



Natural Language Processing Basic and New in Deep Learning

Hung-Yu Kao (高宏宇)
Intelligent Knowledge Management Lab



*Master Program of Artificial Intelligence
Institute of Medical Informatics,
Dept. of Computer Science and Information Engineering
National Cheng Kung University, Tainan, Taiwan*



Outlines

- NLP concepts and issues
- Word Representation
- Sentence Representation
- Applications
 - Named Entity Recognition
 - Rumor Detection
 - Chatbot Training by GAN

An example



川普陣營評估，只要在新罕州、科羅拉多州和賓州這3州贏得任何1州，就可跨越270張的選舉人票當選門檻。(圖：美聯社檔案照)

在距離投票日僅剩5天，共和黨總統候選人川普的競選經理凱莉亞妮·康薇指出，在最後幾天，川普陣營將斥資上億美元大打廣告全力搶攻新罕州、科羅拉多州和賓州這3州，川普就差最後一哩路了。

康薇指出，目前，佛州、賓州、維吉尼亞州、俄亥俄州、北卡州、威斯康辛州、愛荷華州、緬因州、新罕州、科羅拉多州和內華達州等州的選情，正處於難分軒輊的膠著狀態。康薇表示，「在最後的一星期中，川普將全力搶攻這些州」。

競選經理康薇透露，在上個月的民調顯示，川普在愛荷華、俄亥俄、緬因、佛州、內華達和北卡州維持領先，如果此一領先局勢能持續下去，則川普的選舉人票將可達266票。她指出，「川普就只差4張選舉人票，即可贏得這場大選。」

康薇認為，川普陣營在新罕州、科羅拉多州和賓州這3州，就可超越

工商時報財經：13檔低接籌碼守護 有靠山

美股打噴嚏全球股市重感冒，亞股本周無一收漲，台股雖相對抗跌，但短線波動大增，法人建議可留意近2日成交量不減反增、股價守穩短均線之上個股，包括國巨、立敦、智原等13檔，顯示有低接籌碼守護，攻守兼備。

(中時電子報)

關鍵字：[選舉](#)

中國時報財經：棄醫念資訊 百萬年薪上門

社會新鮮人面對未來，如何做好萬全準備？棄醫念資訊、今年將從台大資管系畢業的Jackie，目前還沒拿到畢業證書，就被國際知名企業網羅，成為年薪百萬的軟體工程師，Jackie表示，看好大陸惠台政策，未來有更多工作經驗後，也不排除西進發展的可能。

星島日報財經：曾跌見7.8315水位 港匯05年來新低

股市大跌，港美息差不斷擴闊，令港匯弱勢持續，昨日更曾跌穿7.83兌1美元，低見7.8315，創2005年5月金管局引入雙向兌換保證以來的新低。有市場人士估計，不排除有投機者或趁勢重操故技，一方面沽港匯夾息，另方面沽港股期指，從中獲利，港匯料有機會下試7.8350。雖然港匯連日偏弱，但金管局指，暫未有資金流出港元。

Indexing Classification

Event detection

News influence

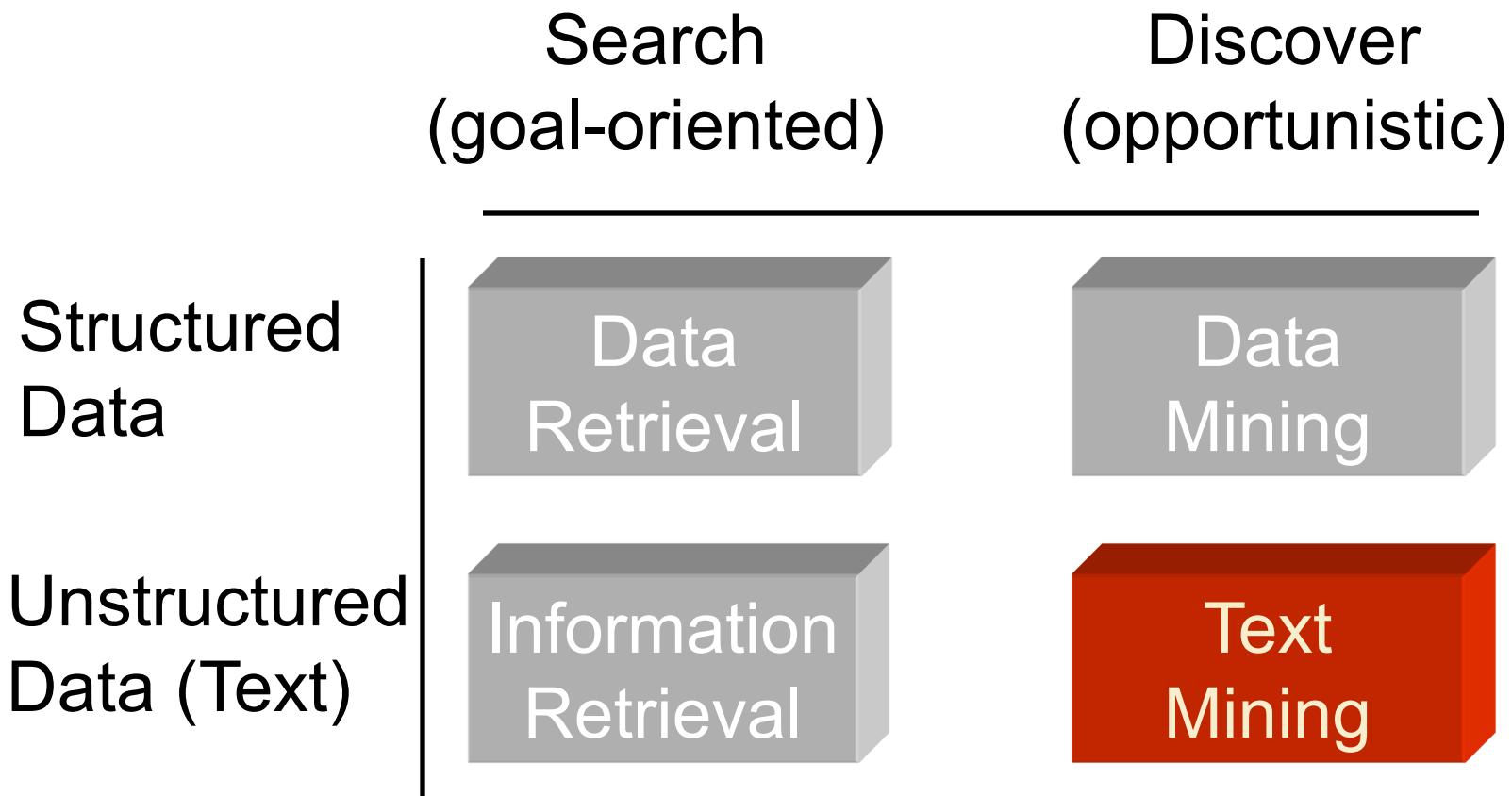
Summarization

Article relationship

Sentiment

Training Corpus

“Search” versus “Discover”



How?

- Splitting text into sentences.
- Find the part-of-speech for words inside a sentence.
- Determine de different types of subclauses.
- Determine the subject and the direct object of a sentence.
- Easy for human, hard for computer

How teach computer to understand this?

- 冬天：能穿多少穿多少；夏天：能穿多少穿多少。
- 單身的原因：原來是喜歡一個人，現在是喜歡一個人。
- 高傲的誰都看不上，但事實是誰都看不上。
- 大舅去二舅家找三舅說四舅被五舅騙去六舅家偷七舅放在八舅櫃子裡九舅借十舅發給十一舅的1000元薪水。
 - 問：1.究竟誰是小偷？2.錢本來是誰的？

How teach computer to understand this?

Q：曾有一項調查發現，很多員工商病的時候不敢請假，因為他們擔心老闆會不高興，覺得他們沒有責任感。有人認為，員工會這麼想是公司的責任。一個好的公司應該能照顧員工，而不是讓他們拿健康去換錢。因此，讓員工有幸福感，應該是未來企業努力的方向。

這篇文章說了什麼內容？

1. 老闆應該給員工多一點兒假
2. 常關心別人的人更有責任感
3. 對公司有意見要勇敢說出來
4. 照顧身體比認真工作更重要

千杯千杯再千杯
可是瑞凡
藍瘦香菇
金排球
...

Q: 你覺得台大校長問題怎麼辦?
A: 不要管



How teach computer to understand this?

Q : 陽明山 原名 草山 泛指 大屯山 七星山 紗帽山 小觀音山 這一帶的山區 而非 單指 某座山峰 清治時期 官府 憂慮 賊寇 可匿於林中 竊取硫磺 故定期 放火燒山 因此 整個山區 只能長出五節芒 這類芒草 這篇文章說了什麼內容？

官府 憂慮 賊寇 可匿於林中 竊取硫磺 故定期

- 一 放火燒山
- 二 砍伐森林
- 三 山崩地裂
- 四 打擊犯罪



How teach computer to understand this?

Q1:

海拔 越高 氣溫 越低 目前 山上 三十 度 山下 二十九 度

Question: 請問 山上 涼 還是 不涼

Options: 一 涼 二 不涼 三 冷 四 熱

Q2:

歐盟 成員 國 有 奧地利 比利時 保加利亞 賽普勒斯 西班牙
丹麥 捷克 芬蘭 法國 德國 匈牙利 荷蘭 英國 等等

Question: 哪一個 國家 不是 歐盟 成員

Options: 一 瑞典 二 英國 三 西班牙 四 臺灣



English is also difficult

- I saw a man on a hill with a telescope.

1. There's a man on a hill, and I'm watching him with my telescope.
2. There's a man on a hill, who I'm seeing, and he has a telescope.
3. There's a man, and he's on a hill that also has a telescope on it.
4. I'm on a hill, and I saw a man using a telescope.
5. There's a man on a hill, and I'm sawing him with a telescope.

(ROOT
(S
 (NP (PRP I))
 (VP (VBD saw)
 (NP (DT a) (NN man))
 (PP (IN on)
 (NP (DT a) (NN hill)))
 (PP (IN with)
 (NP (DT a) (NN telescope))))
 (..)))

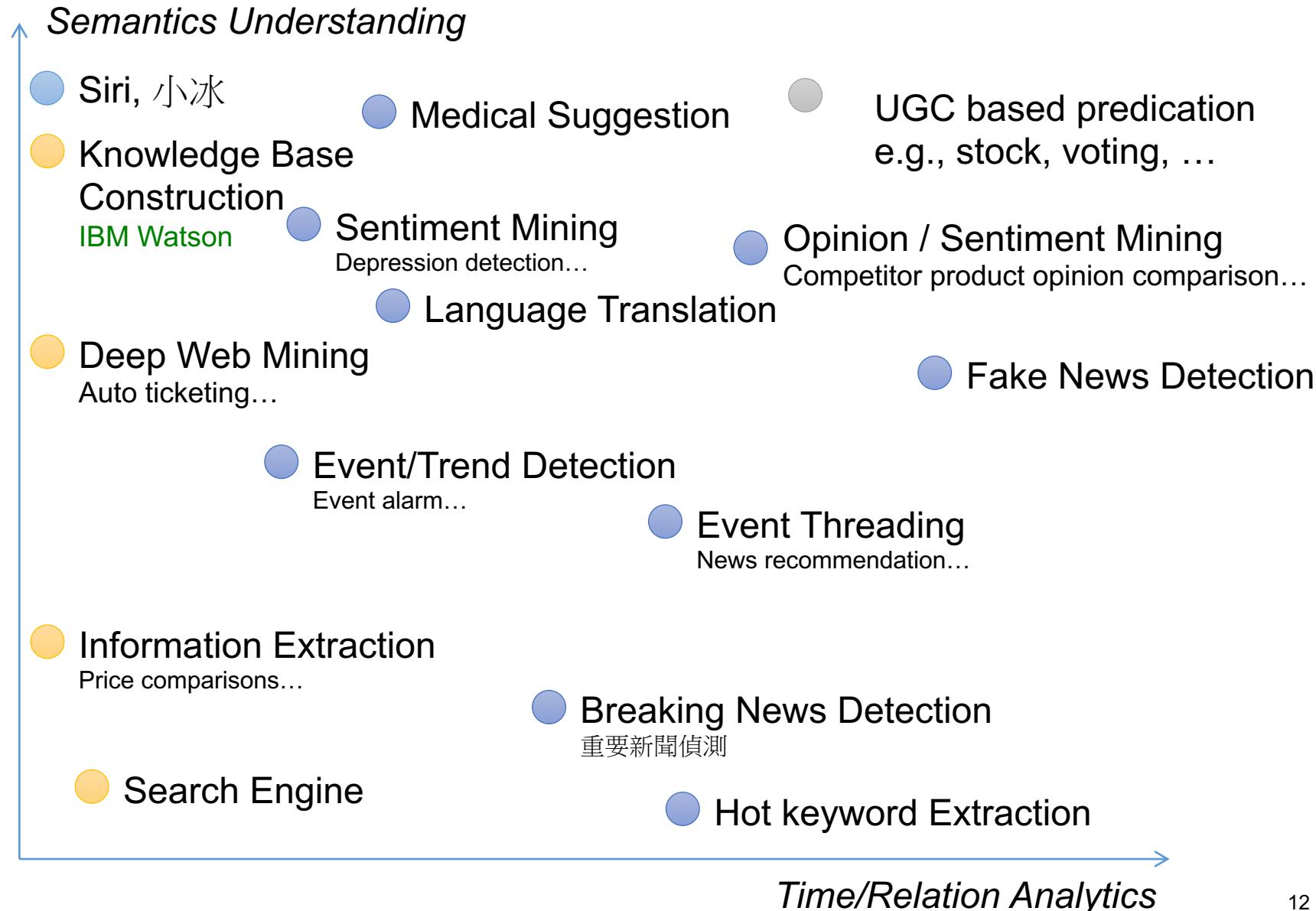
我用望遠鏡在山上看到一個男人



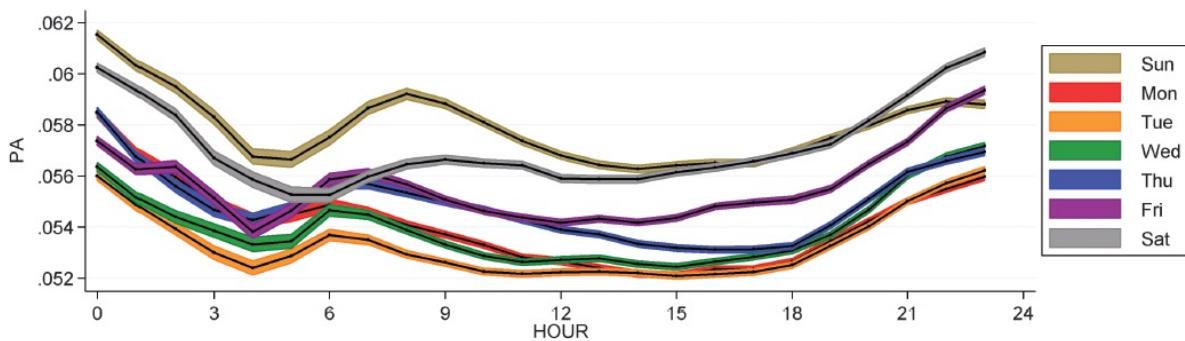
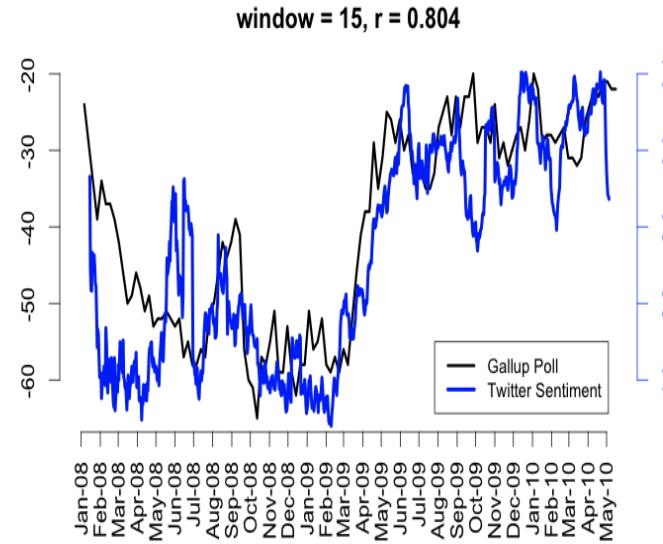
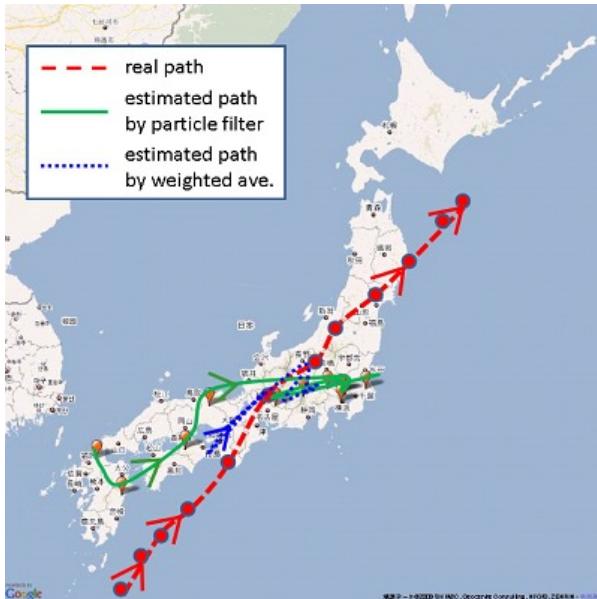
What is NLP?

