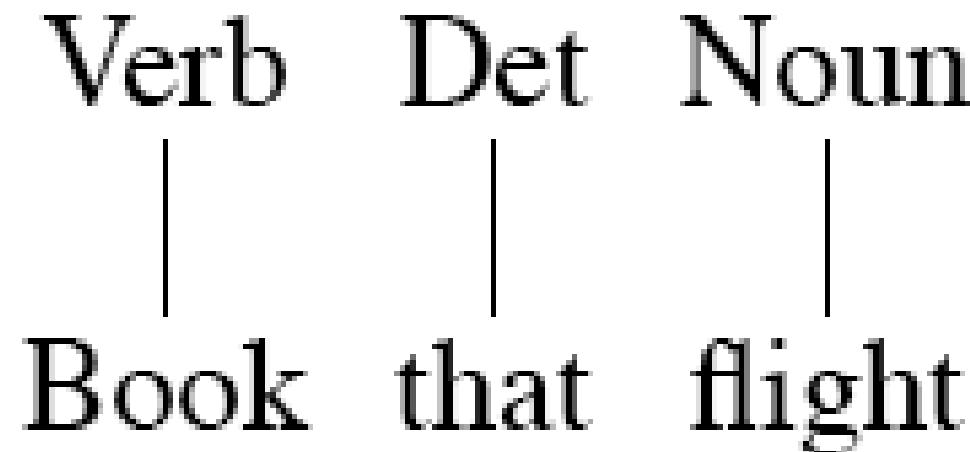
Top Down Space



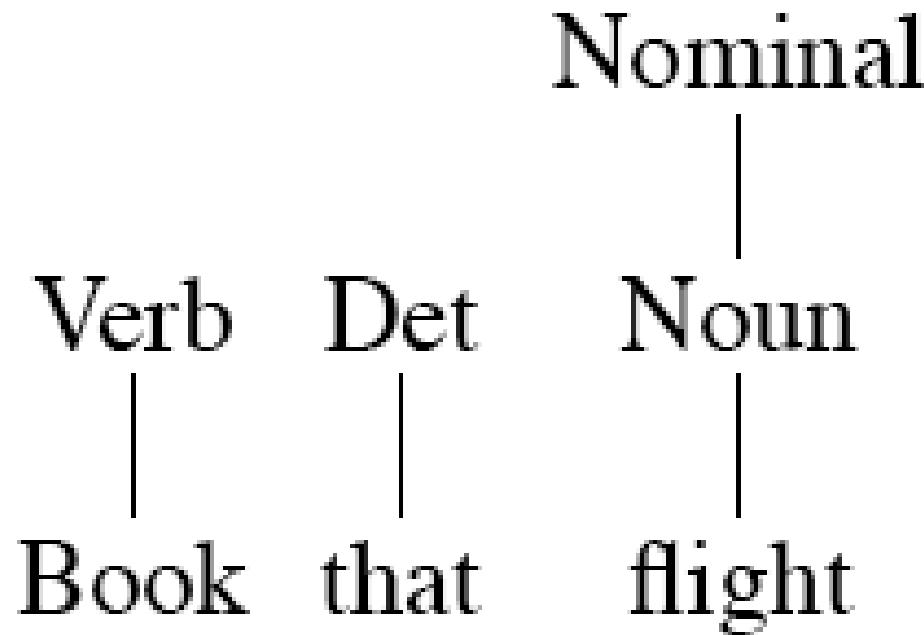
Bottom-Up Search

Book that flight

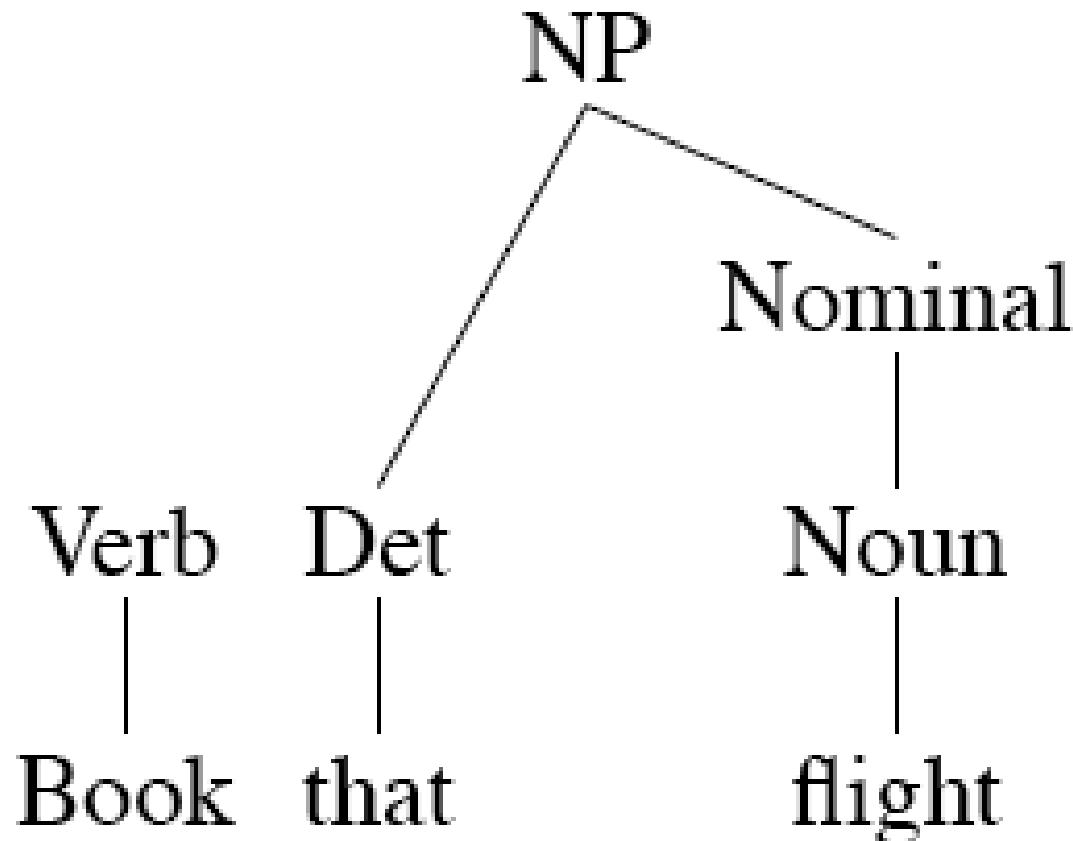
Bottom-Up Search



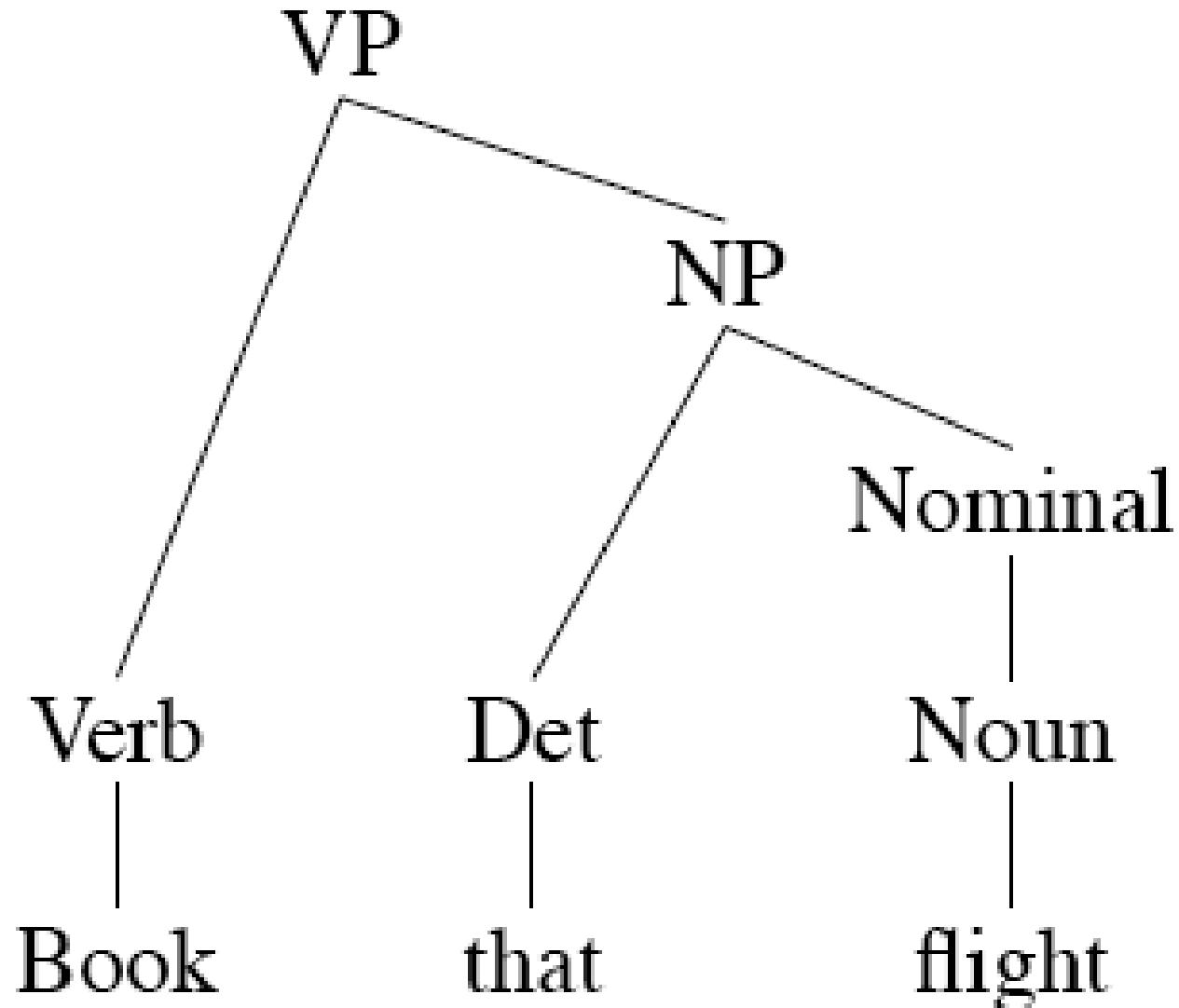
Bottom-Up Search



Bottom-Up Search



Bottom-Up Search



Top-Down and Bottom-Up

■ Top-down

- Only searches for trees that can be answers (i.e. S's)
- But also suggests trees that are not consistent with any of the words

■ Bottom-up

- Only forms trees consistent with the words
- But suggests trees that make no sense globally

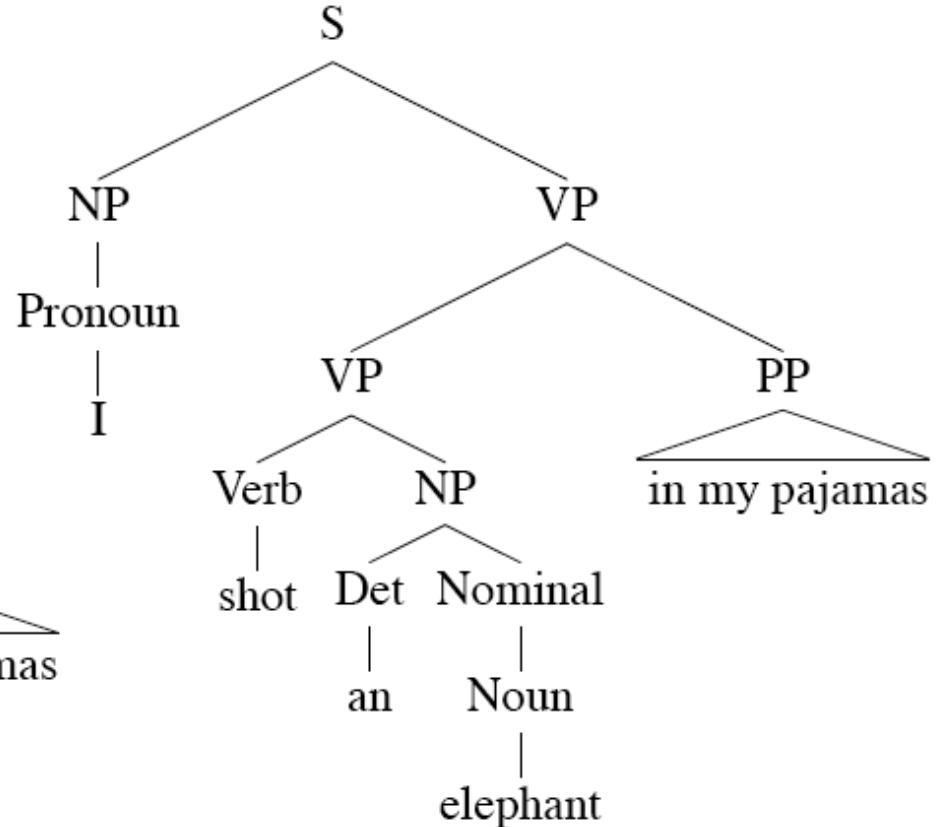
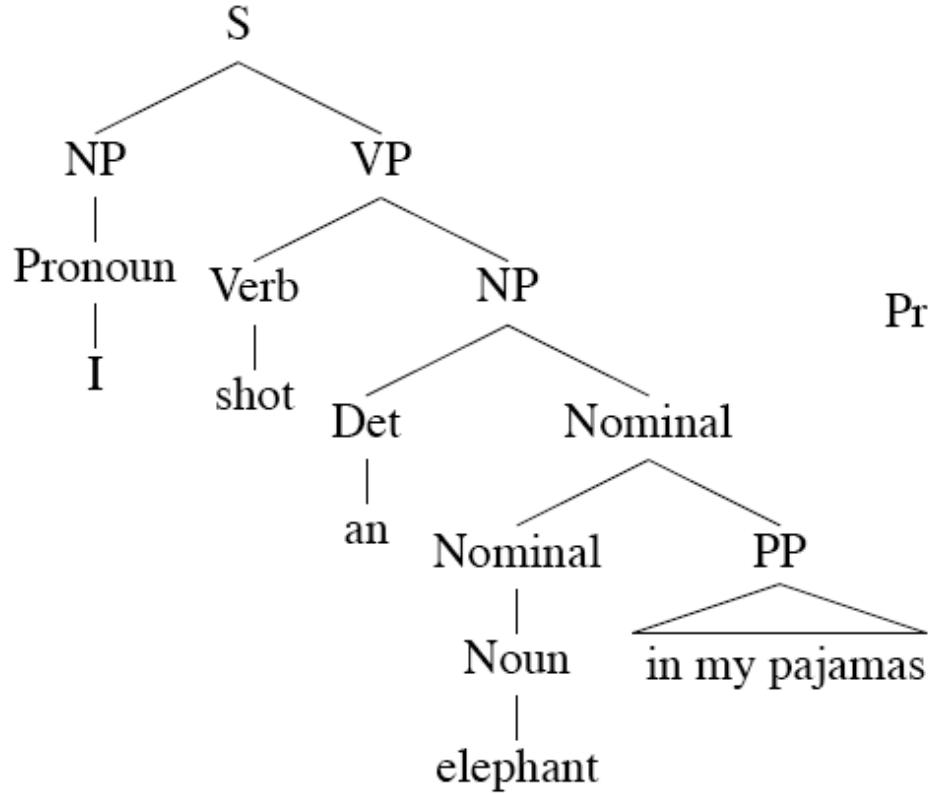
Control

- How to keep track of the search space and how to make choices
 - Which node to try to expand next
 - Which grammar rule to use to expand a node
- One approach is called backtracking.
 - Make a choice, if it works out then fine
 - If not then back up and make a different choice

Problems

- Even with the best filtering, backtracking methods are doomed because of two inter-related problems
 - Ambiguity
 - Shared subproblems

Ambiguity



Ambiguity

- lexical ambiguity

Word nurses:

$\langle \text{cat} \rangle = \text{NP}$.