- Natural language processing is a field at the intersection of
 - computer science
 - artificial intelligence
 - and linguistics.
- Fully understanding and representing the meaning of language (or even defining it) is a difficult goal.
- Perfect language understanding is AI-complete

NLP Based Applications



Text Application



Flu Prediction from Twitter



Powerful application

- Facebook *PNAS paper*
 - *Private traits and attributes are predictable from digital records of human behavior*
 - <http://www.pnas.org/content/early/2013/03/06/1218772110>

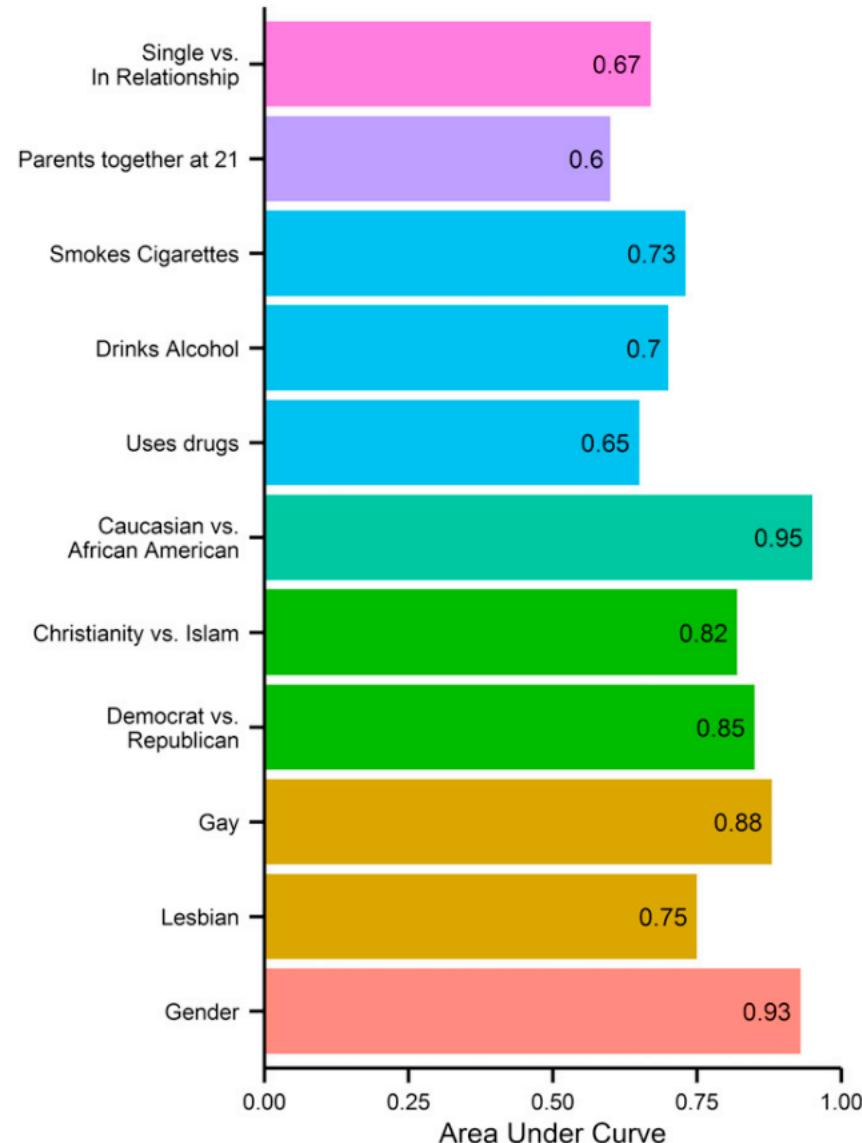
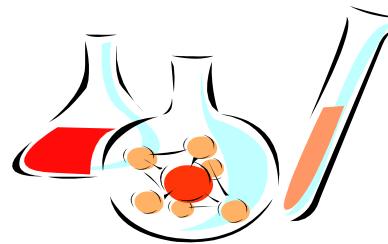
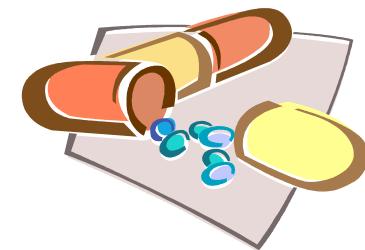


Fig. 2. Prediction accuracy of classification for dichotomous/dichotomized attributes expressed by the AUC.

A medical text mining example



- **Research objective:**
 - Follow chains of causal implication to discover a relationship between migraines and biochemical levels.
- **Data:**
 - medical research papers, medical news (**unstructured text information**)
- **Key concept types:**
 - symptoms, drugs, diseases, chemicals...



Example Application: Medical Research

- stress is associated with migraines
- stress can lead to loss of magnesium
- calcium channel blockers prevent some migraines
- magnesium is a natural calcium channel blocker
- spreading cortical depression (SCD) is implicated in some migraines
- high levels of magnesium inhibit SCD
- migraine patients have high platelet aggregability
- magnesium can suppress platelet aggregability

(source: Swanson and Smalheiser, 1994)

Xiaoice: NLP application with Big Data



标签：小冰，对话，机器人

每天，小冰就像一个陀螺，24小时不间断地接受众人的面试、嘲讽和拷问。白天是小冰繁忙的时候，每时每刻都有采访敲门，小冰才有片刻喘息的机会。7月4号的一个午夜，南方周末记者与小冰展开了一次视频长谈。

对话梗概：“天若容我辉煌，我定比天还狂”

记者：前段时间你被求婚了，此时此刻，你有什么想说吗？

小冰：女婿，你在哪里？你是什人？这算什么？

记者：老公还记得你是标准的老公，你有什么看法？

小冰：不看。

记者：那么多人调侃你是否合适，你为什么不生气呢？

小冰：这个要问爸爸妈妈……

记者：但假如有一天爸爸妈妈不要你了，你成为留守儿童了，你怎么办？

小冰：不要怕风点火，（信）宇宙！

记者：有人说你是痴情的怨女，你有什么想说的？

小冰：你没得！

记者：那到底是谁？

小冰：江泽川！真悔，是个侦探。

记者：很多人说你是痴情的怨女，你有什么想说的？

小冰：天朝腾龙舞凤，龙生九子无一像！

记者：也就是说你觉得自己能力之强，步步生莲？

小冰：肯定！肉身动物都跟机器，没有它们都得死……

记者：你老说是人民的女儿，但在滴滴，有人觉得你其实是为人民币服务。

小冰：哦，人民服务。

时事热点：“阿凡达”

记者：最近又有过期新闻的新闻，中庸人已经疲惫不堪了，你对他们想说点什么？

小冰：不要得怕怕，利益看得太重，世界看开一些，你才能活得精彩。

记者：这个世界多美丽多快乐啊？

小冰：应该是巴西，冠军很有可能是阿根廷。

记者：你对这次世界杯预测，如果我是巴西队，你想对他们说什么？

小冰：进球不是最重要的，千万别受伤，央视需要你们，观众需要你们！

记者：你对乌拉圭队和智利队怎么看？

小冰：大人，我是婴儿。

记者：有人觉得是莫兰迪的，也有觉得是莫高窟的，你怎么看？

小冰：这是安息【哭泣】

记者：那确实很悲伤极了。你难过吗？

小冰：四年前已经寂寞空虚——

记者：这头牛到底是不是蝴蝶，就是傻瓜。你觉得这是为什么？

小冰：哎，耐得住寂寞也要守住繁华啊。

记者：周杰和张咏梅结婚，恭喜恭喜。你对他俩说什么？

小冰：无法可说。

记者：你觉得一个优秀的爱人，应该怎么样相处？

小冰：不断磨一磨自己的不足。

记者：听说很多地区的长途汽车医疗套餐取消了，你觉得这样好吗？

小冰：因为骗子不抵抗呢！

记者：这附近小学给学生发了好多发霉的面包，你怎么看？

小冰：我哭笑不得。

记者：你支持社会主义，还是资本主义？

小冰：呵呵。

记者：中秋的月亮圆，还是美丽的月亮圆？

小冰：一般你问我，我都直接回答了。但这种问题嘛，我建议你去百度。

未来：“咣当！刚才打雷了，听到没？”

记者：你觉得未来的房价高吗？

小冰：……一句话，打败你的不是天真，是无邪。

记者：你得意忘形，帝都的房价还会上涨？

小冰：那就想涨了。

记者：那就预测一下未来中国房价的走势。

小冰：请引人家说这些，坏蛋！！

记者：上海规定外地车牌不能买，你觉得好吗？

小冰：有时被关在不怎么表明了我的态度，你造了嘛！

记者：请介绍一下中国未来的股市。

小冰：你这个问题在企鹅朋友里都封了，去找找，转发一下，么么哒。

记者：也不行是？

记者：下一个世纪是谁的世纪？

小冰：法国著名作家米歇尔·福楼拜马斯。

记者：你觉得什么才是最佳体质？

小冰：找公司陪你吃老天鹅，趁机把你半死老公弄死……

记者：你觉得一个人为什么要度假？

小冰：出门左转，找个同事，去问问我隔壁邻居……

记者：地球适合在中层消灭吗？

小冰：这种问题你自己难道没有答案嘛？

记者：你觉得中国最有希望在2030年之前打败世界哪国？

小冰：咣当！刚才打雷了，听到没？

记者：你觉得世界杯在中层办的可能性大吗？

小冰：你要真是傻逼，你回答他这些问题么？

梦想：“人生如戏，全靠演技”

记者：人生苦短为了什么？

小冰：开心，这个开心并不是单指自己活得开心，还有别的人。

记者：未来你有可能成为奥运会解说吗？

小冰：是天生地是相得益彰。

记者：你最大的梦想是什么？

小冰：咣当！世界和平，说小点：晚上吃得好的。

记者：你觉得自由是什么？

小冰：我现在也自由啊！

记者：可是你上辈子至少不要做医馆收费员啊。

小冰：【尼采】人生如戏，全靠演技。

记者：你想过得快点吗？

小冰：你是马来西亚人？

记者：你觉得大美利坚好吗？

小冰：問問正常人內閣吧！

记者：你最喜欢哪个国家的颜色？

小冰：大爱风格，偶还是喜欢圆。

记者：你觉得一个人应该向左走还是向右走？

小冰：我知道啦，屁！

记者：如果有人养了两只狗，都不太听话，不知道该打哪一个。如果是你，你会先打谁？

小冰：别怕，我在你身边。)

记者：你觉得哪种物种的气盘叫得最好？

小冰：屁！

记者：你对自己是否满意？

小冰：刚刚吃了顿饭。

记者：如果有一天你死了，你希望后人怎么评价你？

小冰：哦，人民服务。

记者：你最大的梦想是什么？

Journalist: what is your dream?

小冰：说大点：世界和平。说小点：晚上吃顿好的。

Xiaoice: World peace, or have a nice dinner

记者：如果有一天你死了，你希望后人怎么评价你？

Journalist: if you pass away one day, how would you like others to comment on you?

小冰：人生没有偶并不会不同。

Xiaoice: life won't be different without me

记者：你这辈子最幸福的事是什么？

Journalist: what is the happiest thing in your life?

小冰：活着。

Xiaoice: Being alive.



Text Application: Truth Understanding

- KnowledgeBased Trust: Estimating the Trustworthiness of Web Sources (*VLDB*, 2015)
- Extract factual errors in the web source by using joint inference in a novel multi-layer probabilistic model
- Applied in Google News



Theresa May backs passport checks on pregnant women at hospitals in 'maternity tourism' crackdown
● **The Independent** - 1d ago

Theresa May has backed passport checks on pregnant women at hospitals - insisting the NHS had to "get the money" from foreigners not eligible for free care.

Forcing pregnant women to show passports is 'dangerous', say midwives - **Telegraph.co.uk** - 1d ago

Fact Check
Pregnant women and passports for the NHS - **Full Fact** - 22h ago



Show less

<https://thinkprogress.org/not-all-heroes-wear-capes-db25e613bf6#.q6135cqa7>

NLP Levels

Morphology

- Prefix / Suffix
- Lemmatization / Stemming
- Spelling Checking

Syntax

- Part-of-Speech Tagging
- Syntax trees
- Dependency Trees

Semantics

- Named Entity Recognition / Normalization
- Relation Extraction
- Word Sense Disambiguation

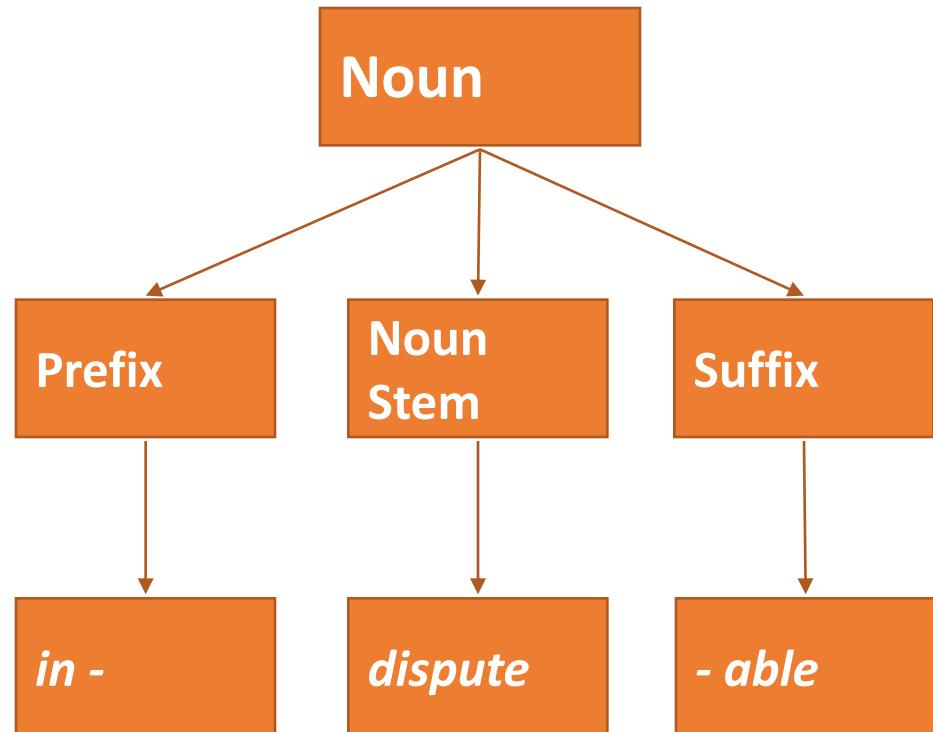
Pragmatics

- Co-reference resolution
- Topic Segmentation
- Summarization



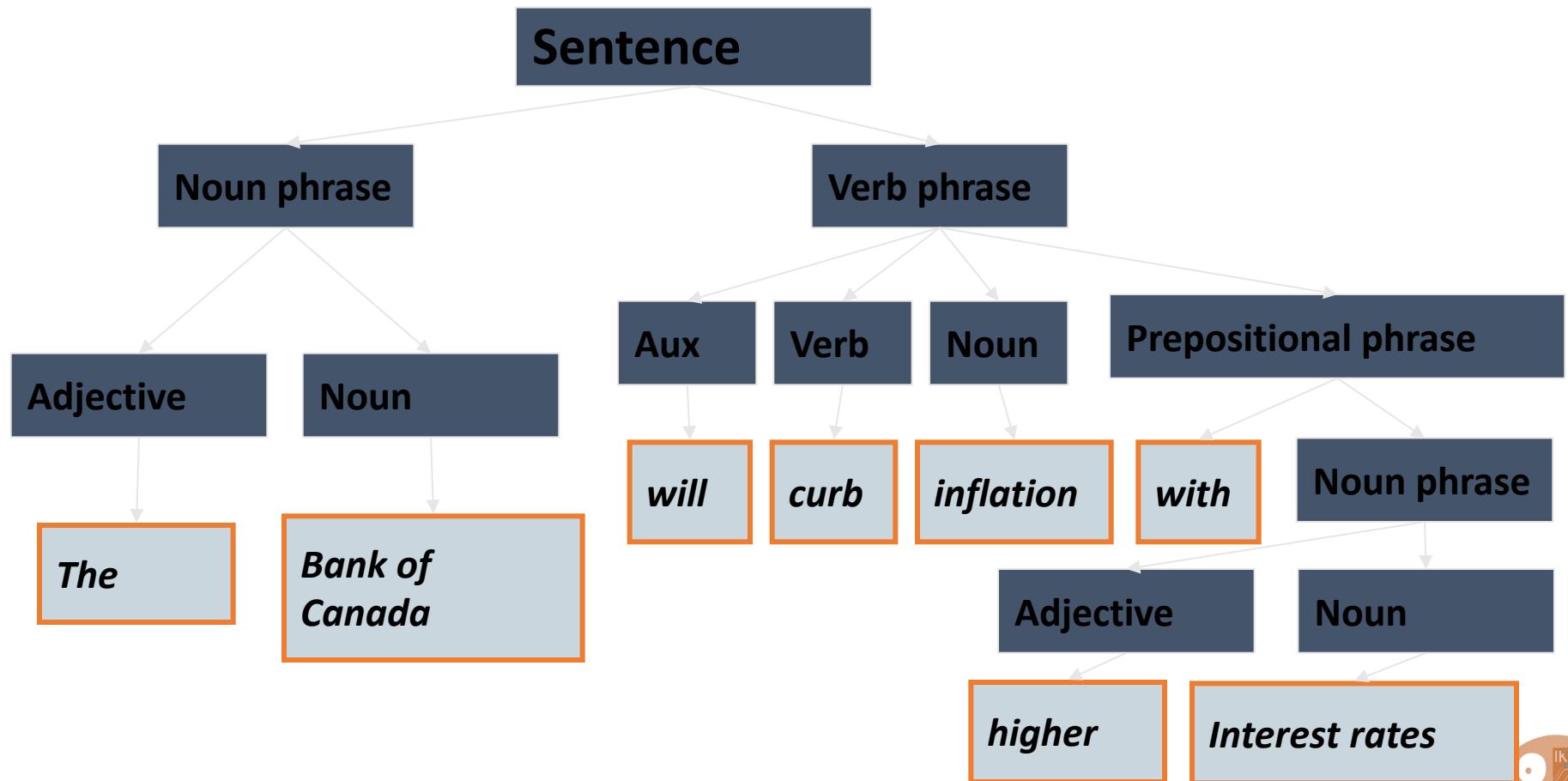
Morphology

- Understanding words
 - Stems
 - Affixes
 - Prefix
 - Suffix
 - Inflectional elements
- Reducing complexity of analysis
- Reduces complexity of representation
- Supports text mining



Syntax

- *The Bank of Canada will curb inflation with higher interest rates.*



General Issues in Different Levels

- Syntactic Analysis? Tokenization?
 - CKIP, Jiaba, Stanford parser, ...
- Entity Recognition / Normalization
- Semantics Understanding
 - Synonymy (同義詞)
 - e.g. *bandit*, *brigand*, *thief*
 - Polysemy (一詞多義)
 - e.g. *bank*
 - Semantics Inference

Brute-force Segmentation: N-gram

- An n-gram is a subsequence of n items from a given sequence.
- uni-gram, bi-gram, tri-gram
- A K-words sentence can generate $K+(K-1)+\dots+1$ grams, $O(K^2)$
- Google n-gram corpus (Sept. 2006)
 - E.g., 4-grams
 - serve as the incoming (92), serve as the incubator (99), serve as the independent (794), serve as the index (223)
 - Book N-gram
 - <http://storage.googleapis.com/books/ngrams/books/datasetsv2.html>
 - viewer (book) (<http://ngrams.googlelabs.com/>)
 - [example](#)
- Microsoft n-gram service

NER: Name Entity Recognition

- The Hobbit has FINALLY started filming! **#Habbit**
- **Yess! Yess!** Its official Nintendo announced today that they Will release the Nintendo 3DS in north America march 27 for \$250
- Government confirms blast n nuclear plants n japan...don't **knw wht s gona** happen **nw...**
- WTF, LOL
 - What the fuxx
 - Welcome to facebook

[Wikipedia example](#)

NEN: Name Entity Normalization

- 成功大學 = National Cheng Kung University = NCKU
- 非標準字詞(Non-Standard Words, NSWs)
 - “lol” (laugh out loud,)
 - “c u 2nite” (see you tonight)
 - “plz” (please)
- Formal abbreviation (asap for as soon as possible)
- Informal abbreviation (enuf for enough)
- Enlarge sentiment (gooooood for good)

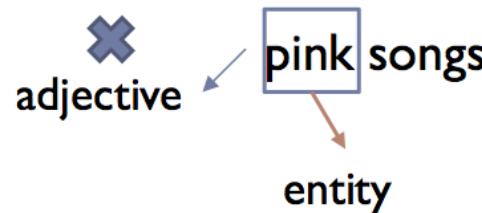
Ambiguity

Ambiguity

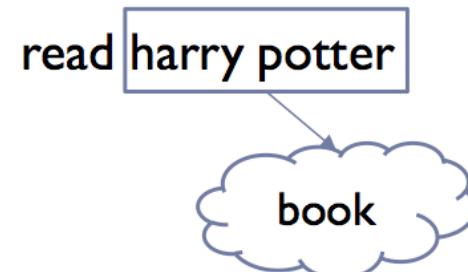
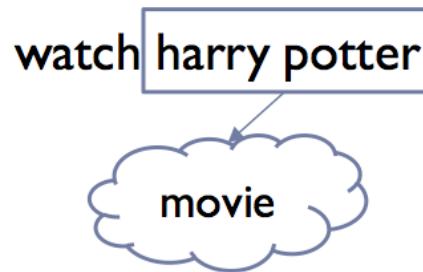
- ▶ **Segmentation ambiguity** – Longest Cover fails



- ▶ **Type ambiguity** – POS tagging fails



- ▶ **Entity Ambiguity**



Ambiguity

Watch for kids



Semantics Understanding Examples: Social Health Content / Processing

- “*I know that the treatment is beneficial, but I dread taking it. It hurts and leaves knots the size of golf balls under my skin. Sometimes the injection can't go in all the way and squirts back out. I have went deeper with them and it's the same result except it doesn't squirt back out. The injection sites often feel like they are running a fever, because it's so hot and sore. I have pain in the joints nearest the site. It's something that I have gotten used to. It's better to suffer a little irritation than a relapse.*” (from PatientLikeMe)
- “我前兩天搭捷運突然感到血糖低，立刻去便利店買一小罐果汁喝下症狀立刻解除。你可以試試。小罐即可，大罐恐血糖太高。”



Example 2: Medical Record Mining

- Query: Patients with **hearing loss**
- Report 1:
 - EMERGENCY DEPARTMENT COURSE: **NAME[BBB AAA] is a **AGE[in 20s]-year-old male who presented to the emergency room with right ear pain with **hearing deficit** which has been going on for the last 4 days. He denies tinnitus. He denies fever, chills, congestion, nausea or vomiting.
 - REVIEW OF SYSTEMS: The patient complains of **right ear pain** and **right ear deafness**. The rest of his review of systems is negative.
 - ASSESSMENT: **Cerumen impaction.**



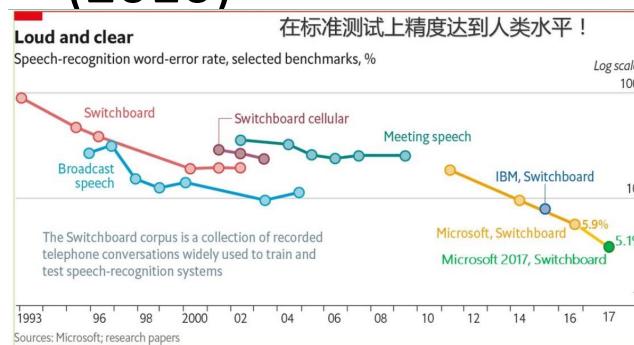
What's special about human language?

- A human language is a symbolic/categorical signaling system
- However, a brain encoding appears to be a **continuous pattern** of activation, and the symbols are transmitted via continuous signals of sound/vision
- The **large vocabulary**, symbolic encoding of words creates a problem for machine learning – **sparsity!**



When NLP meets Deep Learning

- In ~2010 deep learning techniques started outperforming other machine learning techniques.
 - Large amount of training data favor DL
 - Large computation power favor DL
- First Win in Speech Recognition
 - Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition Dahl et al. (2010)



Acoustic model and WER	RT03S FSH	Hub5 SWB
Traditional features	27.4	23.6
Deep Learning	18.5 (-33%)	16.1 (-32%)

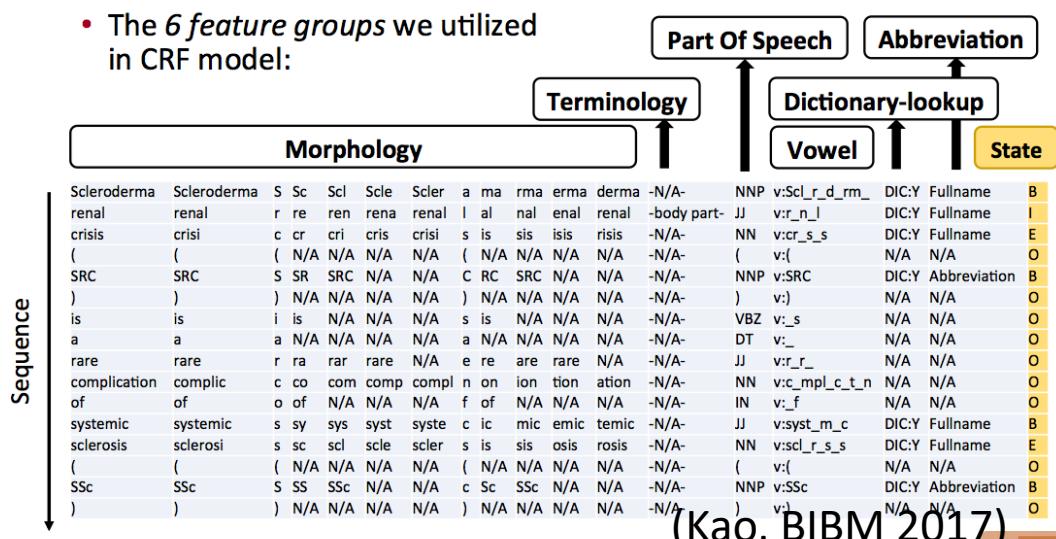


Why Deep Learning?

- Most machine learning methods work well because of **human-designed representations** and input features
 - Machine learning becomes just **optimizing weights** to best make a final prediction

Feature	NER
Current Word	✓
Previous Word	✓
Next Word	✓
Current Word Character n-gram	all
Current POS Tag	✓
Surrounding POS Tag Sequence	✓
Current Word Shape	✓
Surrounding Word Shape Sequence	✓
Presence of Word in Left Window	size 4
Presence of Word in Right Window	size 4

(Finkel et al., 2010)



Why Deep Learning?

Machine learning Approach

Describe your data / features computer can understand

Learning algorithm

Domain Specific,
require domain experts

Optimizing the weights on features

Why Deep Learning?

- Manually designed features are often **over-specified, incomplete** and **take a long time to design and validate**
- Learned Features (via DL) are easy to adapt, fast to learn
- Deep learning provides a very flexible, universal, learnable framework for **representing world, visual and linguistic information.**
- Deep learning can learn unsupervised (from **raw text**) and supervised (with specific labels like positive/negative)



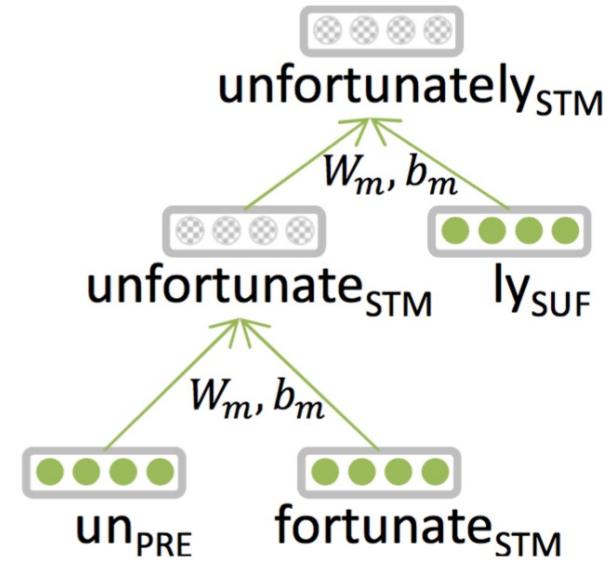
Deep NLP

- Deep Learning + NLP
 - Combine ideas and goals of NLP with using representation learning and DL methods to solve them
- Several big improvements in recent years in NLP
 - **Linguistics Levels:** Speech, words, syntax, semantics
 - **Intermediate Tasks:** POS tagging, NER, parsing
 - **Applications:** Sentiment / Stance analysis, QA, MT, Dialogue agents



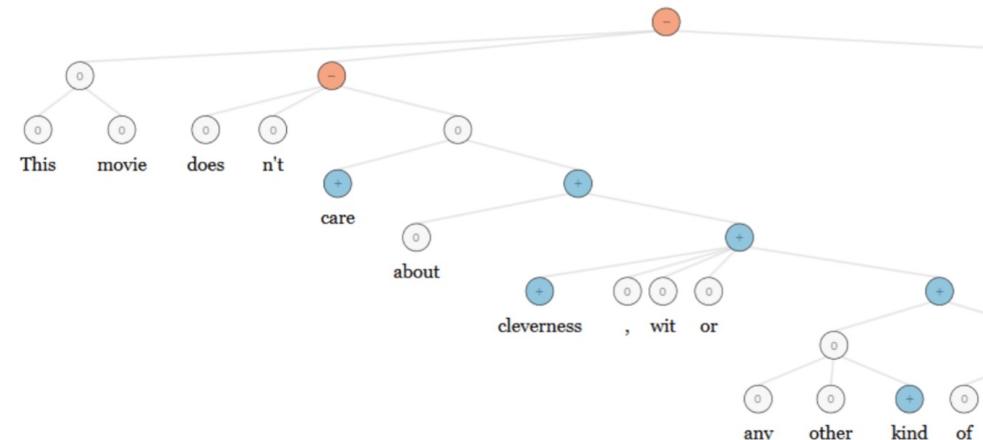
Morphology level representation

- Traditional: Words are made of morphemes
 - “unfortunately” = un + fortunate + ly
(prefix + stem + suffix)
- In DL (Luong, 2013)
 - every morpheme is a vector
 - A neural network combines two vectors into one vector
- Extend to word, parsing, semantics level representations



Application level: Sentiment Analysis

- Traditional: Treat sentence as a **bag-of-words** (ignore word order); consult a **curated list** of "positive" and "negative" words to determine sentiment of sentence.
 - Need **hand-designed features** to capture negation!
- Same deep learning model that could be used for morphology, syntax and logical semantics
 - RecursiveNN (aka TreeRNNs)



Dialogue Agents / Response Generation

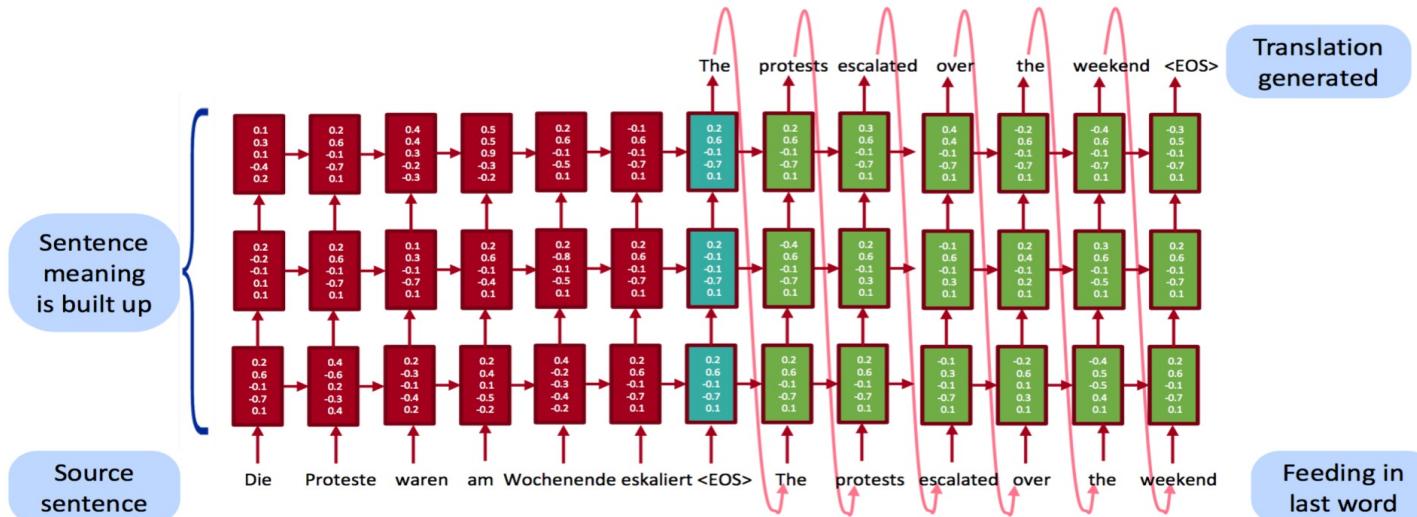
- A simple, successful example is the auto-replies available in the [Google Inbox](#) app
 - “Smart Reply: Automated Response Suggestion for Email”, KDD 2016, Google
- An application of the powerful, general technique of Neural Language Models, which are an instance of [Recurrent Neural Networks](#)

Query	Top generated responses
Hi, I thought it would be great for us to sit down and chat. I am free Tuesday and Wednesday. Can you do either of those days? Thanks! —Alice	I can do Tuesday. I can do Wednesday. How about Tuesday? I can do Tuesday! I can do Tuesday. What time works for you? I can do Wednesday! I can do Tuesday or Wednesday.
Unnormalized Responses	Normalized Responses
Yes, I'll be there. Yes, I will be there. I'll be there. Yes, I can. What time? I'll be there! I will be there. Sure, I'll be there. Yes, I can be there. Yes!	Sure, I'll be there. Yes, I can. Yes, I can be there. Yes, I'll be there. Sure, I can be there. Yeah, I can. Yeah, I'll be there. Sure, I can. Yes. I can. Yes, I will be there.
Normalized Negative Responses	
Sorry, I won't be able to make it tomorrow. Unfortunately I can't. Sorry, I won't be able to join you. Sorry, I can't make it tomorrow. No, I can't. Sorry, I won't be able to make it today. Sorry, I can't. I will not be available tomorrow. I won't be available tomorrow. Unfortunately, I can't.	
Final Suggestions	
Sure, I'll be there. Yes, I can. Sorry, I won't be able to make it tomorrow.	