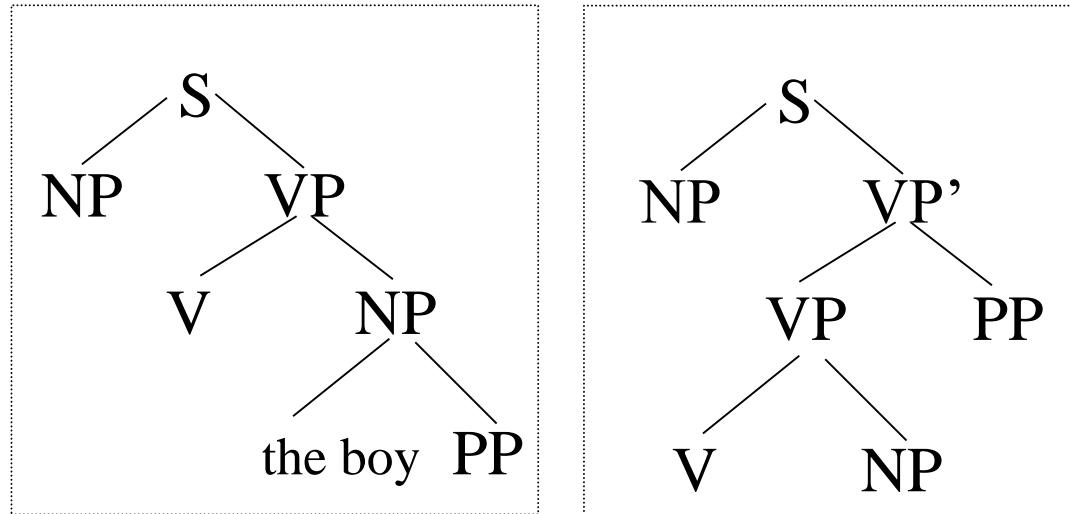
Word nurses:

$\langle \text{cat} \rangle = \text{V}$. Dr. Chan nurses patients.

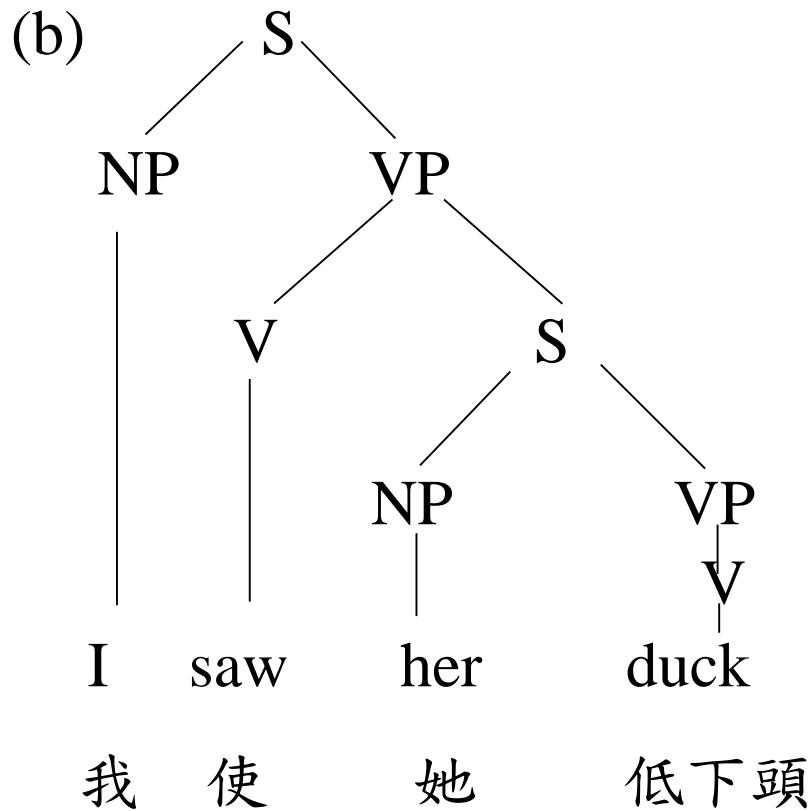
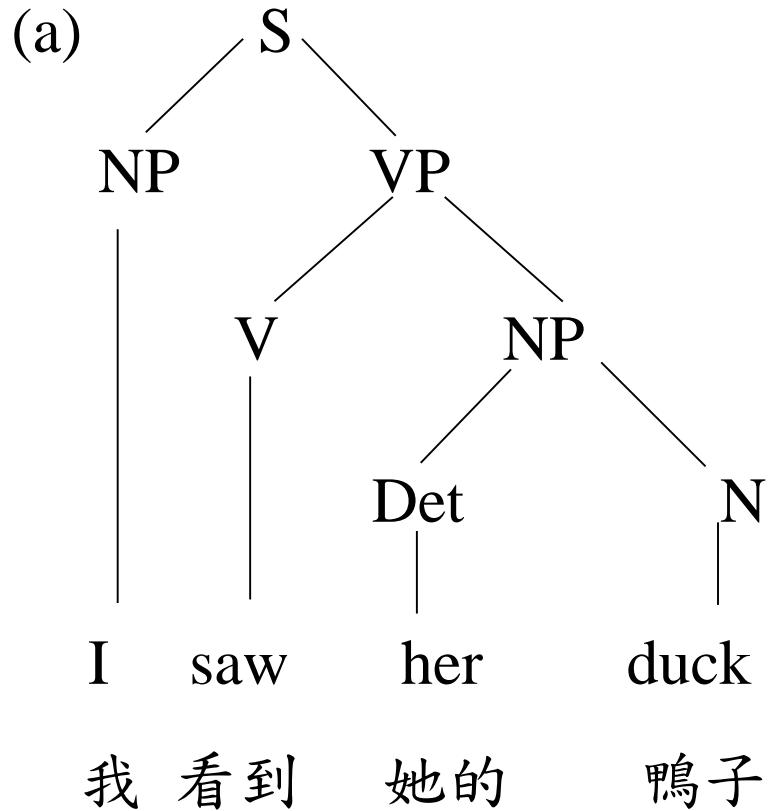
I walked to the bank. (岸/銀行)

- structural ambiguity

I observed the boy with a telescope.



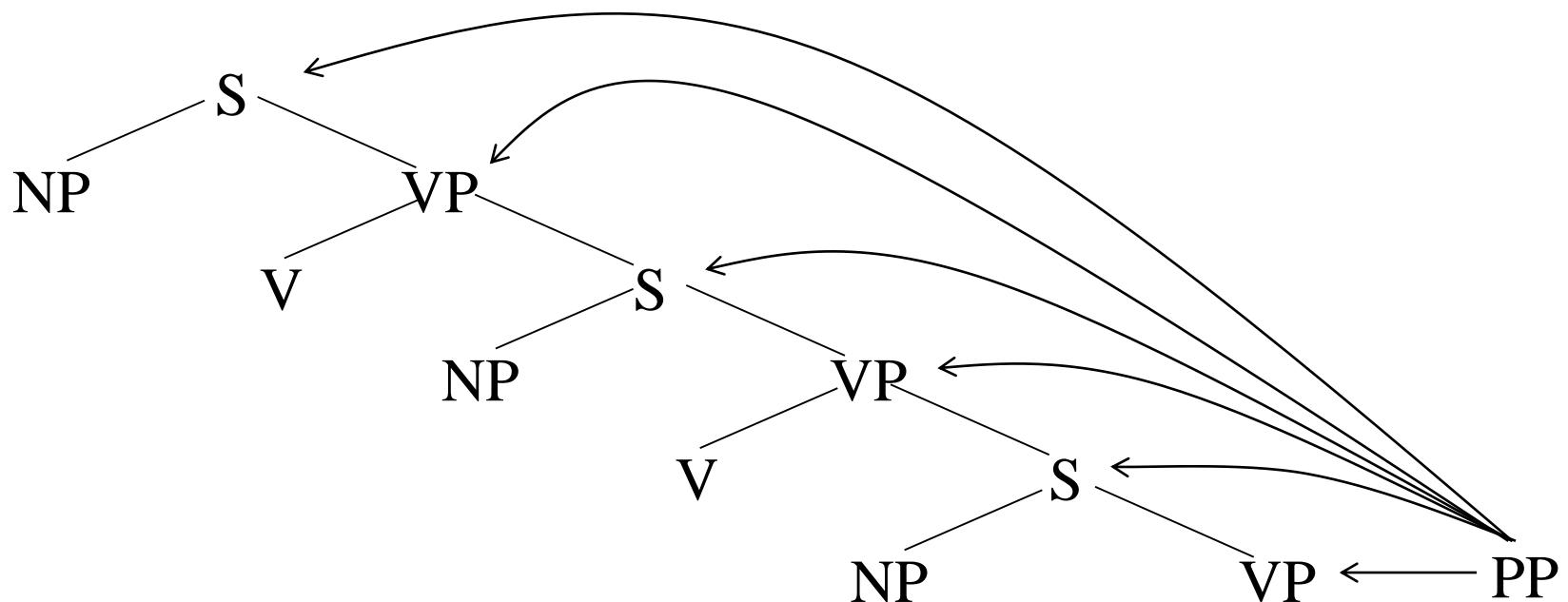
Combined lexical and structural ambiguity



Three major sources of structure ambiguity

- PP attachment,
- coordination, and
- noun-noun compounding.

PP can attach to sentences, verb phrases, noun phrases, and adjectival phrases.



Coordination

Mayumi and Hans or Beryl left early.

$$\begin{array}{ll} P \text{ and } Q \text{ or } R \equiv P \text{ and } (Q \text{ or } R) & (\text{alternative 1}) \\ & \\ (P \text{ and } Q) \text{ or } R & (\text{alternative 2}) \end{array}$$

widget (工具) hammer (鎚)

-- widget used as hammer, (當鎚使用的工具)

-- hammer for hitting widgets, (敲打小器械的鎚子)

-- hammer used by widgets, ...

no structural ambiguity

town widget hammer

((town widget) hammer)

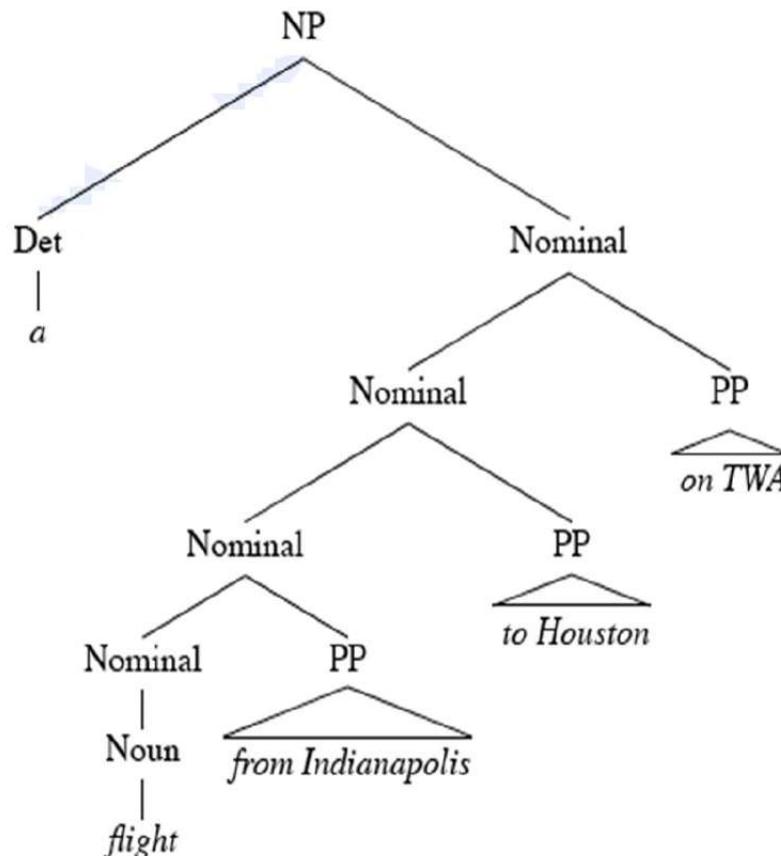
(town (widget hammer)) \leftarrow *structural ambiguity*

Shared Sub-Problems

- No matter what kind of search (top-down or bottom-up or mixed) that we choose.
 - We don't want to redo work we've already done.
 - Unfortunately, naïve backtracking will lead to duplicated work.