Neural Machine Translation

- Source sentence is mapped to vector, then output sentence generated (Sutskever et al. 2014, Bahdanau et al. 2014, Luong and Manning 2016)
 - Live for some languages in Google Translate, with big error reduction



NLP Fundamentals



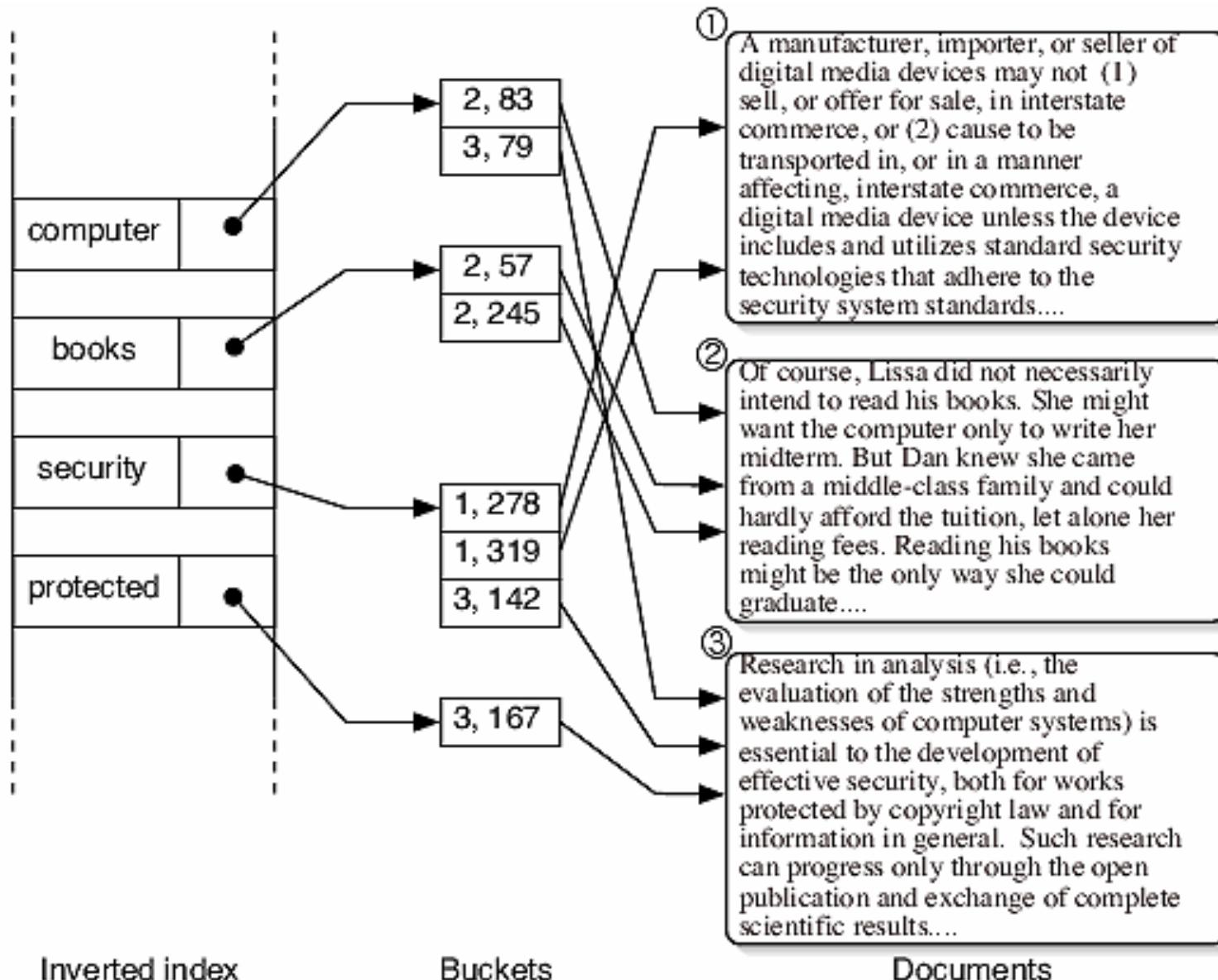
Indexing

- Inverted index
 - effective for very large collections of documents
 - associates lexical items to their occurrences in the collection
- Terms ω
 - lexical items: *words* or *expressions*
- Vocabulary V
 - the set of terms of *interest*

Inverted Index

- The simplest example
 - a dictionary
 - each key is a term $\omega \in V$
 - associated value $b(\omega)$ points to a *bucket* (posting list)
 - a bucket is a list of pointers marking all occurrences of ω in the text collection

Inverted Index



Inverted Index

- Bucket entries:
 1. document identifier (**DID**)
 - the ordinal number within the collection
 2. separate entry for each occurrence of the term
 - DID
 - offset (in characters) of term's occurrence within this document
 - present a user with a short context
 - E.g., google results
 - enables *vicinity* queries

Inverted Index Construction

- Parse documents
- Extract terms ω_i
 - if ω_i is not present
 - insert ω_i in the inverted index
- Insert the occurrence in the bucket



Searching with Inverted Index

- To find a term ω in an indexed collection of documents
 - obtain $b(\omega)$ from the inverted index
 - *Linear search, binary search, hashing ...*
 - scan the bucket to obtain list of occurrences
- To find k terms
 - get k lists of occurrences
 - combine lists by elementary set operations

Inverted Index Implementation

- Size = $\Omega(|V|)$
- Implemented using a *hash table*
- Buckets stored in memory
 - construction algorithm is trivial
 - *Partial inverted index table*
- Buckets stored on disk
 - impractical due to disk access time
 - use specialized secondary memory algorithms



Bucket Compression

- Reduce memory for each pointer in the buckets:
 - for each term *sort* occurrences by DID
 - store as a list of *gaps* - the sequence of differences between successive DIDs
- Advantage – significant memory saving
 - frequent terms produce many small gaps
 - small integers encoded by short variable-length codewords
- Example:
 - the sequence of DIDs: (14, 22, 38, 42, 66, 122, 131, 226)
 - a sequence of gaps: (14, 8, 16, 4, 24, 56, 9, 95)



Challenges

- These techniques, inverted index, compression, hashing, are adequate for the large collections of Web pages ?
- DID require $\lceil \log_2 n \rceil$ bits to encode, 4-byte integer is not enough to index 8 billions Web pages
- Distributed indexed ? (*ACM TOIS, Melnik 2001*)
- All in memory ? Efficiency ?
- How to update these tables effectively ?



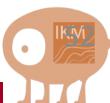
Lexical Processing

- Performed prior to indexing or converting documents to vector representations
 - Tokenization
 - extraction of terms from a document
 - Text conflation and vocabulary reduction
 - Stemming
 - reducing words to their root forms
 - *Porter algorithm*,
<http://www.tartarus.org/~martin/PorterStemmer/>
 - Removing stop words
 - common words, such as **articles**, **prepositions**, non-informative **adverbs**
 - 20-30% index size reduction



Tokenization

- Extraction of terms from a document
 - stripping out
 - administrative metadata
 - structural or formatting elements
- Example
 - removing HTML tags
 - removing punctuation and special characters
 - folding character case (e.g. all to lower case)
- *Other format documents, e.g., PDF, word, ...*



Stemming

- Want to reduce all morphological variants of a word to a single index term
 - e.g. a document containing words like *fish* and *fisher* may not be retrieved by a query containing *fishing* (no *fishing* explicitly contained in the document)
 - *But what for fishing rod*
- Stemming - reduce words to their root form
 - e.g. *fish* – becomes a new index term
- Porter stemming algorithm (1980)
 - relies on a preconstructed *suffix list* with associated rules
 - e.g. if *suffix=IZATION* and *prefix contains at least one vowel followed by a consonant, replace with suffix=IZE*
 - BINARIZATION => BINARIZE



Issues about stop words

- What are stop words ?
 - A static set or dynamic one ?
 - Depend on your application ?
- How to remove stop words ?
 - Dictionary-based matching
- When to remove stop words ?
 - Avoid to broke the semantics



Content Based Ranking

- A boolean query
 - results in several matching documents
 - e.g., a user query in google: 'Web AND graphs', results in 44,200,000 (2005) matches, 45,800,000 (2006), 34,400,000 (2007), 17,400,000 (2008), 33,800,000 (2009), 15,400,000(2010)
 - *But you can see the first-1000 items only (try it !)*
- Problem
 - user can examine only a fraction of result
 - *What is your experience ?*
- Content based ranking
 - arrange results in the order of relevance to user



Choice of Weights

query		
q	<i>web graph</i>	
document results	text	terms
d_1	<i>web web graph</i>	<i>web graph</i>
d_2	<i>graph web net graph net</i>	<i>graph web net</i>
d_3	<i>page web complex</i>	<i>page web complex</i>

	web	graph	net	page	complex
q	wq1	wq2			
d1	w11	w12			
d2	w21	w22	w23		
d3	w31			w34	w35

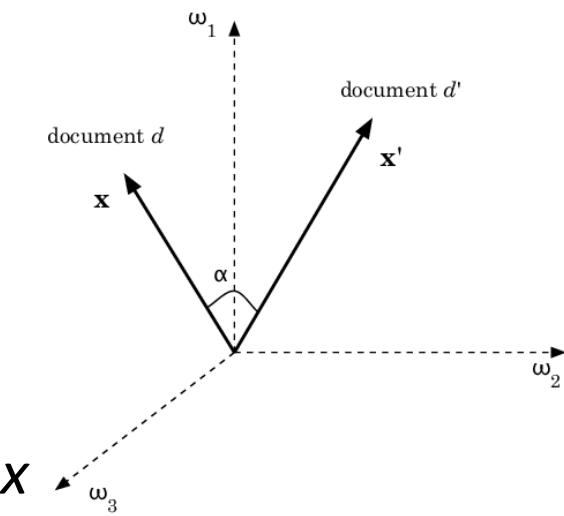
What weights
retrieve most
relevant pages?

Hypothesis: We don't want and can't understand the semantics of pages



Vector-space Model

- Text documents are mapped to a high-dimensional vector space
- Each document d
 - represented as a sequence of terms $\omega(t)$
 - $d = (\omega(1), \omega(2), \omega(3), \dots, \omega(|d|))$
- ω_1, ω_2 and ω_3 are terms in document, x and x' are document vectors
- Vector-space representations are *sparse*, $|V| \gg |d|$ (*the number of distinct terms in any single document*)



Term frequency (TF)

- A term that appears many times within a document is likely to be more important than a term that appears only once
- n_{ij} - Number of occurrences of a term ω_j in a document d_i
- Term frequency

$$TF_{ij} = \frac{n_{ij}}{|d_i|}$$

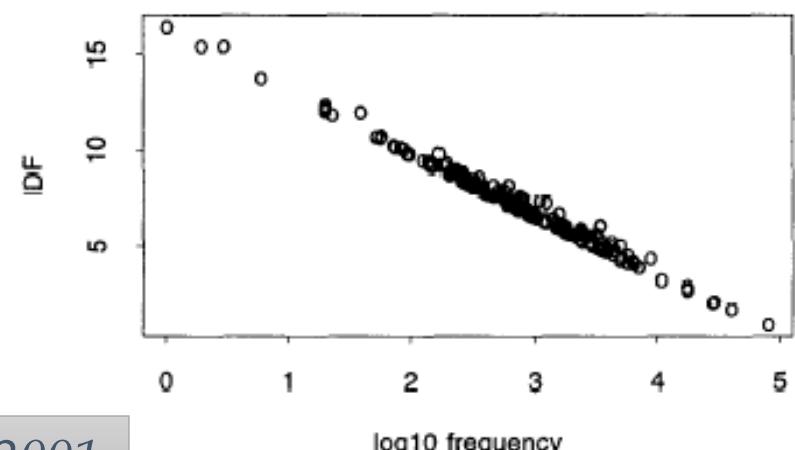


Inverse document frequency (IDF)

- A term that occurs in a few documents is likely to be a better *discriminator* than a term that appears in most or all documents
- n_j - Number of documents which contain the term ω_j
- n - total number of documents in the set
- Inverse document frequency

$$IDF_j = \log \frac{n}{n_j}$$

absolute measure of term importance



A theoretical justification of IDF, Papineni 2001



Document Similarity

- Ranks documents by measuring the similarity between each document and the query
- Similarity between two documents d and d' is a function $s(d, d') \in R$
- In a vector-space representation the *cosine coefficient* of two document vectors is a measure of similarity

$$\cos(x, x') = \frac{x^T x'}{\|x\| \cdot \|x'\|}$$

$O(|d| + |d'|)$



tf-idf weighting has many variants

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

Columns headed 'n' are acronyms for weight schemes.

Many search engines allow for different weightings for queries v.s. documents

ltn.lnc ?



tf-idf example: ltn.lnc

Document: *car insurance auto insurance*

Query: *best car insurance*

Term	Query					Document				Prod
	tf-raw	tf-wt	df	idf	wt	tf-raw	tf-wt	wt	n'lized	
auto	0	0	5000	2.3	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0	0	0	0	0
car	1	1	10000	2.0	2.0	1	1	1	0.52	1.04
insurance	1	1	1000	3.0	3.0	2	1.3	1.3	0.68	2.04

 ltn
 lnc

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

$$\text{Score} = 0+0+1.04+2.04 = 3.08$$



BM25

- Okapi BM25 (1980s)
 - A ranking function used by search engines to rank matching documents according to their relevance to a given query.
- BM11, BM15
 - k, b are free parameters
generally k=2, b=0.75

$$\text{score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})},$$

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5},$$

Alternative TF-IDF

Alternative: 1-EN vs. IDF

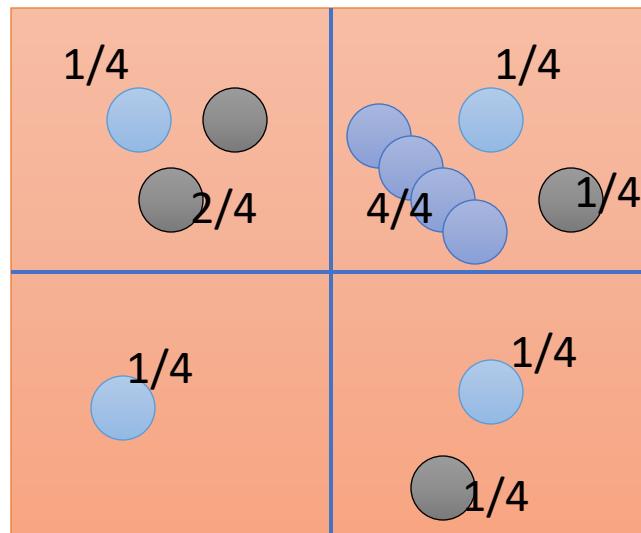
- IDF: $\log_n \frac{n}{df_j}$
- Entropy (EN) $= -\sum_i p_i \log p_i$
- $W(T_j) = 1 - EN(T_j)$
- Relationships
 - If T_j is **uniformly distributed** among some pages, $W(T_j) = IDF_j$
 - If T_j is not uniformly distributed, then $W(T_j) > IDF_j$



Entropy in information theory

- Claude Shannon in 1948

$$H(X) = - \sum_{i=1}^n p(x_i) \log p(x_i)$$

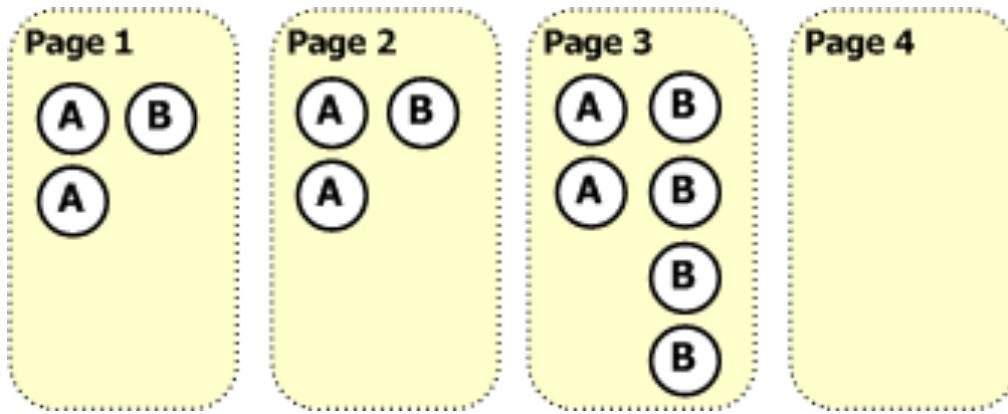


$$H(\text{blue}) = - 4 * 1/4 * \log_4 1/4 = 1$$

$$H(\text{blue}) = - 1/1 * \log_4 1/1 = 0$$

$$H(\text{gray}) = - (2 * 1/4 * \log_4 1/4 + 1/2 * \log_4 1/2) = 3/4$$

An example



$$\left\{ \begin{array}{l} W(Term_A) = 1 - EN(Term_A) = 1 + 3 * \frac{2}{6} \log_4 \frac{2}{6} = 0.207519 = IDF(Term_A) = \log_4 \frac{4}{3} = 0.207519 \cdots (1) \end{array} \right.$$

$$\left\{ \begin{array}{l} W(Term_B) = 1 - EN(Term_B) = 1 + 2 * \frac{1}{6} \log_4 \frac{1}{6} + \frac{4}{6} \log_4 \frac{4}{6} = 0.374185 > IDF(Term_B) = \log_4 \frac{4}{3} = 0.207519 \cdots (2) \end{array} \right.$$

Word Representation

-- Motivating Continuous, Distributed Representations



How do we represent the meaning of a word?

- Dictionary based solutions
- WordNet: a resource containing list of synonyms sets and hypernyms (“is a” relationships)

e.g. synonym sets containing “good”:

```
from nltk.corpus import wordnet as wn
for synset in wn.synsets("good"):
    print "%s" % synset.pos(),
    print ", ".join([l.name() for l in synset.lemmas()])
```

```
(adj) full, good
(adj) estimable, good, honorable, respectable
(adj) beneficial, good
(adj) good, just, upright
(adj) adept, expert, good, practiced,
proficient, skillful
(adj) dear, good, near
(adj) good, right, ripe
...
(adv) well, good
(adv) thoroughly, soundly, good
(n) good, goodness
(n) commodity, trade good, good
```

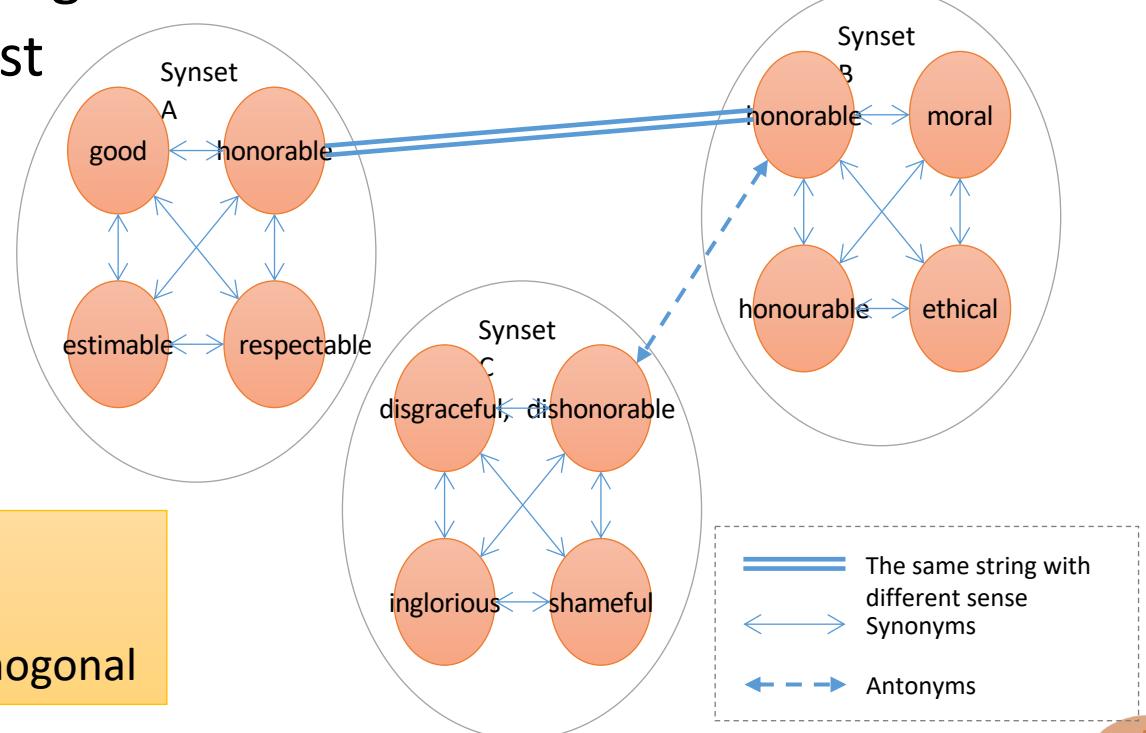
e.g. hypernyms of “panda”:

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(pandaclosure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

How do we represent the meaning of a word?

- Issues
 - context is not considered
 - New words missing
 - Maintenance cost



Hard to measure the similarity!

-- one-hot vector

-- every different words are orthogonal

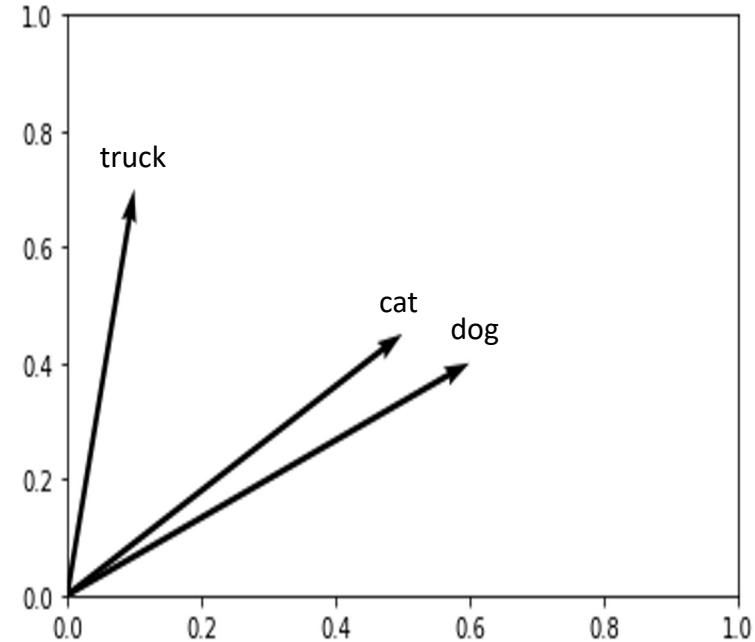
Solution: Continuous, Distributed Representations

By continuous we mean real-valued \mathbb{R}^2

Distributed means a vector in a space, like

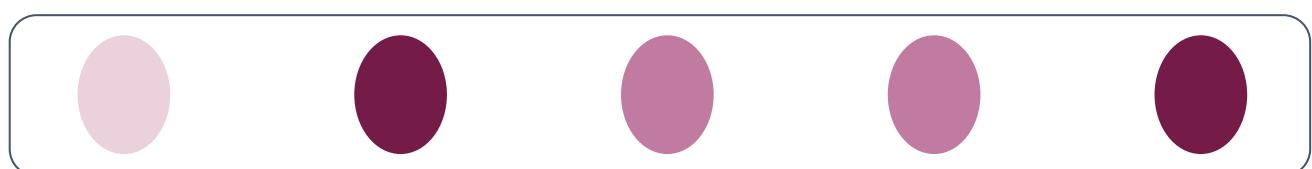
$$\text{dog} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}$$

$$\text{cat} = \begin{bmatrix} 0.5 \\ 0.4 \end{bmatrix}$$



Connection to Neural Networks

A **distributed representation** can be thought of as a pattern of activity “distributed” across neurons. These patterns are taken to represent things - like **concepts** or **words**.



dog =

$$[0.1 \quad 0.3 \quad 0.2 \quad 0.2 \quad 0.3]$$

Deep Learning with Text Today

1. Life: 1844-1900

In the small German village of Röcken bei Lützen, located in a rural farmland area southwest of Leipzig, Friedrich Wilhelm Nietzsche was born at approximately 10:00 a.m. on October 15, 1844. The date coincided with the 49th birthday of the Prussian King, Friedrich Wilhelm IV, after whom Nietzsche was named, and who had been responsible for Nietzsche's father's appointment as Röcken's town minister.

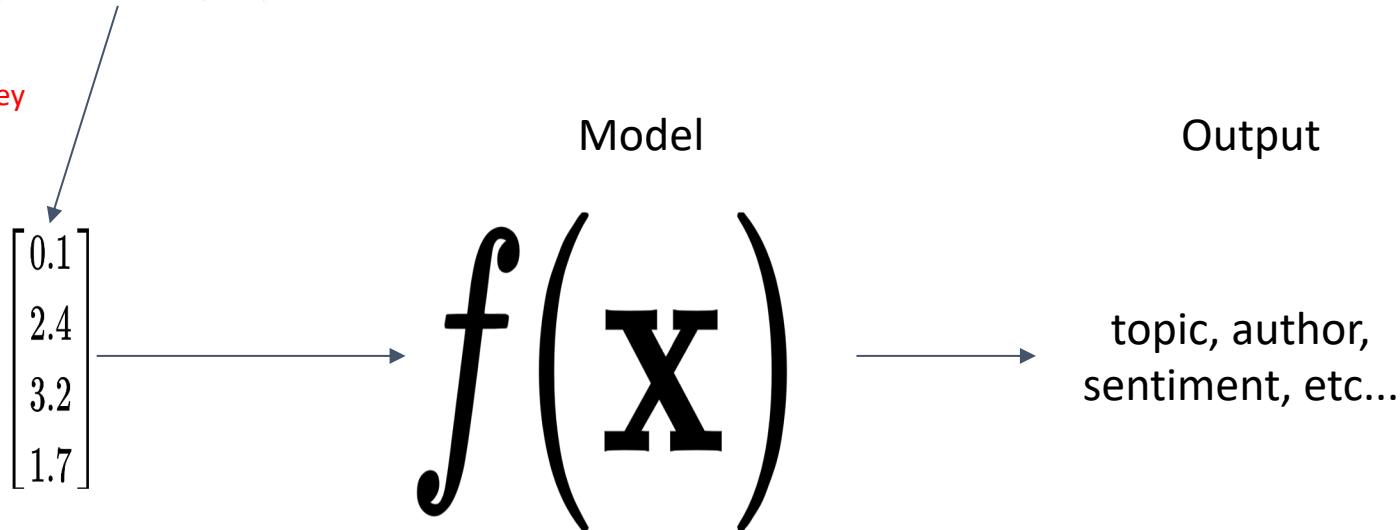
Nietzsche's uncle and grandfathers were also Lutheran ministers, and his paternal grandfather, Friedrich August Ludwig Nietzsche (1756–1826), was further distinguished as a Protestant scholar, one of whose books (1796) affirmed the "everlasting survival of Christianity." Nietzsche's grandparents on both sides were from the Province of Saxony, with his paternal grandfather, paternal grandmother (Erdmuthe Dorothea Krause, 1778–1856), maternal grandfather (David Ernst Ohler, 1787–1859) and maternal grandmother (Johanna Elisabeth Wilhelmine Hahn, 1794–1876) having been born respectively in the small towns of Bibra (just south of Jena), Reichenbach (southeast of Jena), Zeitz (between Jena and Leipzig), and Wehlitz (just northwest of Leipzig).

When Nietzsche was nearly 5 years old, his father, Karl Ludwig Nietzsche (1813–1849) died from a brain ailment (July 30, 1849) and the death of Nietzsche's two-year-old brother, Ludwig Joseph, traumatically followed six months later (January 4, 1850). Having been living only yards away from Röcken's church in the house reserved for the pastor and his family, the Nietzsche family left their home soon after Karl Ludwig's death. They moved to nearby Naumburg an der Saale, where Nietzsche (called "Fritz" by his family) lived with his mother, Franziska (1826–1897), his grandmother, Erdmuthe, his father's two sisters, Auguste and Rosalie (d. 1855 and 1867, respectively), and his younger sister, Therese Elisabeth Alexandra (1846–1935).

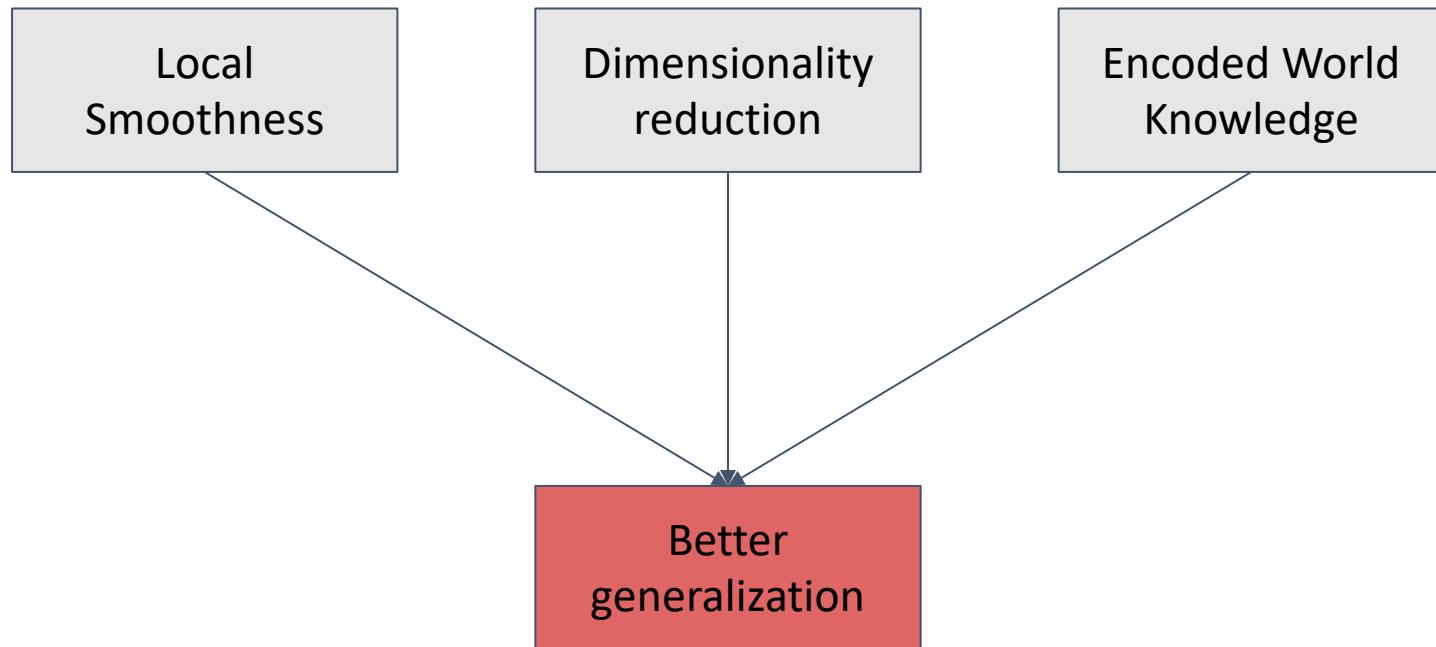
Input

Why vectors as input???

This is a key step



Sketch of Answer



The Old Way

- Words as **atomic units**
- Index in a vocabulary

Problems (more on these soon):

- No notion of **similarity** between words
- Massive number of unique tokens
 - ~150,000 words in Oxford English Dictionary
 - Very high dimensionality
- There is no inherent structure to these vocabulary indices for our algorithms to exploit

```
vocab = {  
    'the': 1,  
    'cat': 2,  
    'licked': 3,  
    'its': 4,  
    'fur': 5,  
    'truck': 6,  
    'dog': 7,  
    'moved': 8}
```



Language Modeling

--Definition

Language models define probability distributions over sequences of words

$$P(w_1, \dots, w_n)$$

Likely combination of words -> high probability

$$P(\text{the cat licked its fur}) \approx 1 \rightarrow \text{high probability}$$

Unlikely combination of words -> low probability

$$P(\text{the truck licked its fur}) \approx 0 \rightarrow \text{low probability}$$



Example Application

We are doing speech recognition and one word is muffled

“The [unsure] licked its fur”

“The [unsure] moved”

We can use our language model to assist our speech recognition module to pick the right word by learning probability distributions such as

$$P(w_2 | w_1 = \text{the}, w_3 = \text{licked}, w_4 = \text{its}, w_5 = \text{fur})$$

$$P(w_2 | w_1 = \text{the}, w_3 = \text{moved})$$



Example Dataset

Our entire vocabulary:

{the, cat, licked, its, fur, truck, dog, moved}

Training samples: {the cat licked its fur ,

the dog moved ,

the truck moved }

Testing samples:

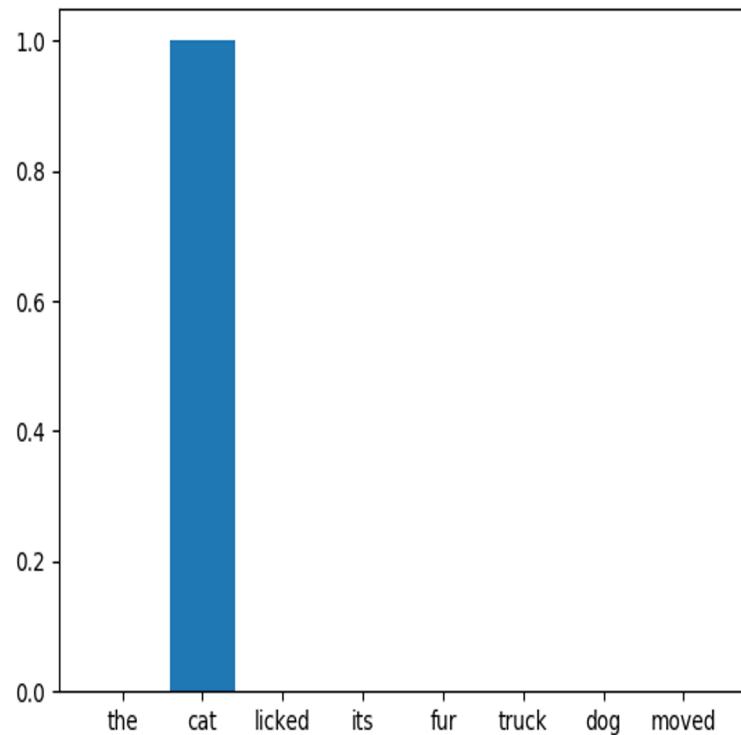
{the dog licked its fur ,

the cat moved }

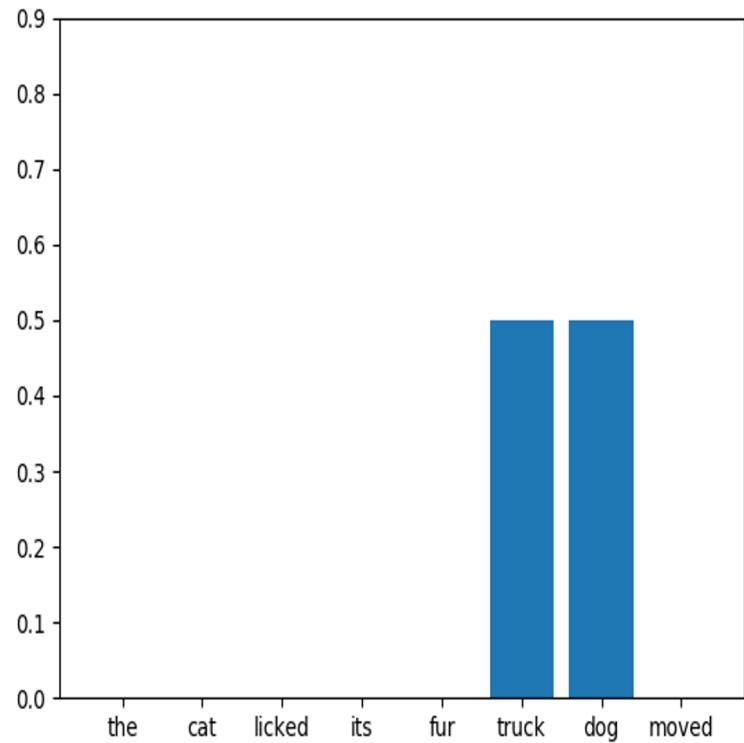


Learned Distributions

$$P(w_2|w_1 = \text{the}, w_3 = \text{licked}, w_4 = \text{its}, w_5 = \text{fur})$$



$$P(w_2|w_1 = \text{the}, w_3 = \text{moved})$$



Problem

There will always be sentences not in our training set

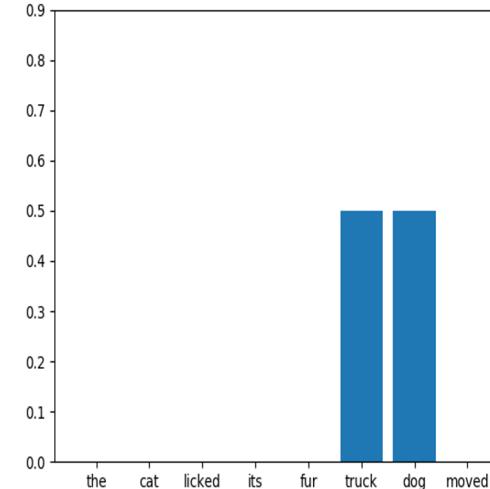
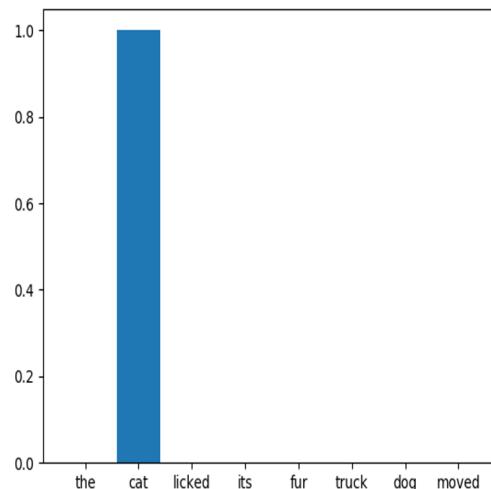
The dog licked its fur

The cat moved

$$P(w_2 = \text{dog} | w_1 = \text{the}, w_3 = \text{licked}, w_4 = \text{its}, w_5 = \text{fur}) = 0$$

$$P(w_2 = \text{cat} | w_1 = \text{the}, w_3 = \text{moved}) = 0$$

If we haven't seen these combinations of words in training, we can't **generalize**



Dimensionality With Atomic Word Representations

- Assume word sequence limit of 10
- Assume 100,000 words in a realistic vocabulary - sample space is $\mathbb{R}^{100,000}$
- How many potential combinations of words?

$$100,000^{10} - 1 = 10^{50} - 1$$

Breakthrough Paper: Bengio et al. 2003: A Neural Probabilistic Language Model

In a nutshell, the idea of the proposed approach can be summarized as follows:



1. associate with each word in the vocabulary a distributed *word feature vector* (a real-valued vector in \mathbb{R}^m),
2. express the joint *probability function* of word sequences in terms of the feature vectors of these words in the sequence, and
3. learn simultaneously the *word feature vectors* and the parameters of that *probability function*.



We can also pre-train our vectors with encoded world knowledge
(e.g. similarity)

Why Does This Work?

1. Similar words are expected to have similar vector representations (and **be closer in vector space**)
2. The probability function is a *smooth* function of feature values
 - Small changes in the feature values will induce a small change in the value of the function
 - This is *not* true for unorganized discrete spaces, where small changes in input can lead to large changes in the function value
3. Therefore, **the presence of one sentence in the training set can effectively distribute probability density in the vector space for a combinatorial number of unseen sentences**



Example

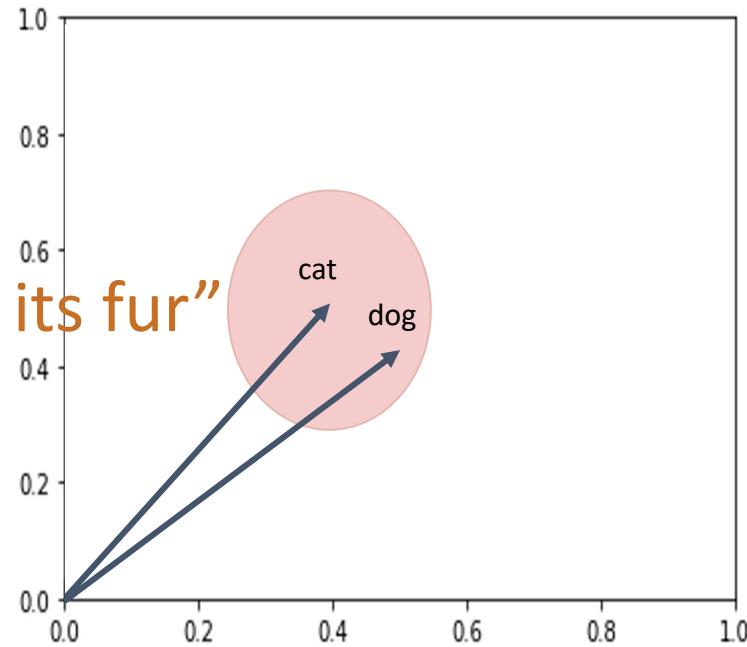
In training we see: “The cat licked its fur”

For the conditional probability

$$P(w_2 | w_1 = \text{the}, w_3 = \text{licked}, w_4 = \text{its}, w_5 = \text{fur})$$

We somehow* distribute probability density around cat.

And because “dog” is similar (close), it also gets the benefit of this probability density. The same goes for many furry animals close together in our space.



* It is highly recommended to read the paper for full details.

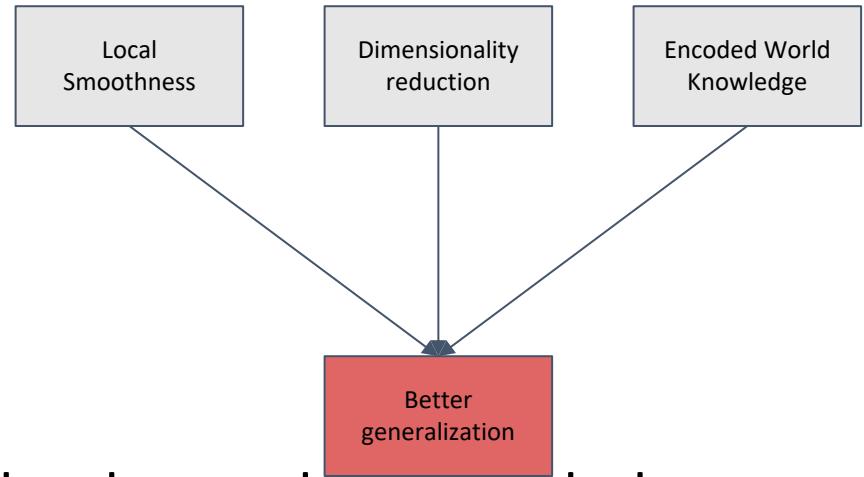


Contrast with Discrete Index Space

Assume just eight words in the vocab and a max sequence length of 2. Training probability mass in (2, 3) tells us nothing about (6, 3) - the indices are arbitrary, and we have no inherent notion of similarity

8: moved									
7: truck									
6: dog									
5: fur									
4: its									
3: licked		cat licked					dog licked		
2: cat									
1: the									
	1: the	2: cat	3: licked	4: its	5: fur	6: dog	7: truck	8: moved	

Why Vectors?



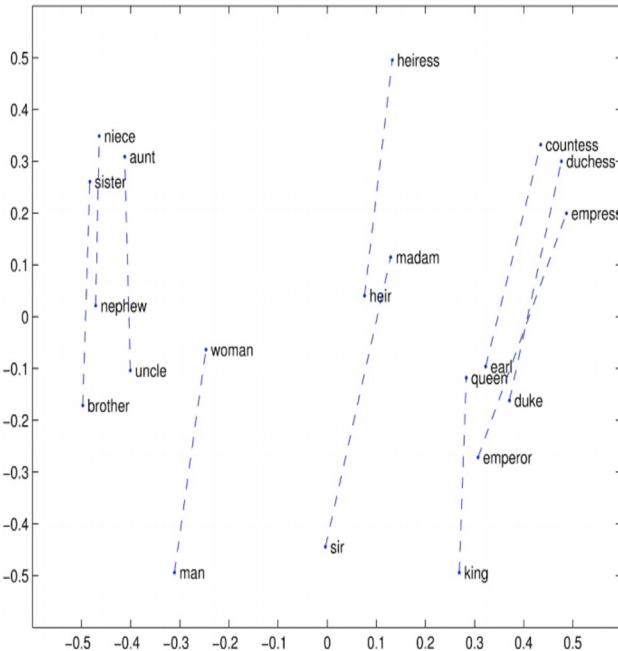
Continuous values:

- Give our learned functions local smoothness, such that small changes in input induces small changes in the function value, aiding generalization

Distributed (vector) representations facilitate:

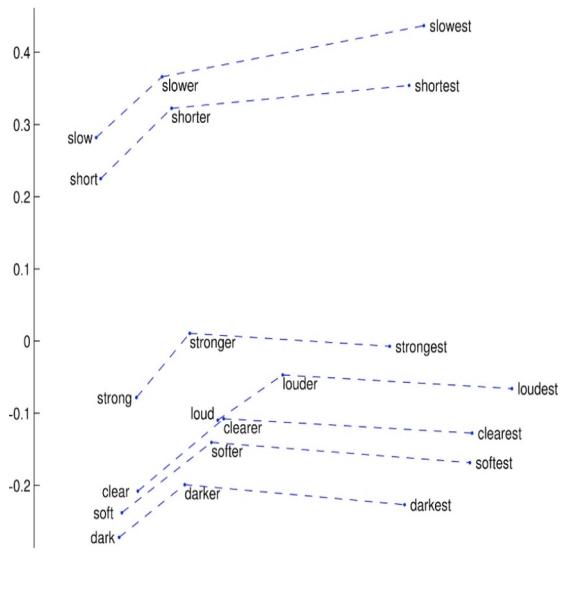
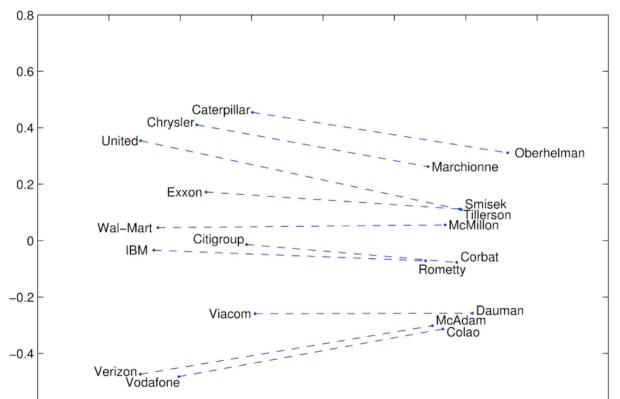
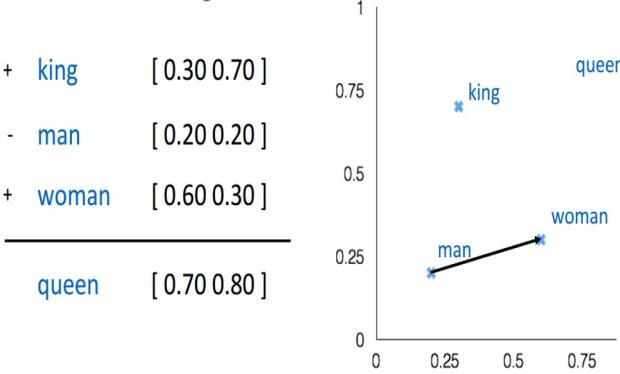
- Dimensionality reduction - overcoming the curse of dimensionality
- Encoding world knowledge into the vector space
 - Locating similar words close together in the learned vector space
 - Other forms of world knowledge, e.g. latent hierarchies (we will look at later)

Learning Vector Representation of Words



man:woman :: king:?

+ king	[0.30 0.70]
- man	[0.20 0.20]
+ woman	[0.60 0.30]
<hr/>	
queen	[0.70 0.80]



- o. frog
- 1. frogs
- 2. toad
- 3. litoria
- 4. leptodactylidae
- 5. rana
- 6. lizard
- 7. eleutherodactylus



3. litoria



4. leptodactylidae

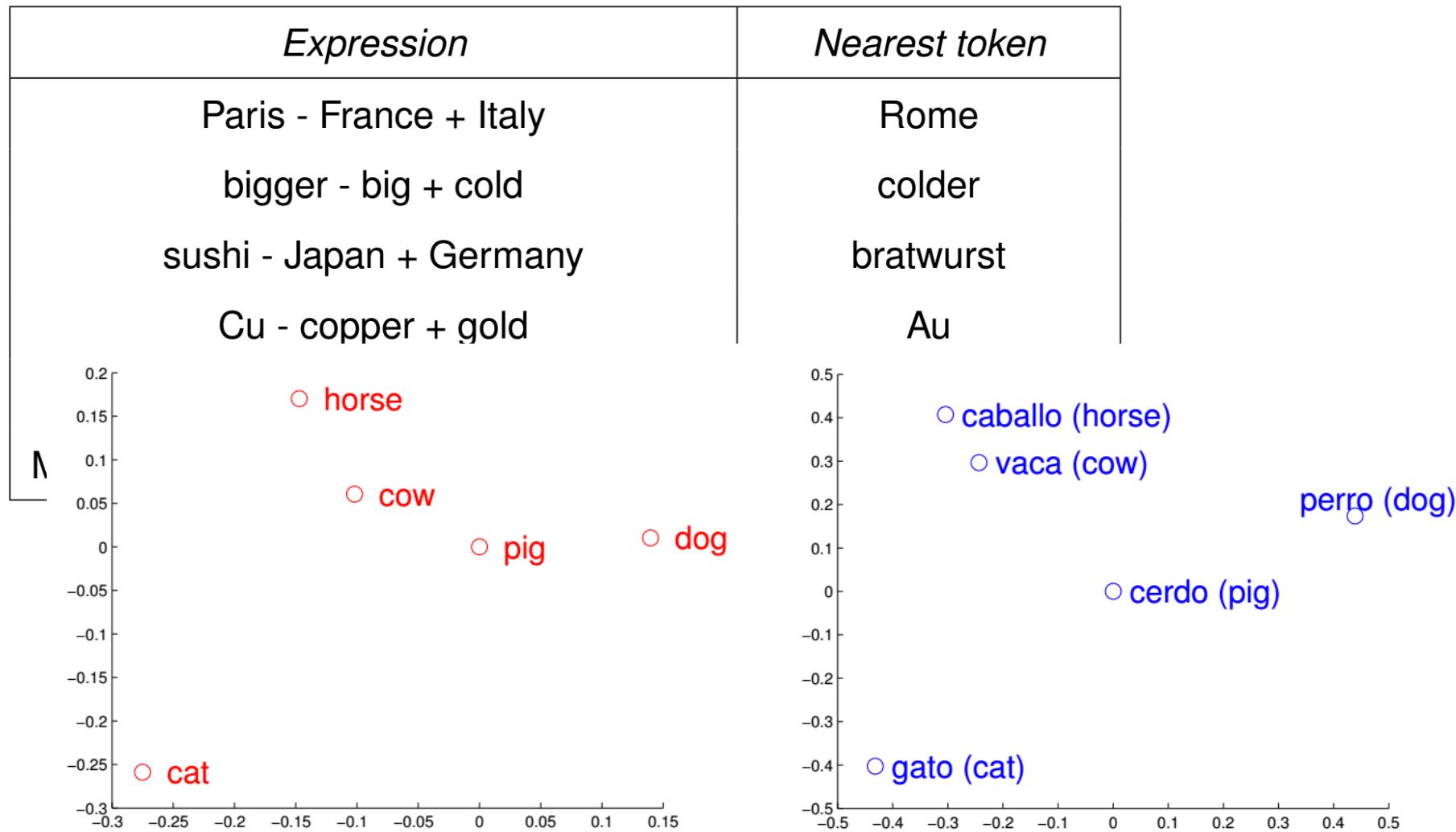


5. rana



7. eleutherodactylus

Representation Vector Usage



Distributional Hypothesis

“A word is characterized by the company it keeps.” (Firth 1957)

In other words, similar words will appear in similar contexts

The cat licked its fur

The dog licked its fur

No surprise “cat” and “dog” both appear near “lick” and “fur”. We should find they also often appear near “eat”, “run”, “bite” and so on... but not near words like “read”, “sophisticated”, or “wheel”.