Shared Sub-Problems

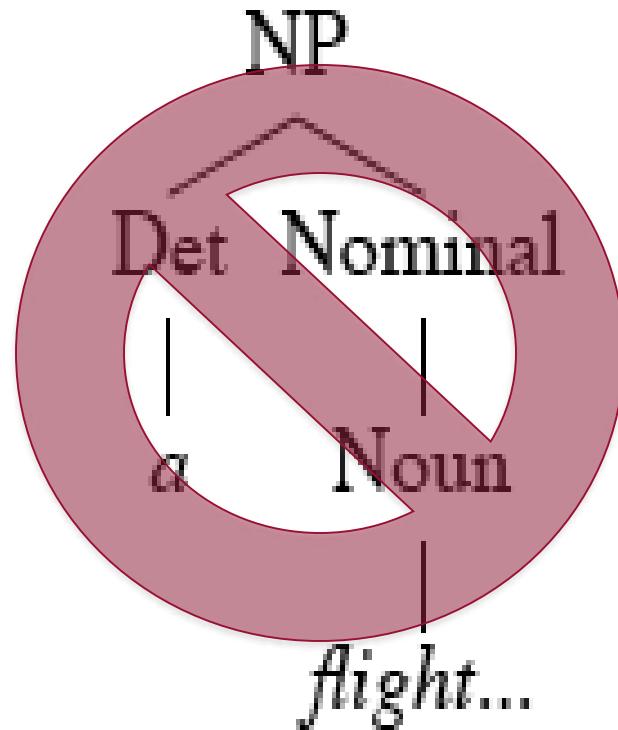
- Consider
 - A flight from Indianapolis to Houston on TWA



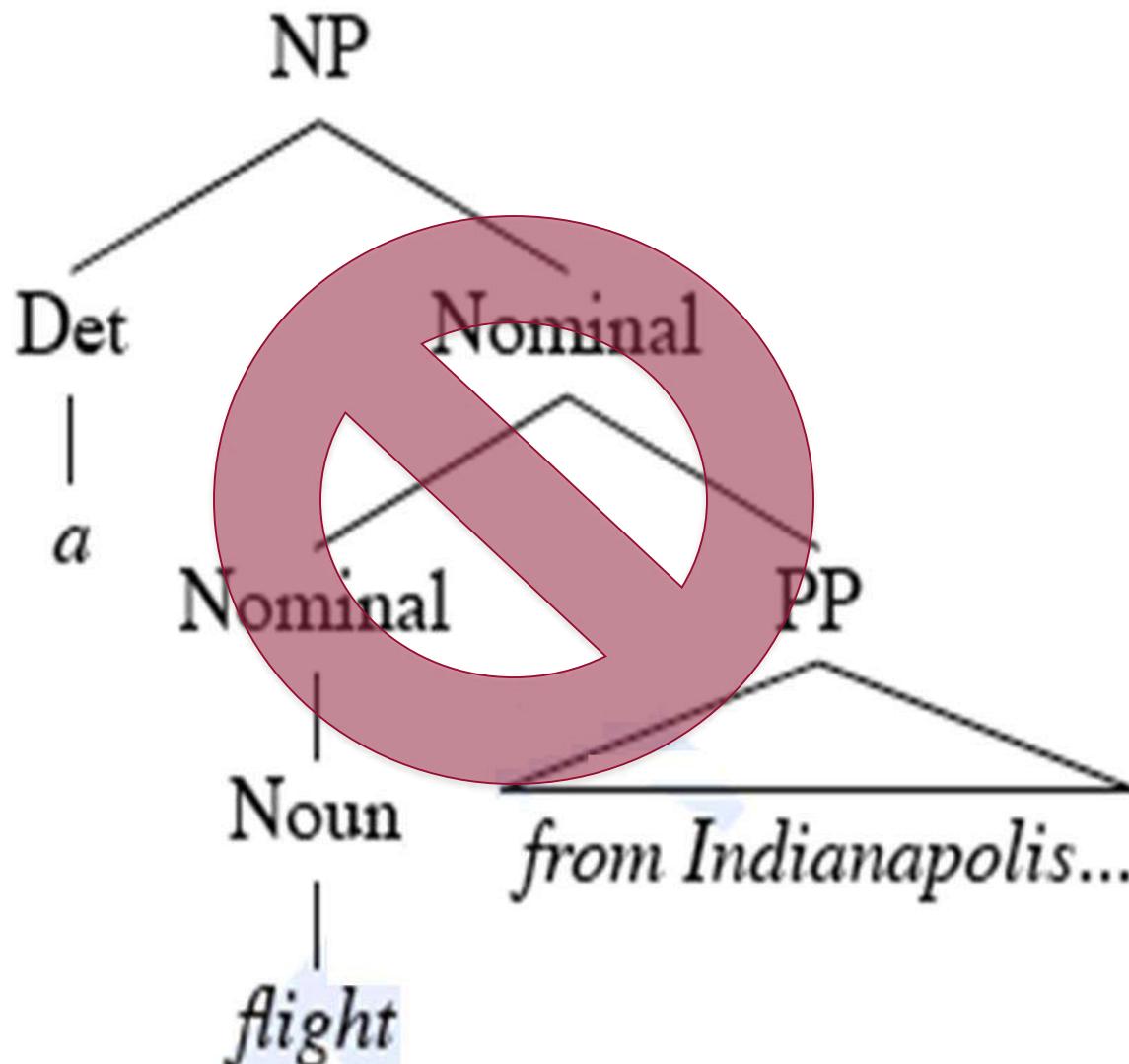
Shared Sub-Problems

- Assume a top-down parse making choices among the various Nominal rules.
- In particular, between these two
 - Nominal -> Noun
 - Nominal -> Nominal PP
- Statically choosing the rules in this order leads to the following bad behavior...

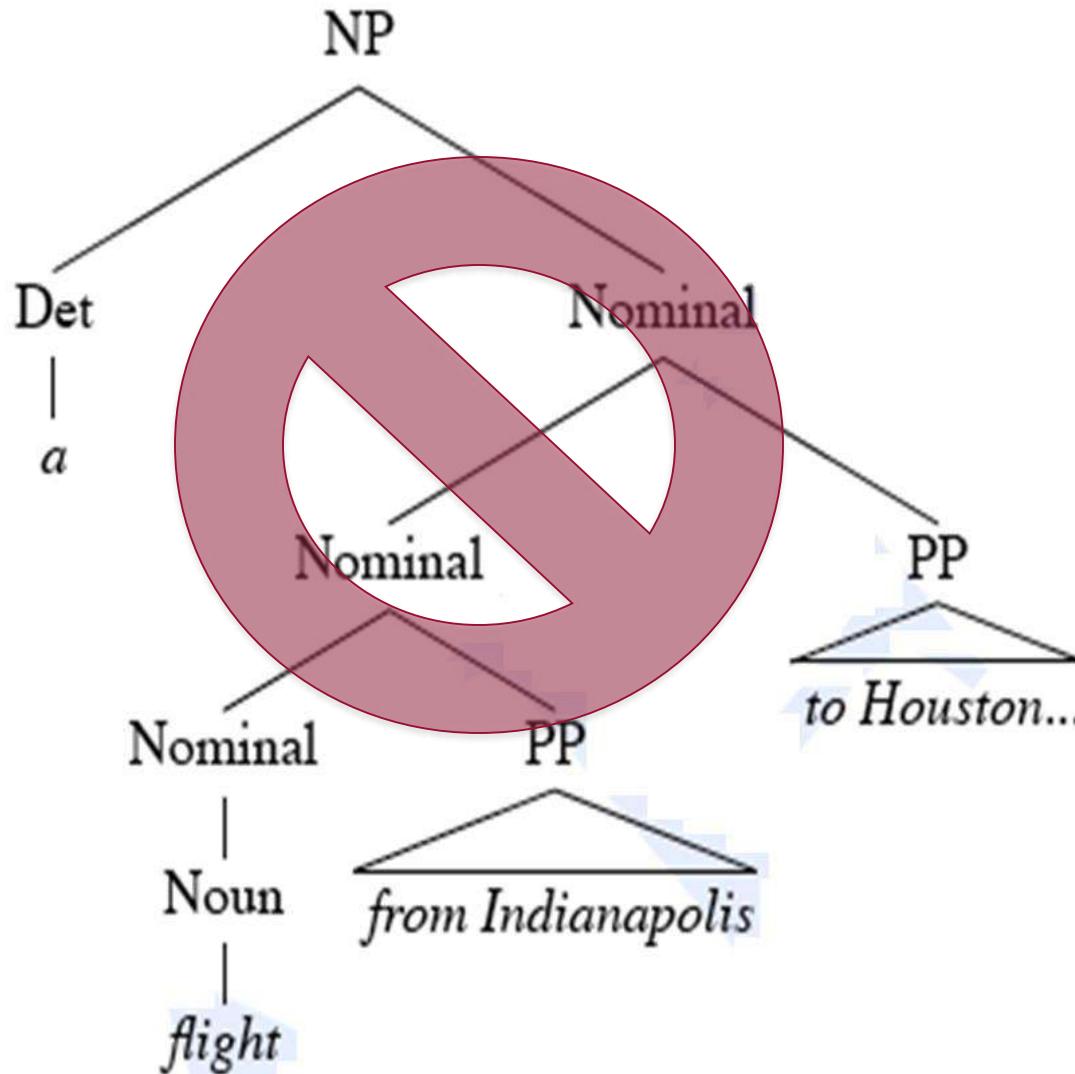
Shared Sub-Problems



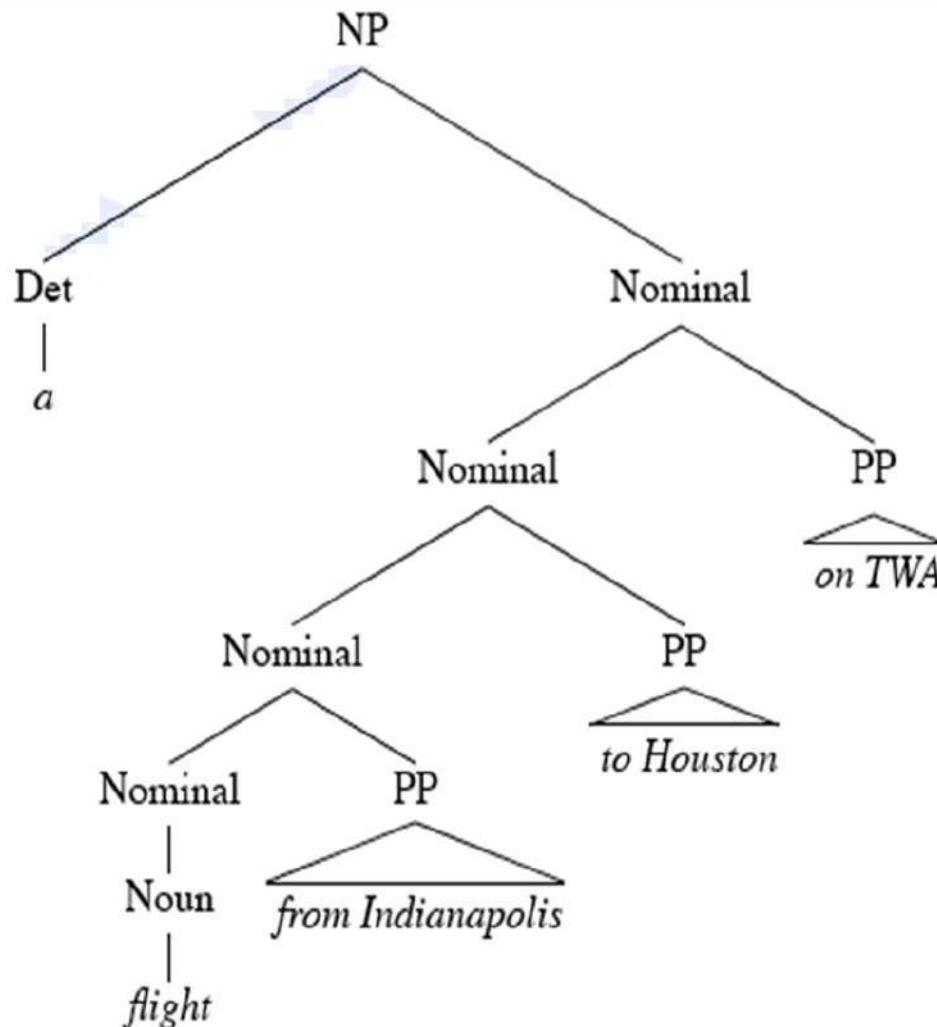
Shared Sub-Problems



Shared Sub-Problems



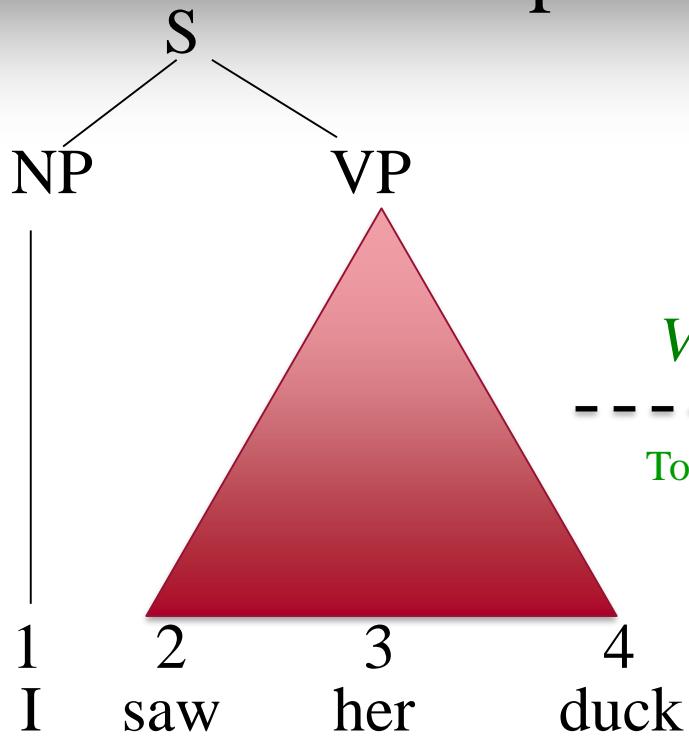
Shared Sub-Problems



Dynamic Programming

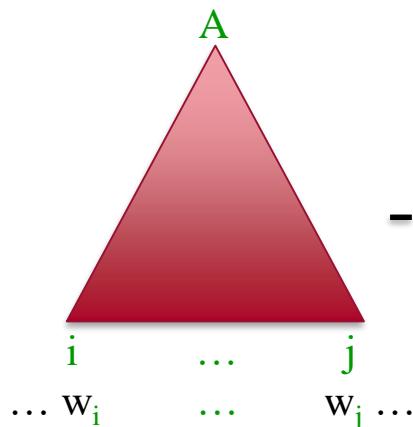
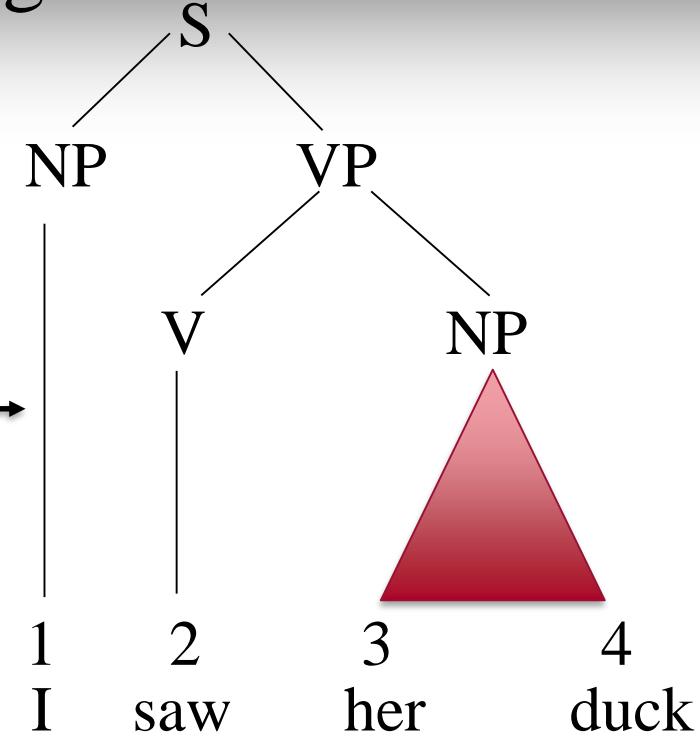
- DP search methods fill tables with partial results and thereby
 - Avoid doing avoidable repeated work
 - Solve exponential problems in polynomial time
 - Efficiently store ambiguous structures with shared sub-parts.
- We'll cover two approaches that roughly correspond to top-down and bottom-up approaches.
 - CKY
 - Earley