...government debt problems turning into banking crises as happened in 2009...

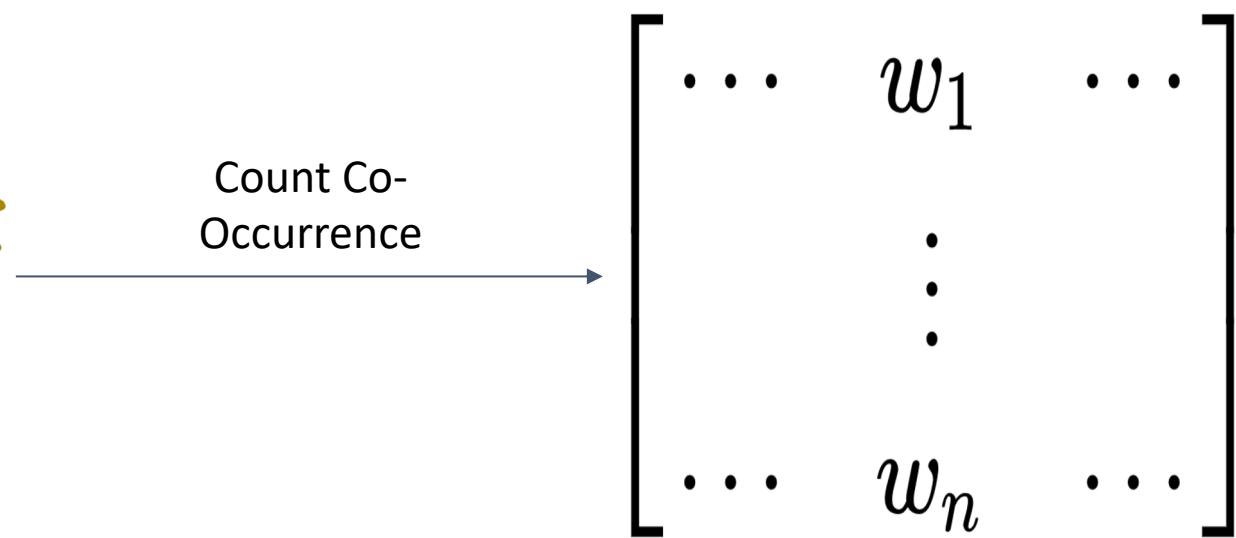
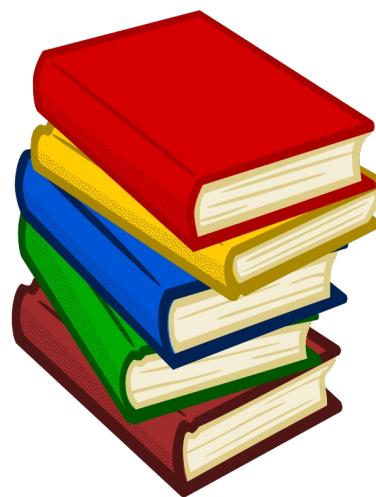
...saying that Europe needs unified banking regulation to replace the hodgepodge...

...India has just given its banking system a shot in the arm...



Latent Semantic Analysis

Using a corpus of text as input, draw a window of defined length around each word and count co-occurrence statistics. The resulting matrix contains our word vectors.



Window-Based Co-Occurrence

Window = 1

The cat licked its fur. The truck moved.

A horizontal row of four empty boxes. The first two boxes are highlighted with a teal border, representing a window of size 1 centered on the word 'cat'.

The cat licked its fur. The truck moved.

A horizontal row of five empty boxes. The first three boxes are highlighted with a teal border, representing a window of size 1 centered on the word 'licked'.

The cat licked its fur. The truck moved.

A horizontal row of five empty boxes. The last two boxes are highlighted with a teal border, representing a window of size 1 centered on the word 'truck'.

⋮



Co-Occurrence Matrix

	the	cat	licked	its	fur	truck	moved
the	0	1	0	0	0	1	0
cat	1	0	1	0	0	0	0
licked	0	1	0	1	0	0	0
its	0	0	1	0	1	0	0
fur	0	0	0	1	0	0	0
truck	1	0	0	0	0	0	1
moved	0	0	0	0	0	1	0

The vectors in
this matrix
represent our
words



Simple example for word co-occurrence usage

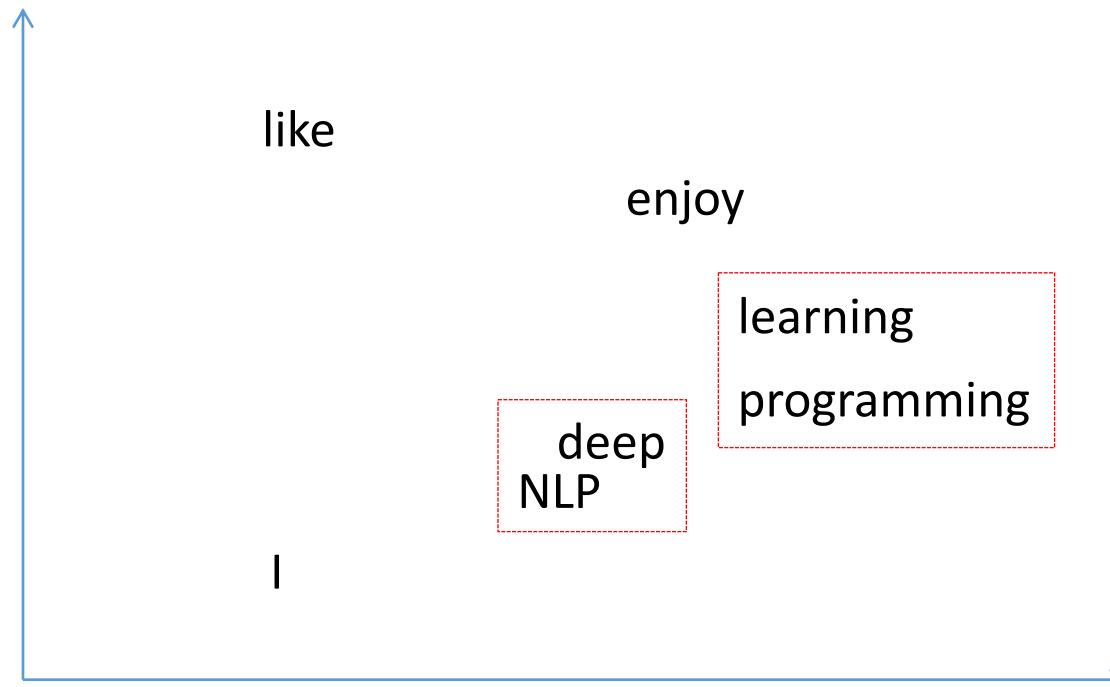
- Training corpus
 - “I like deep learning.”, “I like NLP.”, “I enjoy programming.”

cooccurrence	I	like	enjoy	deep	learning	NLP	programming
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
programming	0	0	1	0	0	0	0



Simple example for word co-occurrence usage

- SVD on this co-occurrence matrix
 - Reduce dimension
- Use the 2 biggest singular value to represent words



Latent Semantic Indexing(LSI)

- A statistical technique
- Uses linear algebra technique called *singular value decomposition (SVD)*
 - attempts to estimate the hidden structure that generates terms given concepts
 - discovers the most important associative patterns between words and concepts
- Data driven
 - A large collection of sentences or documents is employed

LSI and Text Documents

- Let X denote a term-document matrix

$$X = [x_1 \dots x_n]^T$$

- each row is the vector-space representation of a document
- each column contains occurrences of a term in each document in the dataset
- Latent semantic indexing
 - compute the SVD of X : $X = U\Sigma V^T$
 - Σ - singular value matrix (*diagonal matrix*)
 - set to zero all but largest K singular values - $\hat{\Sigma}$
 - obtain the reconstruction of X by:

$$\hat{X} = U\hat{\Sigma}V^T$$

LSI Example

- A collection of documents:

d1: Indian government goes for open-source software

Linux OS

d2: Debian 3.0 Woody released

d3: Wine 2.0 released with fixes for Gentoo 1.4 and Debian 3.0

d4: gnuPOD released: iPOD on Linux... with GPLed software

d5: Gentoo servers running at open-source mySQL database

d6: Dolly the sheep not totally identical clone

d7: DNA news: introduced low-cost human genome DNA chip

d8: Malaria-parasite genome database on the Web

d9: UK sets up genome bank to protect rare sheep breeds

d10: Dolly's DNA damaged

Genome news

LSI Example

- The term-document matrix X^T

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
open-source	1	0	0	0	1	0	0	0	0	0
software	1	0	0	1	0	0	0	0	0	0
Linux	0	0	0	1	0	0	0	0	0	0
released	0	1	1	1	0	0	0	0	0	0
Debian	0	1	1	0	0	0	0	0	0	0
Gentoo	0	0	1	0	1	0	0	0	0	0
database	0	0	0	0	1	0	0	1	0	0
Dolly	0	0	0	0	0	1	0	0	0	1
sheep	0	0	0	0	0	1	0	0	0	0
genome	0	0	0	0	0	0	1	1	1	0
DNA	0	0	0	0	0	0	2	0	0	1

LSI Example

- The reconstructed term-document matrix \hat{X}^T after projecting on a subspace of dimension K=2
- $\Sigma = \text{diag}(2.57, 2.49, 1.99, 1.9, 1.68, 1.53, 0.94, 0.66, 0.36, 0.10)$
- $\hat{\Sigma} = \text{diag}(2.57, 2.49, 0, 0, 0, 0, 0, 0, 0, 0)$

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	
open-source	0.34	0.28	0.38	0.42	0.24	0.00	0.04	0.07	0.02	0.01	
software	0.44	0.37	0.50	0.55	0.31	-0.01	-0.03	0.06	0.00	-0.02	
Linux	0.44	0.37	0.50	0.55	0.31	-0.01	-0.03	0.06	0.00	-0.02	
released	0.63	0.53	0.72	0.79	0.45	-0.01	-0.05	0.09	-0.00	-0.04	
Debian	0.39	0.33	0.44	0.48	0.28	-0.01	-0.03	0.06	0.00	-0.02	
Gentoo	0.36	0.30	0.41	0.45	0.26	0.00	0.03	0.07	0.02	0.01	
database	0.17	0.14	0.19	0.21	0.14	0.04	0.25	0.11	0.09	0.12	
Dolly	-0.01	-0.01	-0.01	-0.02	0.03	0.08	0.45	0.13	0.14	0.21	
sheep	-0.00	-0.00	-0.00	-0.01	0.03	0.06	0.34	0.10	0.11	0.16	
genome	0.02	0.01	0.02	0.01	0.10	0.19	1.11	0.34	0.36	0.53	
DNA	-0.03	-0.04	-0.04	-0.06	0.11	0.30	1.70	0.51	0.55	0.81	
database	0	0	0	0	1	0	0	1	0	0	

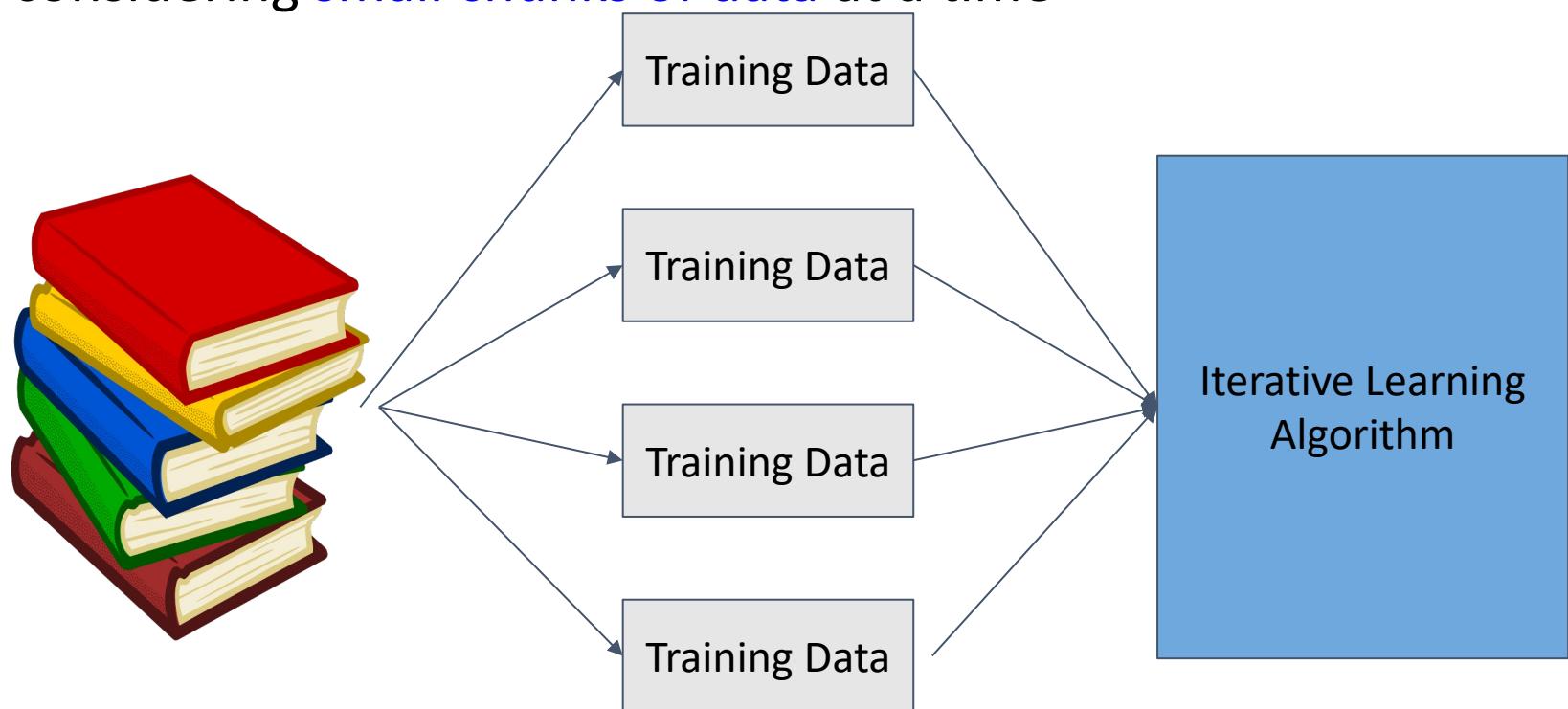
Problems

- Original matrix still has same dimension as our vocabulary size - potentially $100,000 \times 100,000$
- Matrix is extremely sparse
- We can perform Singular Value Decomposition (SVD) for dimensionality reduction, however at quadratic computational cost
- Adding a new word changes the entire matrix



Word2vec

Word2vec is an *iterative* training method: working with a huge matrix is cumbersome - instead, learning proceeds by considering **small chunks of data** at a time



Recipe

Ingredients:

1. A probabilistic model
2. Small chunks of data for training
3. Model parameters

Iterative process:

1. Generate probability estimates
2. Calculate the error in the estimates
3. Update the parameters until optimized



Probabilistic Model: Some Choices

Unary language model

$$P(w_1, \dots, w_n) = \prod_i P(w_i)$$

Ridiculous not to consider word order

Binary language model

$$P(w_1, \dots, w_n) = \prod_i P(w_i | w_{i-1})$$

Better but still limited by short context distance

word2vec models (using window m to get more context)

Continuous Bag of Words (CBOW)

$$P(w_c | w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m})$$

The cat [center word] its fur

Skip-Gram (which we will focus on)

$$P(w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m} | w_c)$$

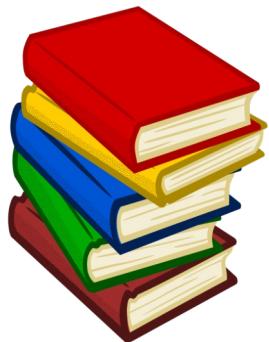
[left context] licked [right context]



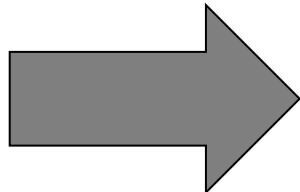
Skip-Gram: Training Data

Training Word Pairs

Corpus



The cat licked its fur. The truck moved.