Top-Down Parsing



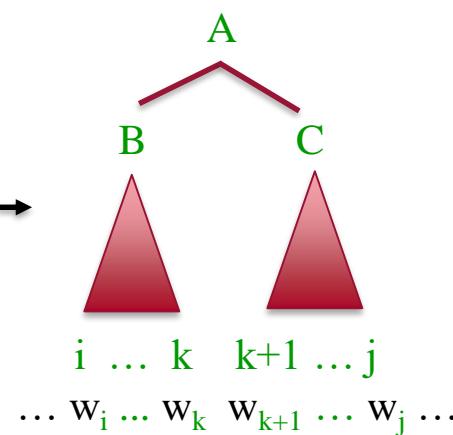
$VP \rightarrow VNP$

Top-down parsing



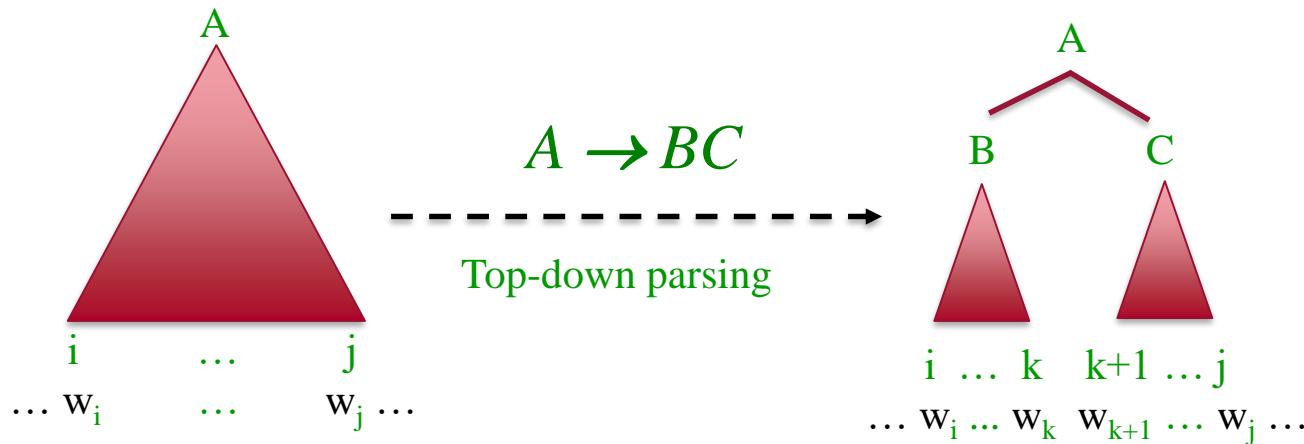
$A \rightarrow BC$

Top-down parsing



CKY Parsing

- First we'll limit our grammar to epsilon-free, binary rules
- Consider the rule $A \rightarrow BC$
 - If there is an A somewhere in the input generated by this rule then there must be a B followed by a C in the input.
 - If the A spans from i to j in the input then there must be some k st. $i < k < j$
 - In other words, the B splits from the C someplace after the i and before the j .



Problem

- What if your grammar isn't binary?
 - As in the case of the TreeBank grammar?
- Convert it to binary... any arbitrary CFG can be rewritten into Chomsky-Normal Form automatically.
- What does this mean?
 - The resulting grammar accepts (and rejects) the same set of strings as the original grammar.
 - **But** the resulting derivations (trees) are different.

Sample L1 Grammar

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that this a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book flight meal money$
$S \rightarrow VP$	$Verb \rightarrow book include prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I she me$
$NP \rightarrow Proper-Noun$	$Proper-Noun \rightarrow Houston NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from to on near through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

CNF Conversion

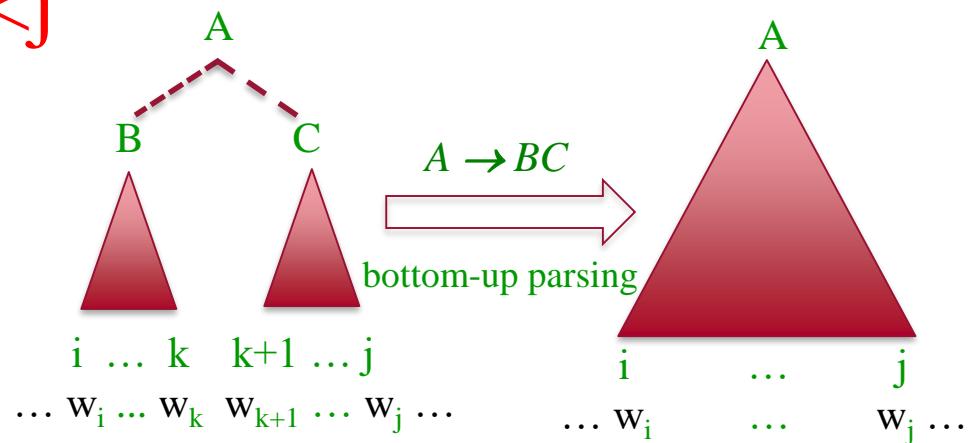
\mathcal{L}_1 Grammar	\mathcal{L}_1 in CNF
$S \rightarrow NP\ VP$	$S \rightarrow NP\ VP$
$S \rightarrow Aux\ NP\ VP$	$S \rightarrow X1\ VP$
	$X1 \rightarrow Aux\ NP$
$S \rightarrow VP$	$S \rightarrow book \mid include \mid prefer$
	$S \rightarrow Verb\ NP$
	$S \rightarrow X2\ PP$
	$S \rightarrow Verb\ PP$
	$S \rightarrow VP\ PP$
$NP \rightarrow Pronoun$	$NP \rightarrow I \mid she \mid me$
$NP \rightarrow Proper-Noun$	$NP \rightarrow TWA \mid Houston$
$NP \rightarrow Det\ Nominal$	$NP \rightarrow Det\ Nominal$
$Nominal \rightarrow Noun$	$Nominal \rightarrow book \mid flight \mid meal \mid money$
$Nominal \rightarrow Nominal\ Noun$	$Nominal \rightarrow Nominal\ Noun$
$Nominal \rightarrow Nominal\ PP$	$Nominal \rightarrow Nominal\ PP$
$VP \rightarrow Verb$	$VP \rightarrow book \mid include \mid prefer$
$VP \rightarrow Verb\ NP$	$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ PP$	$VP \rightarrow X2\ PP$
	$X2 \rightarrow Verb\ NP$
$VP \rightarrow Verb\ PP$	$VP \rightarrow Verb\ PP$
$VP \rightarrow VP\ PP$	$VP \rightarrow VP\ PP$
$PP \rightarrow Preposition\ NP$	$PP \rightarrow Preposition\ NP$

CKY

- Let's build *a table* so that an A spanning from i to j in the input is placed in cell $[i,j]$ in the table.
 - So a non-terminal spanning an entire string will sit in cell $[0, n]$
 - Hopefully it will be an S
- Now we know that the parts of the A must go from i to k and from k to j , for some k

CKY

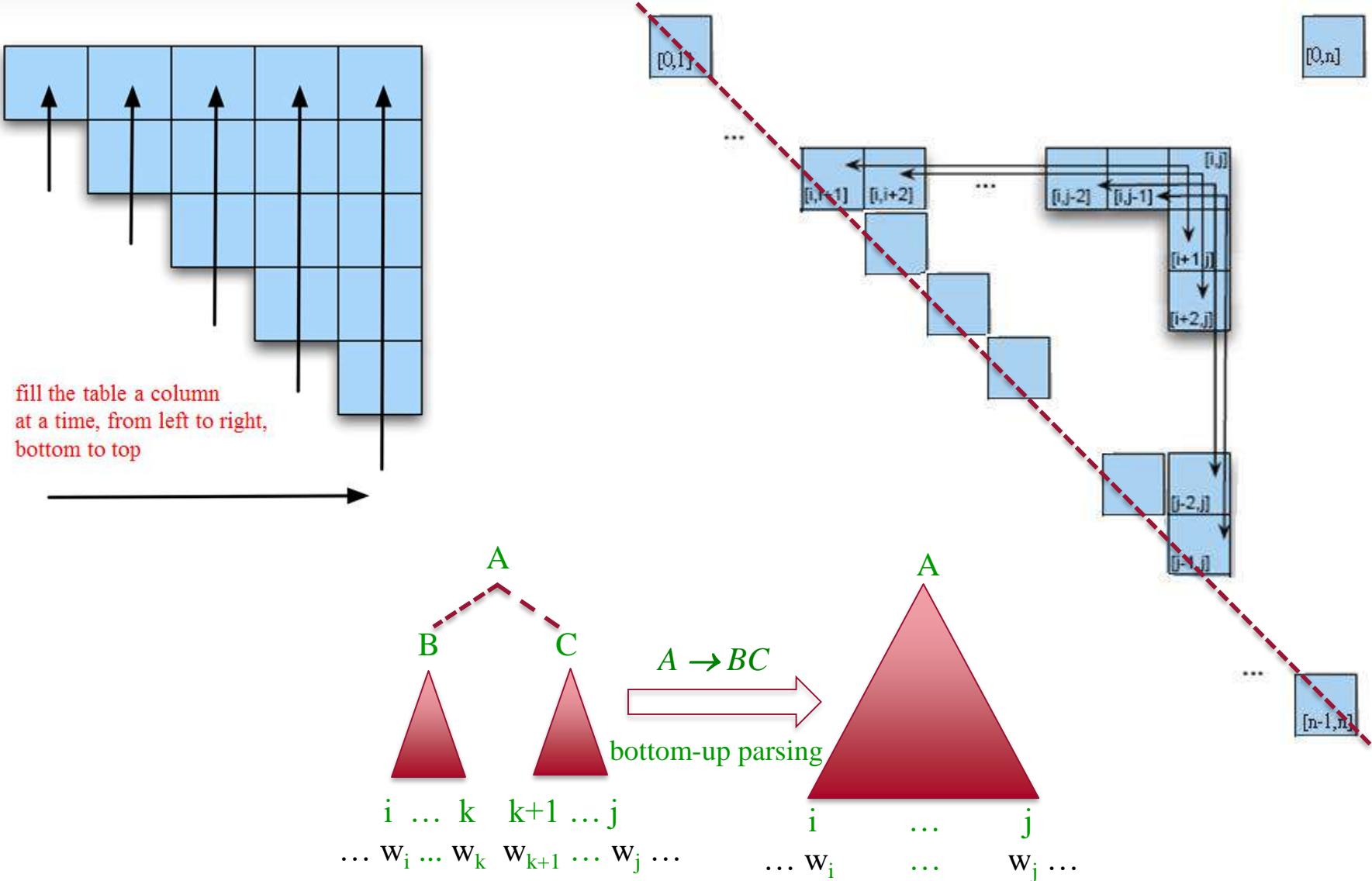
- Meaning that for a rule like $A \rightarrow B C$ we should look for a B in $[i,k]$ and a C in $[k,j]$.
- In other words, if we think there might be an A spanning i,j in the input... AND
 $A \rightarrow B C$ is a rule in the grammar THEN
- There must be a B in $[i,k]$ and a C in $[k,j]$ for some k such that $i < k < j$



CKY

- So to fill the table loop over the cell[i,j] values in some systematic way
 - Then for each cell, loop over the appropriate k values to search for things to add.

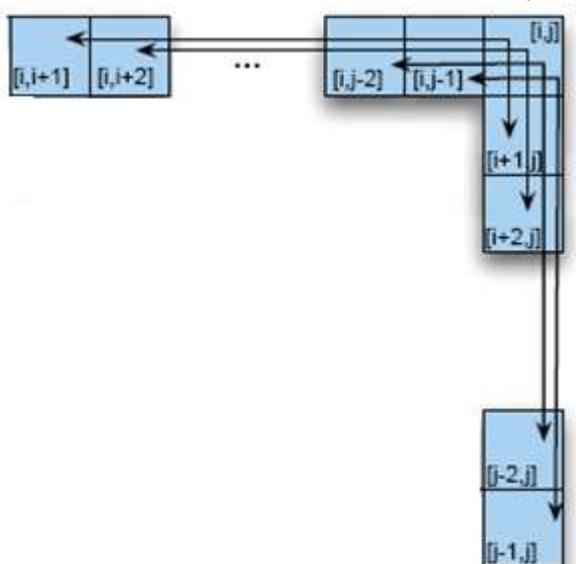
CKY Table



CKY Algorithm

function CKY-PARSE(*words*, *grammar*) **returns** *table*

```
for  $j \leftarrow \text{from } 1 \text{ to } \text{LENGTH}(\text{words}) \text{ do}$            j-th column  
     $\text{table}[j-1, j] \leftarrow \{A \mid A \rightarrow \text{words}[j] \in \text{grammar}\}$   
    for  $i \leftarrow \text{from } j-2 \text{ downto } 0 \text{ do}$            (j-2)-th row to 0-th row  
        for  $k \leftarrow i+1 \text{ to } j-1 \text{ do}$   
             $\text{table}[i, j] \leftarrow \text{table}[i, j] \cup$   
            [0,j]           { $A \mid A \rightarrow BC \in \text{grammar},$   
            .  
            B \in \text{table}[i, k],  
            C \in \text{table}[k, j]}
```

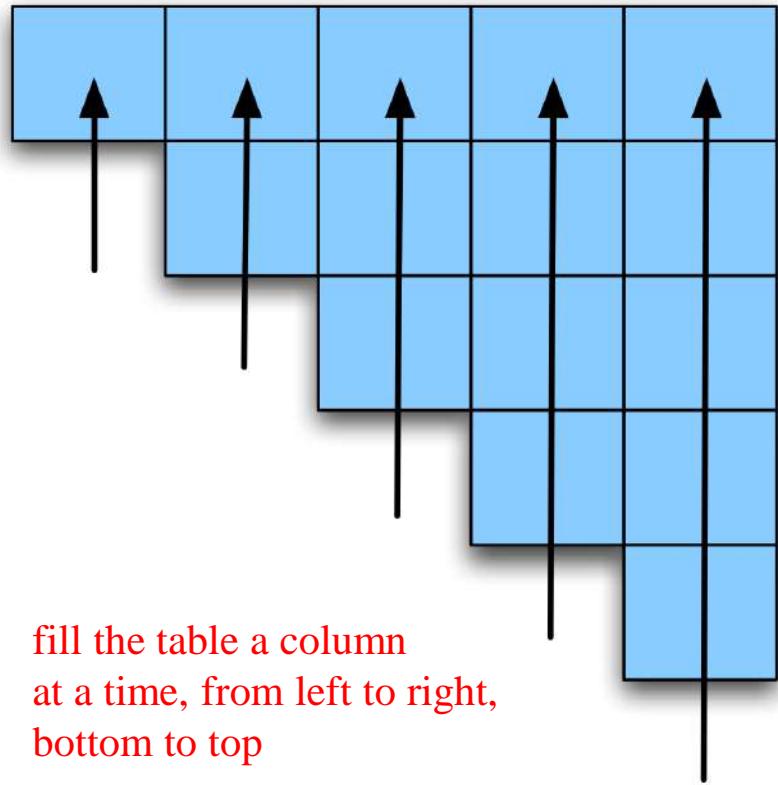


What's the complexity of this?

Example

Book the flight through Houston

S, VP, Verb Nominal, Noun [0,1]		S,VP,X2 [0,3]		S,VP,X2 [0,5]
Det [1,2]	NP [1,3]		[1,4]	NP [1,5]
	Nominal, Noun [2,3]		[2,4]	Nominal [2,5]
		Prep [3,4]	PP [3,5]	
			NP, Proper- Noun [4,5]	



Example

Book the flight through Houston

S, VP, Verb, Nominal, Noun [0,1]		S,VP,X2 [0,3]		
	Det [1,2]	NP [1,3]		
		Nominal, Noun [2,3]		Nominal [2,5]
			Prep [3,4]	
				NP, Proper- Noun [4,5]

Filling column 5

Example

- Filling column 5 corresponds to processing word 5, which is *Houston*.
 - So j is 5.
 - So i goes from 3 to 0 (3,2,1,0)

```
function CKY-PARSE(words, grammar) returns table
    for  $j \leftarrow \text{from } 1 \text{ to } \text{LENGTH}(\text{words})$  do
        table[ $j - 1, j$ ]  $\leftarrow \{A \mid A \rightarrow \text{words}[j] \in \text{grammar}\}$ 
    for  $i \leftarrow \text{from } j - 2 \text{ downto } 0$  do
        for  $k \leftarrow i + 1 \text{ to } j - 1$  do
            table[ $i, j$ ]  $\leftarrow \text{table}[i, j] \cup$ 
             $\{A \mid A \rightarrow BC \in \text{grammar},$ 
             $B \in \text{table}[i, k],$ 
             $C \in \text{table}[k, j]\}$ 
```

Example

<i>Book</i>	<i>the</i>	<i>flight</i>	<i>through</i>	<i>Houston</i>
S, VP, Verb, Nominal, Noun [0,1]		S,VP,X2 [0,3]		
	Det [1,2]	NP [1,3]		NP [1,5]
		Nominal, Noun [2,3]		
			Prep ← [3,4]	PP ↓ [3,5]
				NP, Proper- Noun [4,5]

Example

<i>Book</i>	<i>the</i>	<i>flight</i>	<i>through</i>	<i>Houston</i>
S, VP, Verb, Nominal, Noun [0,1]		S,VP,X2 [0,3]		
	Det [1,2]	NP [1,3]		NP [1,5]
		Nominal, Noun [2,3]		Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper- Noun [4,5]

The diagram illustrates a 5x5 grid of part-of-speech (POS) tags for the sentence "Book the flight through Houston". The grid structure is as follows:

- Row 1:** Book, the, flight, through, Houston
- Row 2:** S, VP, Verb, Nominal, Noun [0,1]
- Row 3:** Det [1,2], NP [1,3]
- Row 4:** Nominal, Noun [2,3]
- Row 5:** Prep [3,4], PP [3,5]
- Row 6:** NP, Proper-Noun [4,5]

Annotations with arrows point to specific words in the sentence:

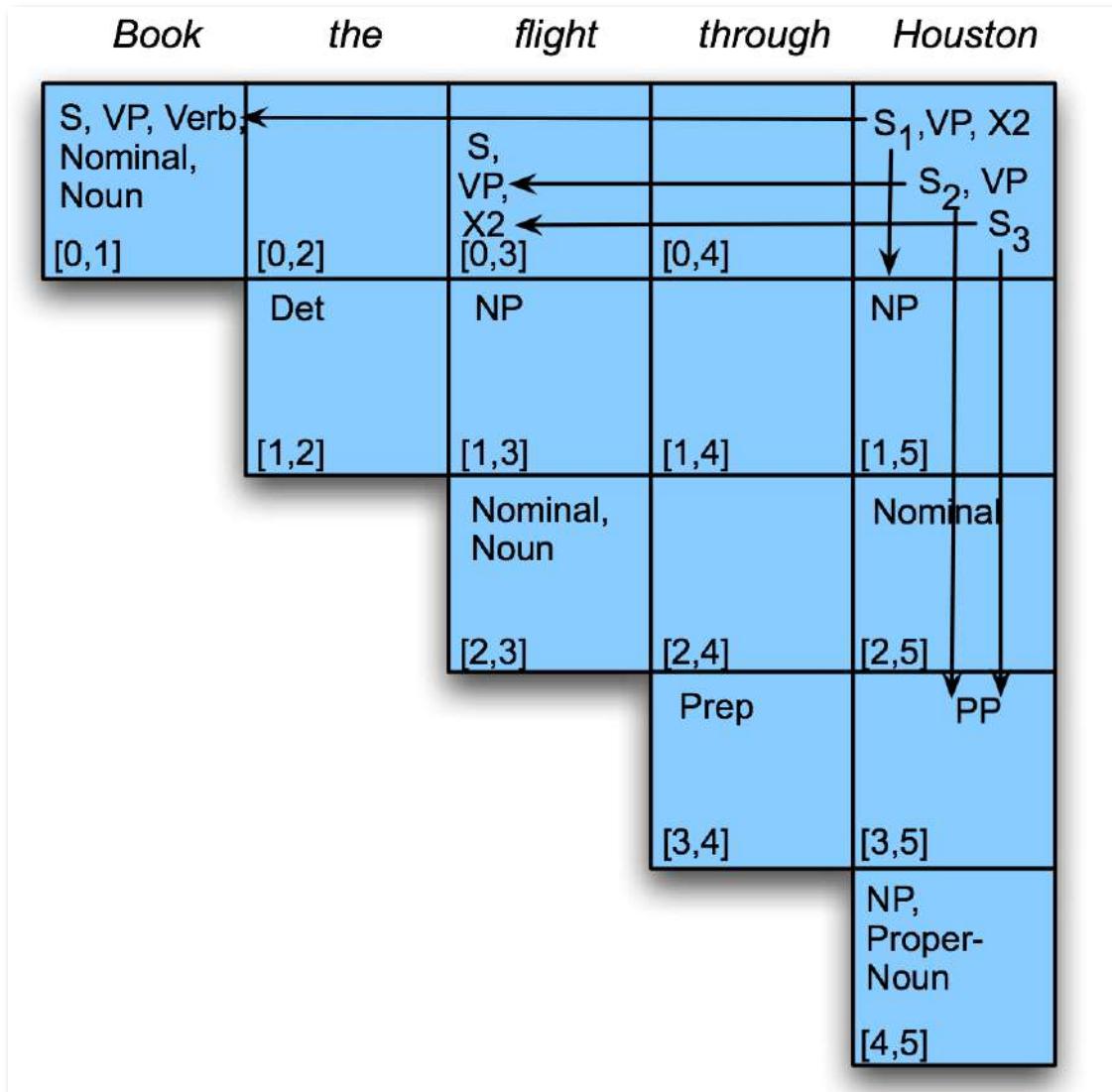
- An arrow points from the "Nominal" tag in Row 4 to the word "flight" in the sentence.
- An arrow points from the "Nominal" tag in Row 4 to the word "Houston" in the sentence.

Example

Book the flight through Houston

S, VP, Verb, Nominal, Noun [0,1]		S,VP,X2 [0,3]	[0,4]	[0,5]
Det ←	NP			NP
[1,2]	[1,3]	[1,4]	[1,5]	
Nominal, Noun [2,3]			Nominal [2,5]	
	Prep [3,4]	PP [3,5]		
			NP, Proper- Noun [4,5]	

Example



Example

- Since there's an S in $[0,5]$ we have a valid parse.
- Are we done? We are sort of left something out of the algorithm

```
function CKY-PARSE(words, grammar) returns table
    for  $j \leftarrow$  from 1 to LENGTH(words) do
        table[ $j - 1, j$ ]  $\leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$ 
    for  $i \leftarrow$  from  $j - 2$  downto 0 do
        for  $k \leftarrow i + 1$  to  $j - 1$  do
            table[ $i, j$ ]  $\leftarrow$  table[ $i, j$ ]  $\cup$ 
                 $\{A \mid A \rightarrow BC \in grammar,$ 
                 $B \in table[i, k],$ 
                 $C \in table[k, j]\}$ 
```

CKY Notes

- Since it's bottom up, CKY populates the table with a lot of phantom constituents.
 - Segments that by themselves are constituents but cannot really occur in the context in which they are being suggested.
 - To avoid this we can switch to a top-down control strategy
 - Or we can add some kind of filtering that blocks constituents where they can not happen in a final analysis.

CKY Notes

- We arranged the loops to fill the table a column at a time, from left to right, bottom to top.
 - This assures us that whenever we're filling a cell, the parts needed to fill it are already in the table (to the left and below)
 - It's somewhat natural in that it processes the input a left to right a word at a time
 - Known as online

Full Syntactic Parsing

- Probably necessary for deep semantic analysis of texts
- Probably not practical for many applications (given typical resources)
 - $O(n^3)$ for straight parsing
 - $O(n^5)$ for probabilistic versions
 - Too slow for applications that need to process texts in real time (search engines)
 - Or that need to deal with large volumes of new material over short periods of time

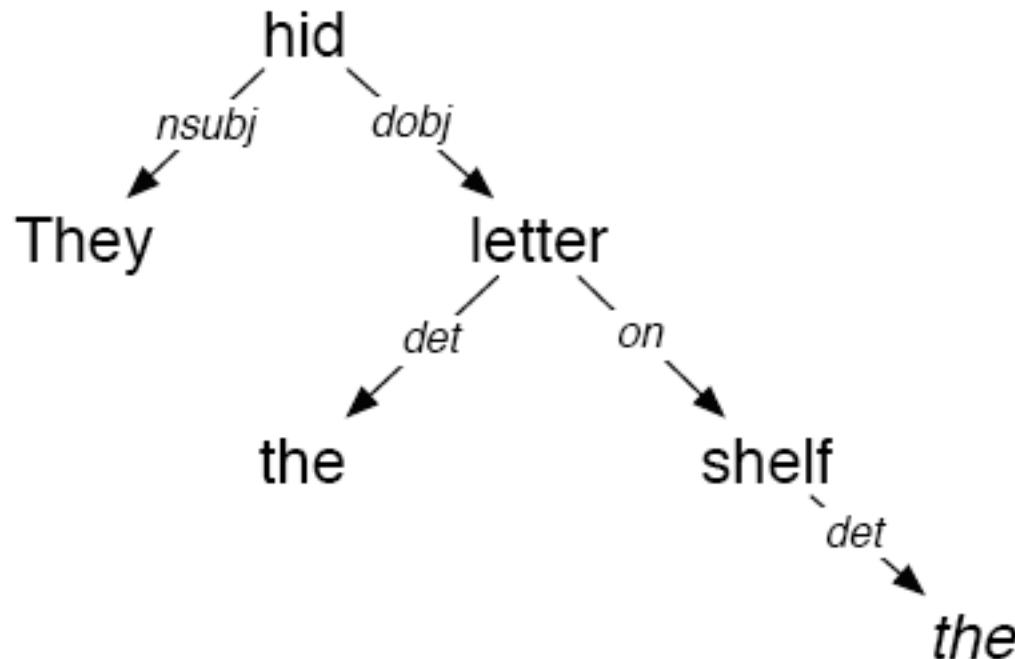
Two Alternatives

- Dependency parsing
 - Change the underlying grammar formalism
- Partial parsing
 - Approximate phrase-structure parsing with finite-state and statistical approaches
- Both of these approaches give up something (syntactic structure) in return for more robust and efficient parsing

Dependency Grammars

- It turns out you can get a lot done with just binary relations among the words in an utterance
- In a **dependency grammar** framework, a parse is a tree where
 - The nodes stand for the words in an utterance
 - The links between the words represent dependency relations between pairs of words.
 - Relations may be typed (labeled), or not.

Dependency Parse



They hid the letter on the shelf

Partial Parsing

- For many applications you don't really need a full-blown syntactic parse. You just need a good idea of where the **base syntactic units** are.
 - Often referred to as chunks.
- For example, if you're interested in locating all the people, places and organizations in a text it can be useful to know where all the NPs are
 - Because that's where you'll find the people, places and things

Examples

[*NP* The morning flight] [*PP* from] [*NP* Denver] [*VP* has arrived.]

[*NP* a flight] [*PP* from] [*NP* Indianapolis][*PP* to][*NP* Houston][*PP* on][*NP* TWA]

[*NP* The morning flight] from [*NP* Denver] has arrived.

- The first two are examples of full partial parsing or chunking. All of the elements in the text are part of a chunk. And the chunks are non-overlapping.
- Note how the second example has no hierarchical structure.
- The last example illustrates base-NP chunking. Ignore anything that isn't in the kind of chunk you're looking for.

Machine Learning-Based Approaches to Chunking

- Train a chunker by using annotated data as a training set

- One of sequence labeling
- Treat chunking as a tagging task
- IOB tagging

The morning flight from Denver has arrived

B_NP I_NP I_NP B_PP B_NP B_VP I_VP

Tagging set
B_NP, I_NP
B_PP, I_PP
B_VP, I_VP
O

- Only the base-NPs tagged

The morning flight from Denver has arrived.

B_NP I_NP I_NP O B_NP O O

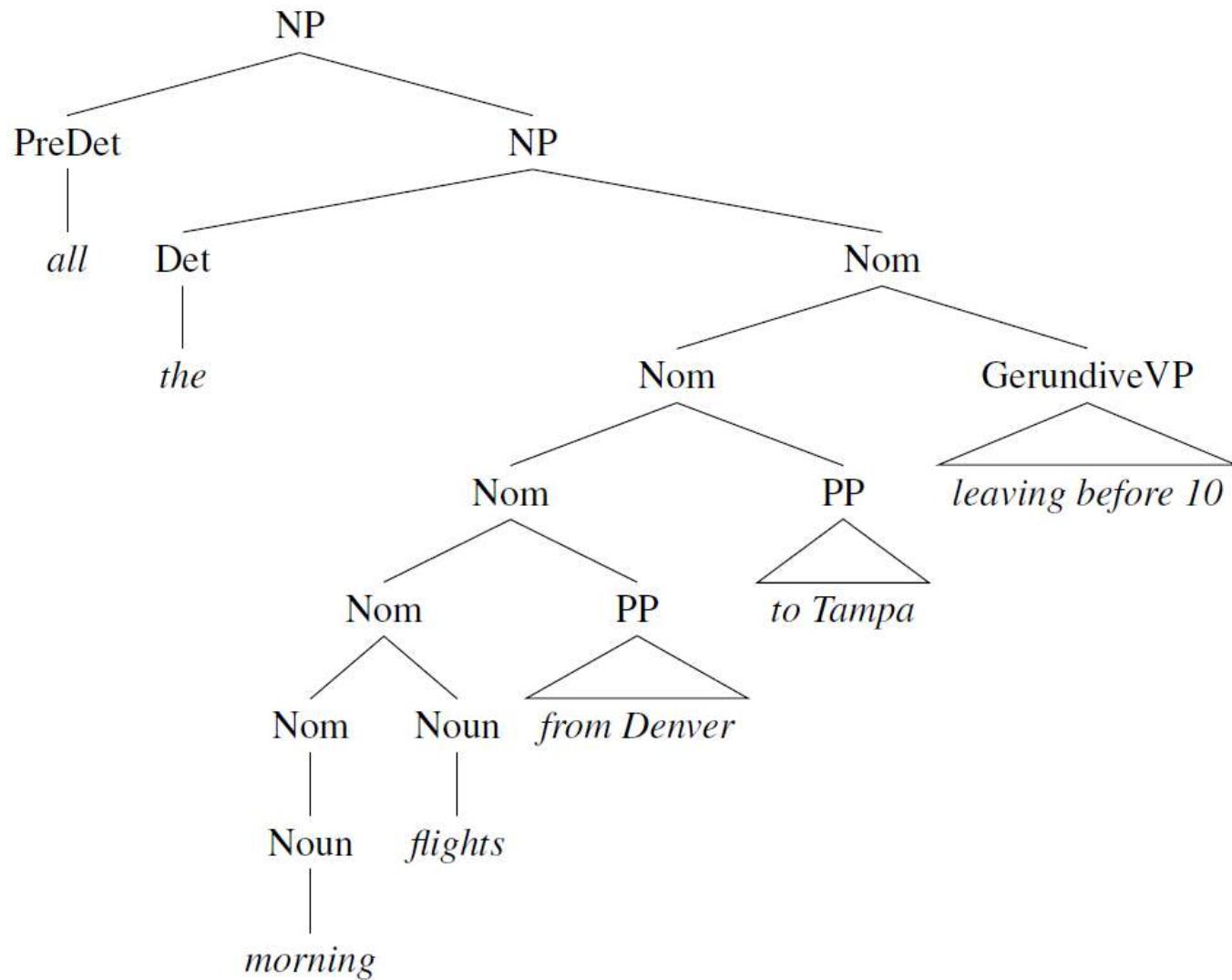
Tagging set
B_NP, I_NP
O

Training a Chunker

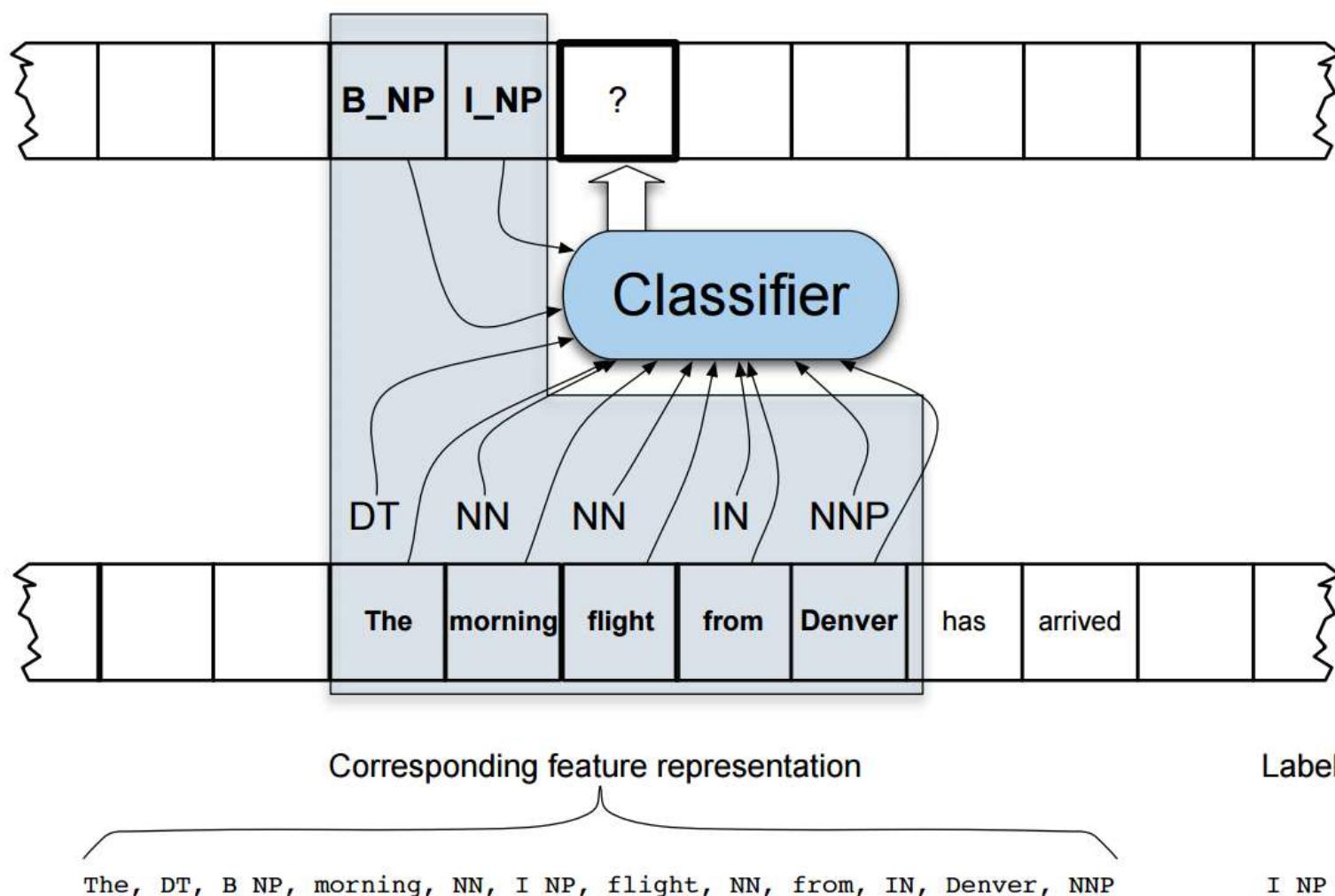
- Train a classifier to label each word of an input sentence with one of the IOB tags from the tagset
 - the phrases of interest delimited
 - marked with the appropriate category
- Where is the training corpus?
 - Annotate a representative corpus?
 - Find such data for chunking from an existing treebank

Construct Training Set from Treebank

- Find the interesting phrases
 - To know the appropriate non-terminal names in the corpus
- Find chunk boundaries
 - To find the head and then including the material to the left of the head



A Sequential-Classifier-based Approach to Chunking



Chunking-System Evaluations

- Precision, Recall, F-measure

$$\text{Precision: } = \frac{\text{Number of correct chunks given by system}}{\text{Total number of chunks given by system}}$$

$$\text{Recall: } = \frac{\text{Number of correct chunks given by system}}{\text{Total number of actual chunks in the text}}$$

$$F_1 = \frac{2PR}{P+R}$$

- Source of errors

- part-of-speech tagging accuracy
- inconsistencies in the training data
- difficulty resolving ambiguities involving conjunctions

[*NP* Late arrivals and departures] are commonplace during winter.

[*NP* Late arrivals] and [*NP* cancellations] are commonplace during winter.

Source of This Lecture

- Chapter 12, Speech and Language Processing
- Chapter 13, Speech and Language Processing

Lecture 8. Statistical Parsing

Probabilistic Context-Free Grammar (PCFG)

N a set of **non-terminal symbols** (or **variables**)

Σ a set of **terminal symbols** (disjoint from N)

R a set of **rules** or productions, each of the form $A \rightarrow \beta [p]$,
where A is a non-terminal,
 β is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$,
and p is a number between 0 and 1 expressing $P(\beta | A)$

S a designated **start symbol**

PCFG

Grammar		Lexicon
$S \rightarrow NP VP$	[.80]	$Det \rightarrow that [.10] \mid a [.30] \mid the [.60]$
$S \rightarrow Aux NP VP$	[.15]	$Noun \rightarrow book [.10] \mid flight [.30]$
$S \rightarrow VP$	[.05]	$\quad \mid meal [.015] \mid money [.05]$
$NP \rightarrow Pronoun$	[.35]	$\quad \mid flight [.40] \mid dinner [.10]$
$NP \rightarrow Proper-Noun$	[.30]	$Verb \rightarrow book [.30] \mid include [.30]$
$NP \rightarrow Det Nominal$	[.20]	$\quad \mid prefer [.40]$
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I [.40] \mid she [.05]$
$Nominal \rightarrow Noun$	[.75]	$\quad \mid me [.15] \mid you [.40]$
$Nominal \rightarrow Nominal Noun$	[.20]	$Proper-Noun \rightarrow Houston [.60]$
$Nominal \rightarrow Nominal PP$	[.05]	$\quad \mid NWA [.40]$
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does [.60] \mid can [40]$
$VP \rightarrow Verb NP$	[.20]	$Preposition \rightarrow from [.30] \mid to [.30]$
$VP \rightarrow Verb NP PP$	[.10]	$\quad \mid on [.20] \mid near [.15]$
$VP \rightarrow Verb PP$	[.15]	$\quad \mid through [.05]$
$VP \rightarrow Verb NP NP$	[.05]	
$VP \rightarrow VP PP$	[.15]	
$PP \rightarrow Preposition NP$	[1.0]	

PCFGs for Disambiguation

- The probability of a particular parse T is defined as the **product of the probabilities of all the n rules** used to expand each of the n non-terminal nodes in the parse tree T , where each rule i can be expressed as $\text{LHS}_i \rightarrow \text{RHS}_i$

$$P(T, S) = \prod_{i=1}^n P(\text{RHS}_i | \text{LHS}_i)$$

$$P(T, S) = P(T)P(S|T)$$

A parse tree includes all the words of the sentence, $P(S|T)$ is 1

$$P(T, S) = P(T)P(S|T) = P(T)$$

<pre> graph TD S1[S] --- VP1[VP] VP1 --- Verb1[Verb] VP1 --- NP1[NP] Verb1 --- Book1[Book] NP1 --- Det1[Det] NP1 --- Nominal1[Nominal] Det1 --- the1["the"] Nominal1 --- Nominal2[Nominal] Nominal1 --- Noun1[Noun] Nominal2 --- flight1["flight"] Noun1 --- dinner1["dinner"] dinner1 --- dinner2["dinner"] </pre>	<pre> graph TD S2[S] --- VP2[VP] VP2 --- Verb2[Verb] VP2 --- NP2[NP] VP2 --- NP3[NP] Verb2 --- Book2[Book] NP2 --- Det2[Det] NP2 --- Nominal3[Nominal] Det2 --- the2["the"] NP3 --- Nominal4[Nominal] NP3 --- Noun2[Noun] Nominal4 --- dinner2["dinner"] Noun2 --- flight2["flight"] flight2 --- flight3["flight"] </pre>																																												
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Rules</th> <th style="text-align: center;">P</th> <th style="text-align: center;">Rules</th> <th style="text-align: center;">P</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">S → VP</td> <td style="text-align: center;">.05</td> <td style="text-align: center;">S → VP</td> <td style="text-align: center;">.05</td> </tr> <tr> <td style="text-align: center;">VP → Verb NP</td> <td style="text-align: center;">.20</td> <td style="text-align: center;">VP → Verb NP NP</td> <td style="text-align: center;">.10</td> </tr> <tr> <td style="text-align: center;">NP → Det Nominal</td> <td style="text-align: center;">.20</td> <td style="text-align: center;">NP → Det Nominal</td> <td style="text-align: center;">.20</td> </tr> <tr> <td style="text-align: center;">Nominal → Nominal Noun</td> <td style="text-align: center;">.20</td> <td style="text-align: center;">NP → Nominal</td> <td style="text-align: center;">.15</td> </tr> <tr> <td style="text-align: center;">Nominal → Noun</td> <td style="text-align: center;">.75</td> <td style="text-align: center;">Nominal → Noun</td> <td style="text-align: center;">.75</td> </tr> <tr> <td></td> <td></td> <td style="text-align: center;">Nominal → Noun</td> <td style="text-align: center;">.75</td> </tr> <tr> <td style="text-align: center;">Verb → book</td> <td style="text-align: center;">.30</td> <td style="text-align: center;">Verb → book</td> <td style="text-align: center;">.30</td> </tr> <tr> <td style="text-align: center;">Det → the</td> <td style="text-align: center;">.60</td> <td style="text-align: center;">Det → the</td> <td style="text-align: center;">.60</td> </tr> <tr> <td style="text-align: center;">Noun → dinner</td> <td style="text-align: center;">.10</td> <td style="text-align: center;">Noun → dinner</td> <td style="text-align: center;">.10</td> </tr> <tr> <td style="text-align: center;">Noun → flight</td> <td style="text-align: center;">.40</td> <td style="text-align: center;">Noun → flight</td> <td style="text-align: center;">.40</td> </tr> </tbody> </table>	Rules	P	Rules	P	S → VP	.05	S → VP	.05	VP → Verb NP	.20	VP → Verb NP NP	.10	NP → Det Nominal	.20	NP → Det Nominal	.20	Nominal → Nominal Noun	.20	NP → Nominal	.15	Nominal → Noun	.75	Nominal → Noun	.75			Nominal → Noun	.75	Verb → book	.30	Verb → book	.30	Det → the	.60	Det → the	.60	Noun → dinner	.10	Noun → dinner	.10	Noun → flight	.40	Noun → flight	.40	
Rules	P	Rules	P																																										
S → VP	.05	S → VP	.05																																										
VP → Verb NP	.20	VP → Verb NP NP	.10																																										
NP → Det Nominal	.20	NP → Det Nominal	.20																																										
Nominal → Nominal Noun	.20	NP → Nominal	.15																																										
Nominal → Noun	.75	Nominal → Noun	.75																																										
		Nominal → Noun	.75																																										
Verb → book	.30	Verb → book	.30																																										
Det → the	.60	Det → the	.60																																										
Noun → dinner	.10	Noun → dinner	.10																																										
Noun → flight	.40	Noun → flight	.40																																										

Figure 12.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the sensible meaning “Book a flight that serves dinner”, while the parse on the right corresponds to the nonsensical meaning “Book a flight on behalf of ‘the dinner’ ”.

$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$

$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

The Disambiguation Algorithm

$$\hat{T}(S) = \operatorname{argmax}_{T \text{ s.t. } S = \text{yield}(T)} P(T|S)$$

$$\hat{T}(S) = \operatorname{argmax}_{T \text{ s.t. } S = \text{yield}(T)} \frac{P(T, S)}{P(S)}$$

$$\hat{T}(S) = \operatorname{argmax}_{T \text{ s.t. } S = \text{yield}(T)} P(T, S)$$

$$\hat{T}(S) = \operatorname{argmax}_{T \text{ s.t. } S = \text{yield}(T)} P(T)$$

PCFGs for Language Modeling

- PCFGs can be applied both to **disambiguation in syntactic parsing** and **to word prediction** in language modeling.

$$P(w_i | w_1, w_2, \dots, w_{i-1}) = \frac{P(w_1, w_2, \dots, w_{i-1}, w_i)}{P(w_1, w_2, \dots, w_{i-1})}$$

- The **probability of an ambiguous sentence** is the **sum of the probabilities** of all the parse trees for the sentence

$$\begin{aligned} P(S) &= \sum_{T \text{ s.t. } S = \text{yield}(T)} P(T, S) \\ &= \sum_{T \text{ s.t. } S = \text{yield}(T)} P(T) \end{aligned}$$

Questions of PCFGs

- The probabilistic model
 - How to assign probabilities to parse trees
 - What is the probability of a sentence w_{1m} according to a grammar G : $P(w_{1m}/G)$?
- Parsing with probabilities (Decoding)
 - Given an input sentence and a model how can we efficiently find the best (or N best) tree(s) for that input
 - What is the most likely parse for a sentence: $\text{argmax}_t P(t/w_{1m}, G)$?
- Training the model (Learning)
 - How to acquire estimates for the probabilities specified by the model
 - How can we choose rule probabilities for the grammar G that maximize the probability of a sentence, $\text{argmax}_G P(w_{1m}/G)$?

Solutions

- Problem 1: Inside (Outside) Probabilities
- Problem 2: Dynamic Programming
- Problem 3: EM
 - Inside-Outside Algorithm

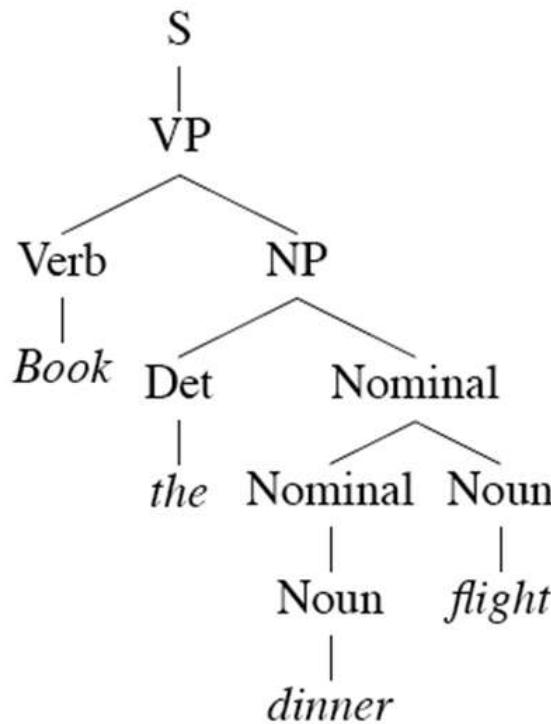
Parsing (Decoding)

- Get the best (most probable) parse for a given input
 1. Enumerate all the trees for a sentence
 2. Assign a probability to each using the model
 3. Return the argmax

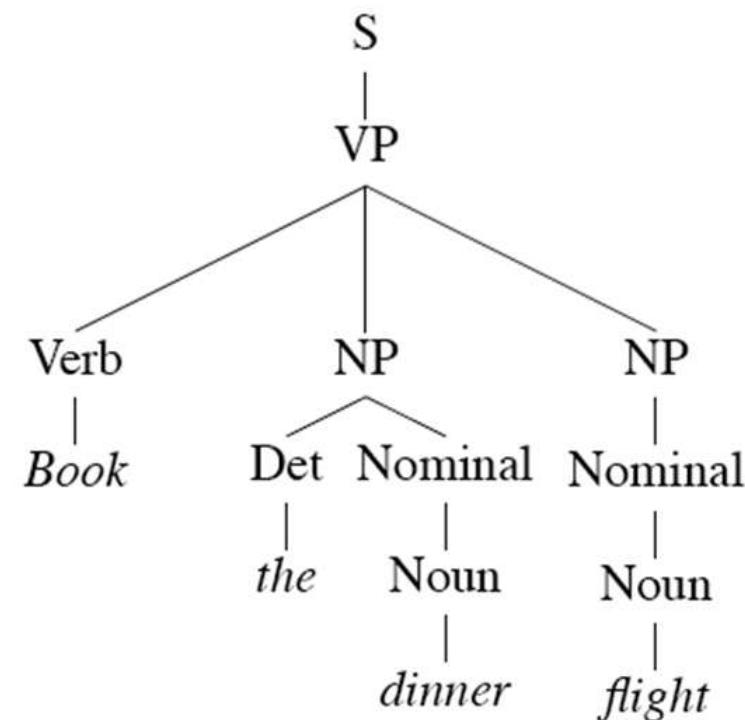
Example

- Consider...

- *Book the dinner flight*



Book a flight that serves dinner



Book a flight on behalf of 'the dinner'

Examples

- These trees consist of the following rules.

	Rules	P		Rules	P
S	$\rightarrow VP$.05	S	$\rightarrow VP$.05
VP	$\rightarrow Verb\ NP$.20	VP	$\rightarrow Verb\ NP\ NP$.10
NP	$\rightarrow Det\ Nominal$.20	NP	$\rightarrow Det\ Nominal$.20
Nominal	$\rightarrow Nominal\ Noun$.20	NP	$\rightarrow Nominal$.15
Nominal	$\rightarrow Noun$.75	Nominal	$\rightarrow Noun$.75
			Nominal	$\rightarrow Noun$.75
Verb	$\rightarrow book$.30	Verb	$\rightarrow book$.30
Det	$\rightarrow the$.60	Det	$\rightarrow the$.60
Noun	$\rightarrow dinner$.10	Noun	$\rightarrow dinner$.10
Noun	$\rightarrow flights$.40	Noun	$\rightarrow flights$.40

$$P(T_{left}) = .05 * .20 * .20 * .20 * .75 * .30 * .60 * .10 * .40 = \mathbf{2.2 \times 10^{-6}}$$

$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

Dynamic Programming

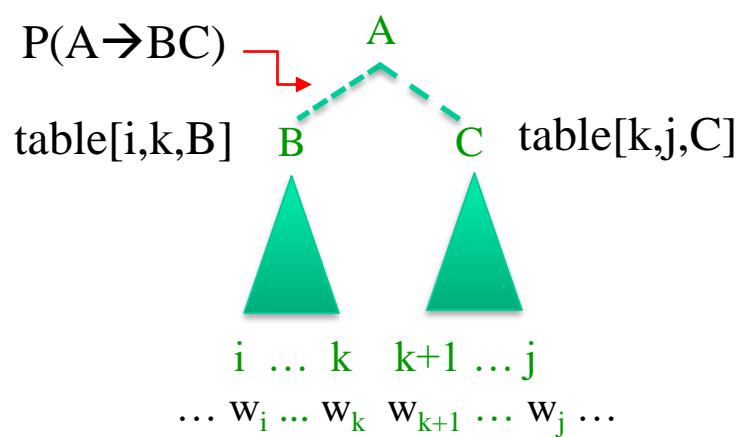
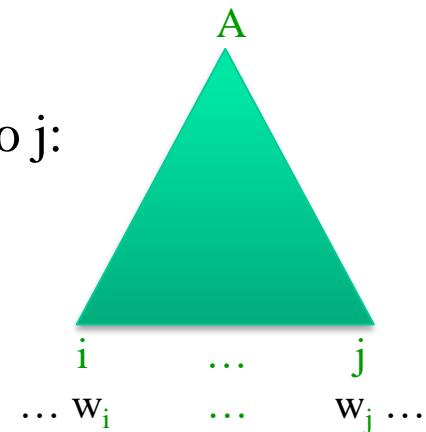
- With normal parsing we don't really want to do it that way.
- Instead, we need to exploit dynamic programming
 - For the parsing (as with CKY and Earley)
 - For computing the probabilities and returning the best parse (as with Viterbi and HMMs)

Restriction

- We only consider the case of *Chomsky Normal Form Grammars*, which only have unary and binary rules of the form:
 - $N^i \rightarrow N^j N^k$
 - $N^i \rightarrow w^j$
- The parameters of a PCFG in Chomsky Normal Form are:
 - $P(N^j \rightarrow N^r N^s | G)$, an n^3 matrix of parameters
 - $P(N^j \rightarrow w^k | G)$, nV parameters
(where n is the number of nonterminals and V is the number of terminals)
- $\sum_{r,s} P(N^j \rightarrow N^r N^s) + \sum_k P(N^j \rightarrow w^k) = 1$

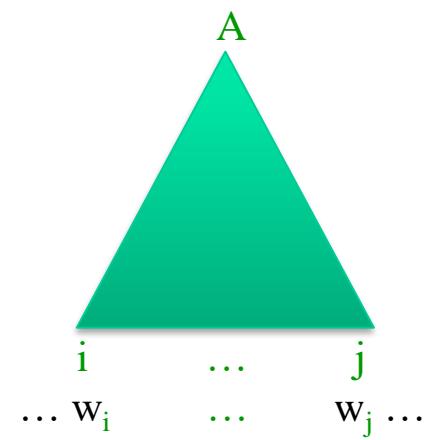
Probabilistic CKY Table

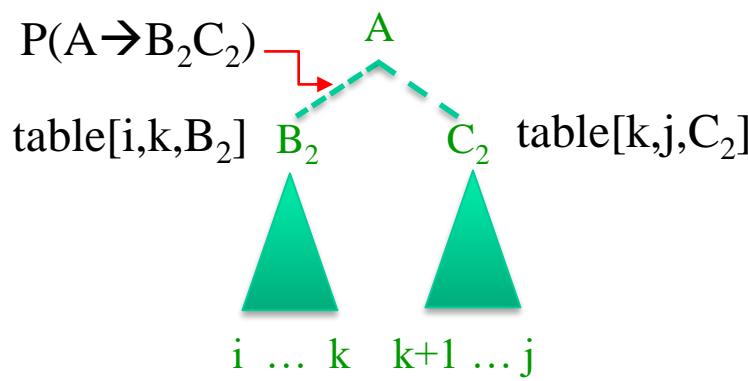
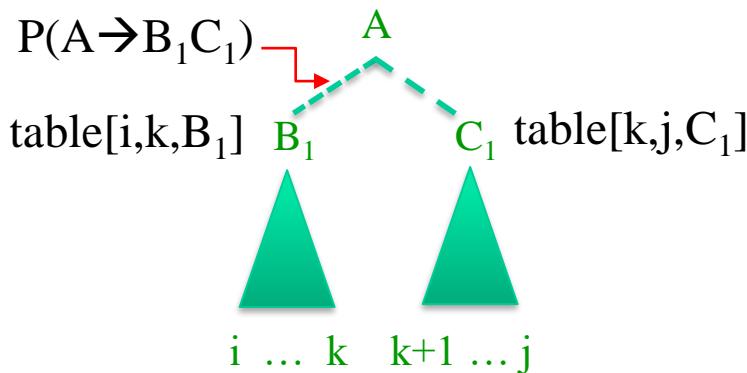
$\text{table}[i,j,A]$: Probability of the subtree with root A spanning from i to j:



$$\text{table}[i,j,A] = \text{table}[i,k,B] \times \text{table}[k,j,C] \times P(A \rightarrow BC)$$

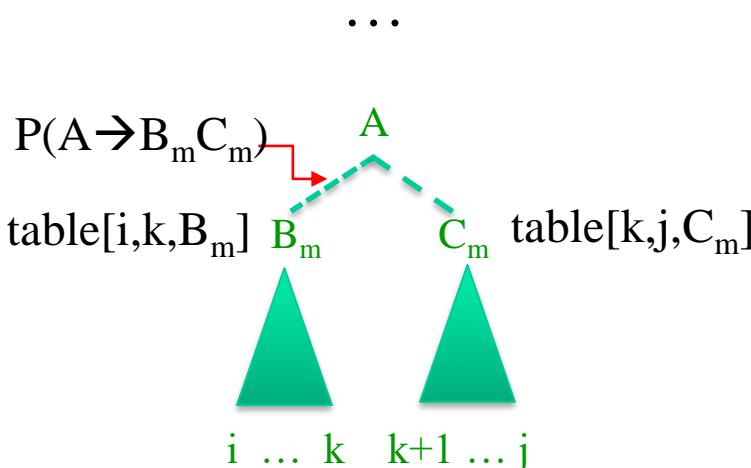
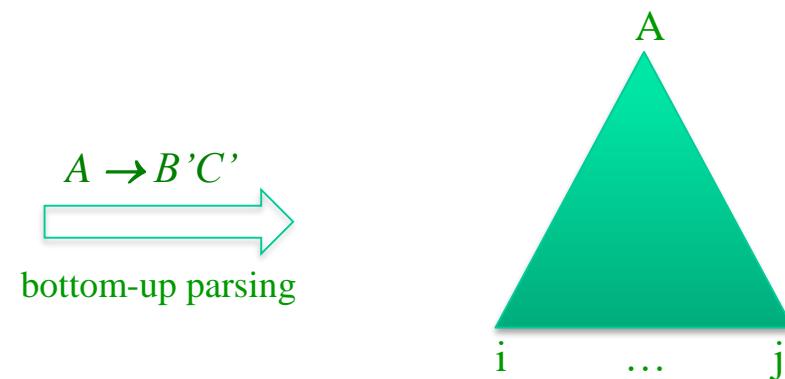
$A \rightarrow BC$
 bottom-up parsing





...

table[i,j,A] =
max(table[i,k,B₁] × table[k,j,C₁] × P(A → B₁C₁),
table[i,k,B₂] × table[k,j,C₂] × P(A → B₂C₂),
...
table[i,k,B_m] × table[k,j,C_m] × P(A → B_mC_m))



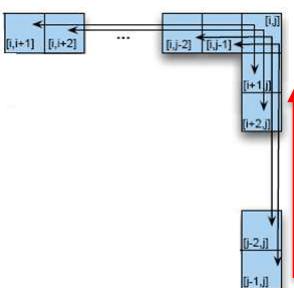
back[i,j,A] = {k,B',C'}

Probabilistic CKY Parsing of PCFGs

```

function PROBABILISTIC-CKY(words,grammar) returns most probable parse
    and its probability
    for  $j \leftarrow$  from 1 to LENGTH(words) do j-th column
        for all {  $A \mid A \rightarrow words[j] \in grammar$  }
             $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$ 
        for  $i \leftarrow$  from  $j-2$  downto 0 do (j-2)-th row to 0-th row
            for  $k \leftarrow i+1$  to  $j-1$  do
                for all {  $A \mid A \rightarrow BC \in grammar,$ 
                    and  $table[i, k, B] > 0$  and  $table[k, j, C] > 0$  }
                    if ( $table[i, j, A] < P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ ) then
                         $table[i, j, A] \leftarrow P(A \rightarrow BC) \times table[i, k, B] \times table[k, j, C]$ 
                         $back[i, j, A] \leftarrow \{k, B, C\}$ 
    return BUILD-TREE( $back[1, \text{LENGTH}(words), S]$ ),  $table[1, \text{LENGTH}(words), S]$ 

```



$S \rightarrow NP VP$.80	$Det \rightarrow the$.40
$NP \rightarrow Det N$.30	$Det \rightarrow a$.40
$VP \rightarrow V NP$.20	$N \rightarrow meal$.01
$V \rightarrow includes$.05	$N \rightarrow flight$.02

<i>The</i>	<i>flight</i>	<i>includes</i>	<i>a</i>	<i>meal</i>
Det: .40 [0,1]	NP: $.30 * .40 * .02 = .0024$ [0,2]	[0,3]	[0,4]	[0,5]
	N: .02 [1,2]	[1,3]	[1,4]	[1,5]
	V: .05 [2,3]	[2,4]	[2,5]	
	Det: .40 [3,4]	[3,5]		N: .01 [4,5]

Rule Probabilities

- What's the probability of a rule?
- Start at the top
 - A tree should have an S at the top. So given that we know we need an S , we can ask about the probability of each particular S rule in the grammar.
 - That is $P(\text{particular rule} \mid S)$
- In general we need

$$P(\alpha \rightarrow \beta \mid \alpha)$$

For each rule in the grammar

Training the Model

- We can get the estimates we need from an annotated database (i.e., a treebank)

$$P(\alpha \rightarrow \beta | \alpha) = \frac{\text{Count}(\alpha \rightarrow \beta)}{\sum_{\gamma} \text{Count}(\alpha \rightarrow \gamma)} = \frac{\text{Count}(\alpha \rightarrow \beta)}{\text{Count}(\alpha)}$$

- For example, to get the probability for a particular *VP* rule, just count all the times the rule is used and divide by the number of *VPs* overall.

Problems with PCFGs

- Poor independence assumptions
- Lack of lexical conditioning
- Probabilistic context-free grammars are incapable of modeling important structural and lexical dependencies

Poor Independence Assumptions

- We compute the probability of a tree by just multiplying the probabilities of each non-terminal expansion
- PCFGs don't allow a rule probability to be conditioned on surrounding context

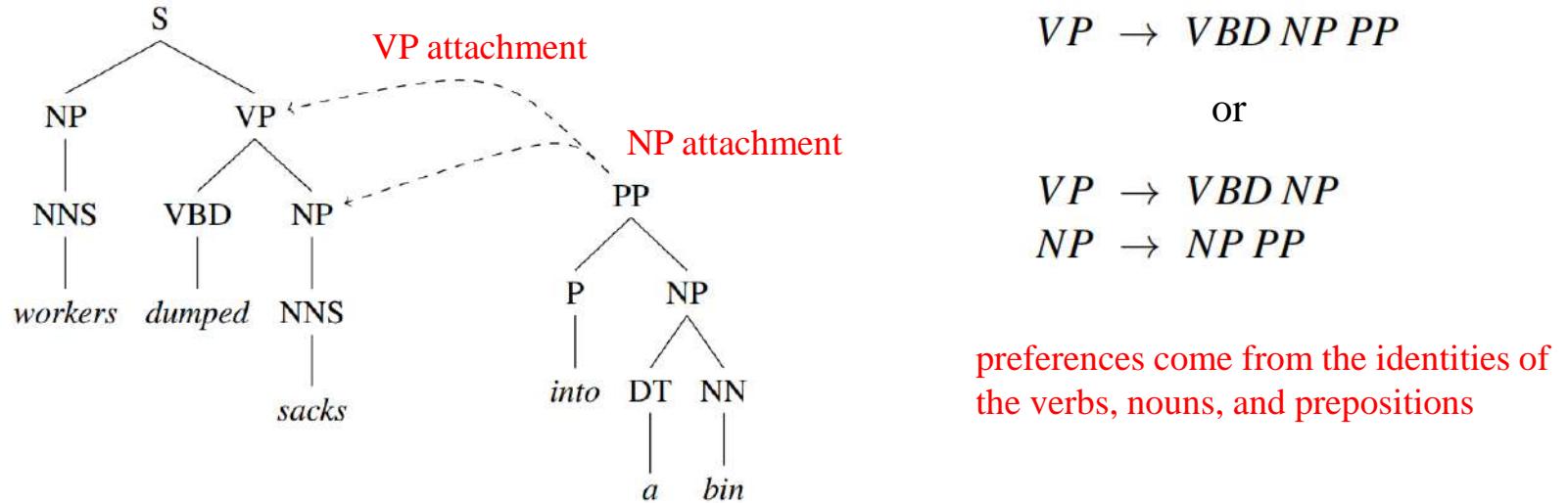
$NP \rightarrow DT\ NN \quad .28$
 $NP \rightarrow PRP \quad .25$

	Pronoun	Non-Pronoun
Subject	91%	9%
Object	34%	66%

- In subject position, the probability for $NP \rightarrow PRP$ should go up to .91, while in object position, the probability for $NP \rightarrow DT\ NN$ should go up to .66.

Lack of Sensitivity to Lexical Dependencies

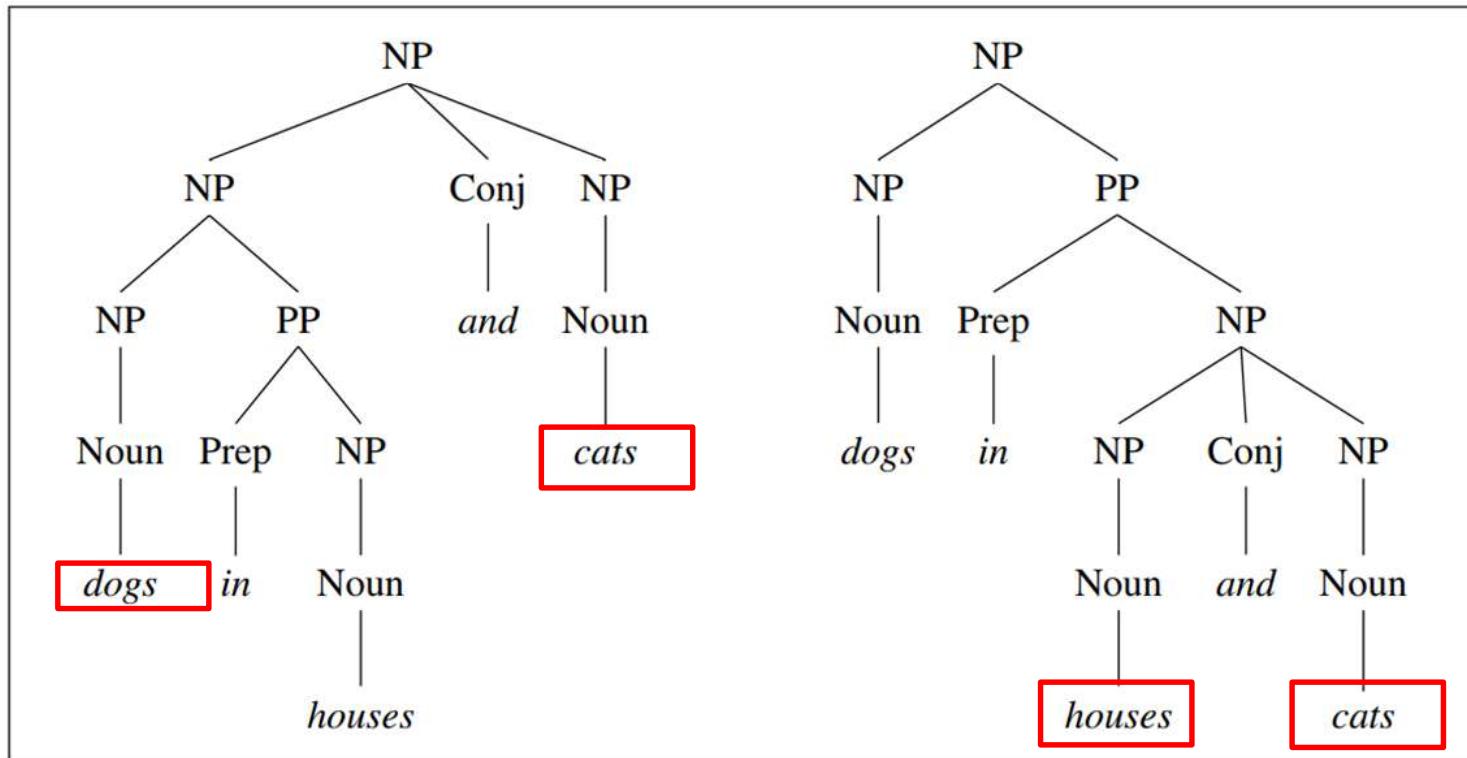
- Prepositional Phrase (PP) Attachment Ambiguities



- Deal with these lexical dependency statistics for different verbs and prepositions
- Lexical dependency statistics for different verbs and prepositions

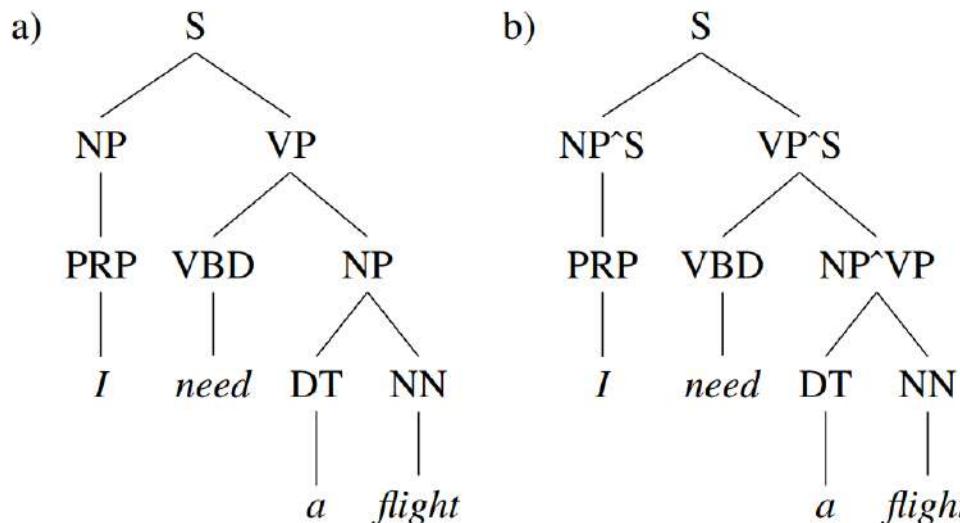
Coordination Ambiguities

- dogs is semantically a better conjunct for cats than houses



Improving PCFGs by Splitting Non-Terminals

- Split the NP non-terminal into two versions: one for subjects, one for objects (e.g., $NP_{subject}$ and NP_{object})
- Parent Annotation
 - NP^S : an NP node that is the subject of the sentence
 - NP^{VP} : a direct object NP whose parent is VP



Sources of This Lecture

- Chapter 12, Foundations of Statistical Natural Language Processing
- Chapter 14, Speech and Language Processing