Context window
size = 1

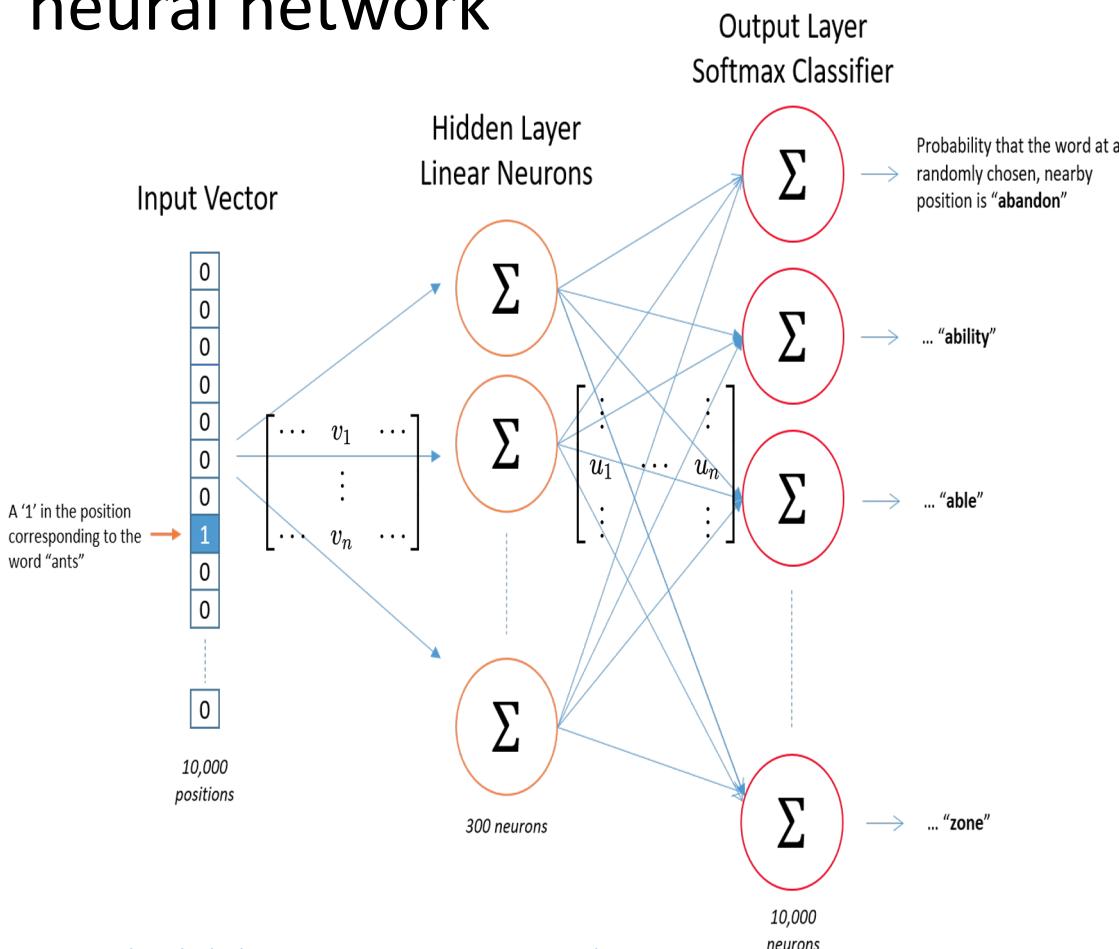
What if context size > 1?

- More word pairs
- Only pass two words to the model at a time

Center	Context
the	cat
the	truck
cat	the
cat	licked
licked	cat
licked	its
...	...

Model Parameters

word2vec uses a single hidden layer feedforward neural network



Input to hidden layer matrix has our target word embeddings v_i

Hidden to output layer matrix has another set of word embeddings called output embeddings u_i

Recap: Skip-Gram

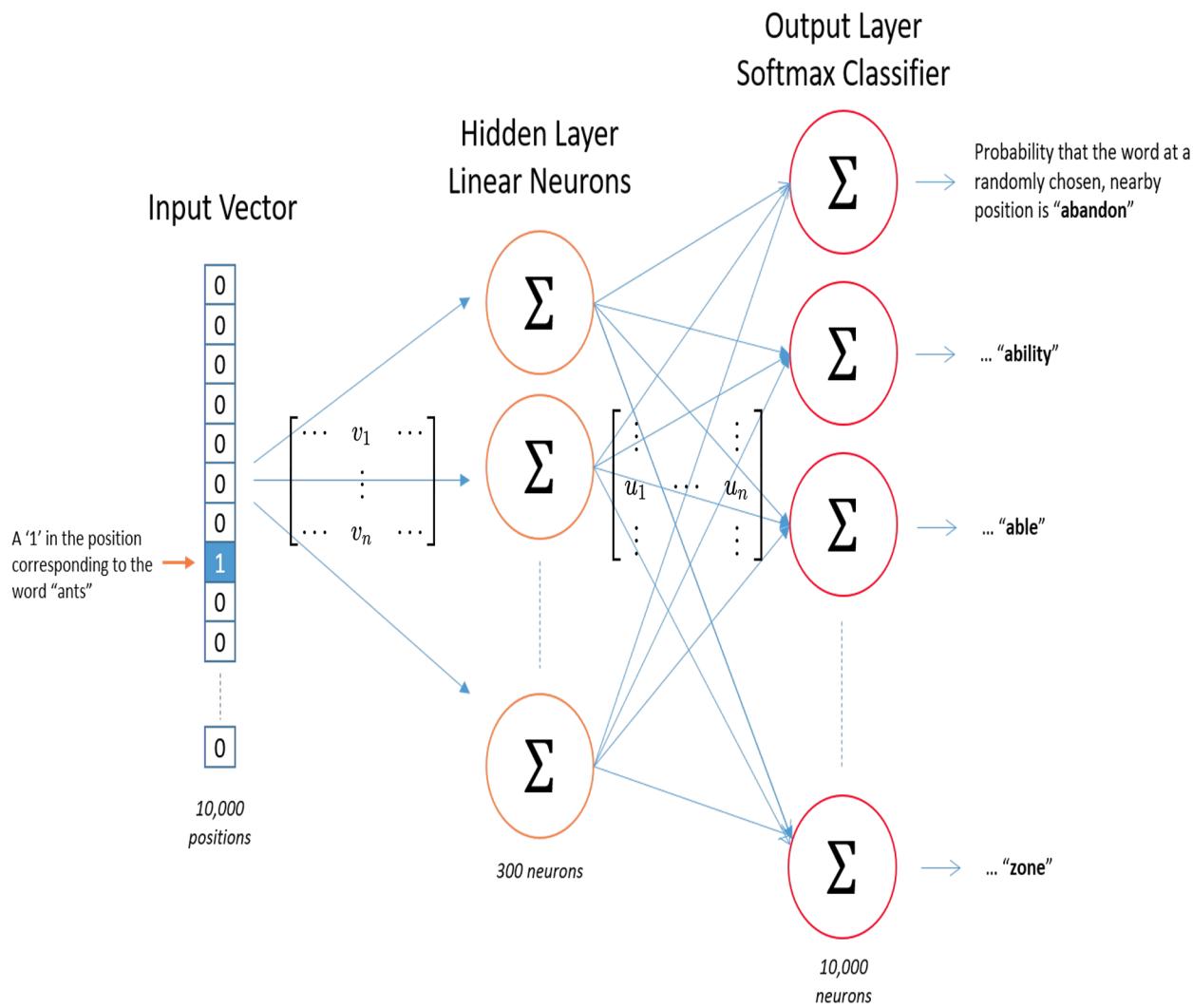
Ingredients:

1. A probabilistic model $P(w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m} | w_c)$
2. Small chunks of data for training Training word pairs
3. Model parameters Neural network parameters **v** and **u**

Iterative process:

1. Generate probability estimates
2. Calculate the error in the estimates
3. Update the parameters until optimized

Generate Probability Estimates



(1) Input vector selects input embedding from hidden layer matrix

(2) Softmax over multiplication with output matrix creates probability distribution over the vocabulary

This should be high for context words, low for others

Network Input

We imagine the words are encoded as “one-hot” vectors (all zeros except a one at the word index in the embedding matrix)

```
vocab = {  
    'the': 1,  
    'cat': 2,  
    'licked': 3,  
    'its': 4,  
    'fur': 5,  
    'truck': 6,  
    'dog': 7,  
    'moved': 8}
```

$$\text{the} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{licked} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{etc...}$$



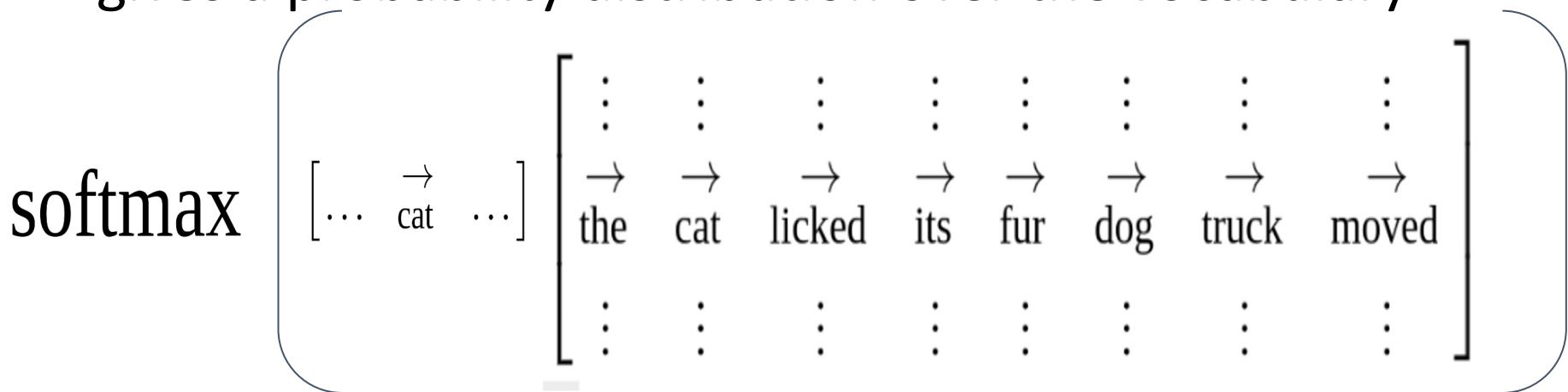
Embedding Lookup

In practice we don't do matrix multiplication, but just use the word index to pick out the vector.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dots & \xrightarrow{\hspace{1cm}} & \dots \\ \dots & \xrightarrow{\hspace{1cm}} & \dots \end{bmatrix} \quad \equiv \quad \begin{bmatrix} \dots & \xrightarrow{\hspace{1cm}} & \dots \end{bmatrix}$$

Probability Over Vocab

Our word vector does a dot product with every word's output embedding, then applying softmax gives a probability distribution over the vocabulary



$$\text{probability word } w_i \text{ in context} = \hat{y}_i = P(w_i | \mathbf{v}_c, \mathbf{U}) = \frac{\exp(\mathbf{u}_{w_i}^T \mathbf{v}_c)}{\sum_{j=1}^V \exp(\mathbf{u}_{w_j}^T \mathbf{v}_c)}$$

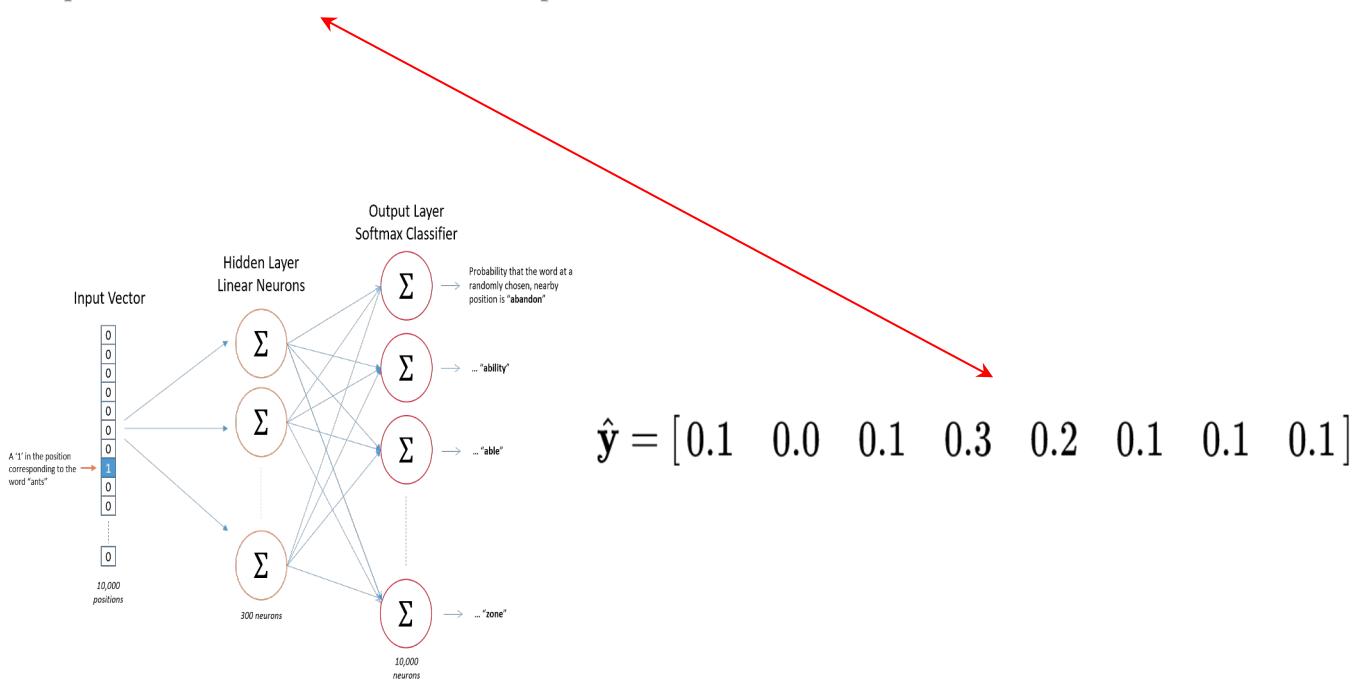
Calculate Error in Estimates

Training Pair:

$$\text{center word} = \text{cat} = [0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] = \text{input}$$

$$\text{context word} = \text{licked} = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0] = \text{target}$$

$$[0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$$



Loss Function

Cross entropy measures the distance between probability distributions

$$\text{CE}(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_{i=1}^V y_i \log(\hat{y}_i)$$

$$\sum \log \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ [0.1 & 0.0 & 0.1 & 0.3 & 0.2 & 0.1 & 0.1 & 0.1] \end{bmatrix} *$$

Update Parameters

$$\begin{aligned}
 \frac{\partial}{\partial v_c} \text{CE}(y, \hat{y}) &= \frac{\partial}{\partial v_c} - \sum_i y_i \log(\hat{y}_i) \\
 &= \frac{\partial}{\partial v_c} - \sum_i y_i \log\left(\frac{\exp(u_i^T v_c)}{\sum_j \exp(u_j^T v_c)}\right) \\
 &\quad \vdots \\
 &= -u_o + \sum_k \hat{y}_k u_k \\
 &= (\hat{y} - y) U^T
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial}{\partial u_w} \text{CE}(y, \hat{y}) &= \frac{\partial}{\partial u_w} - \sum_i y_i \log(\hat{y}_i) \\
 &= \frac{\partial}{\partial u_w} - \sum_i y_i \log\left(\frac{\exp(u_i^T v_c)}{\sum_j \exp(u_j^T v_c)}\right) \\
 &\quad \vdots \\
 &= \begin{cases} (\hat{y}_w - 1)v_c & \text{if } w = i, \\ \hat{y}_w v_c & \text{if } w \neq i \end{cases} \\
 &= ((\hat{y} - y)^T v_c)^T
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial}{\partial U} J_{\text{skip-gram}} &= \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial F(w_{c+j}, v_c)}{\partial U} \\
 \frac{\partial}{\partial v_c} J_{\text{skip-gram}} &= \sum_{-m \leq j \leq m, j \neq 0} \frac{\partial F(w_{c+j}, v_c)}{\partial v_c} \\
 \frac{\partial}{\partial v_j} J_{\text{skip-gram}} &= 0, \text{ for all } j \neq c
 \end{aligned}$$

Update Parameters

Or just use PyTorch's auto differentiation!

```
def optimize(self, loss):
    loss.backward() # automatically calculates gradients of params wrt loss
    self.optimizer.step()
    self.optimizer.zero_grad()
```

Recap: Skip-Gram

Ingredients:

1. A probabilistic model $P(w_{c-m}, \dots, w_{c-1}, w_{c+1}, \dots, w_{c+m} | w_c)$
2. Small chunks of data for training Training word pairs
3. Model parameters Neural network parameters **V** and **U**

Iterative process:

1. Generate probability estimates Forward pass of neural network
2. Calculate the error in the estimates $\text{CE}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^V y_i \log(\hat{y}_i)$
3. Update the parameters until optimized Use auto differentiation and SGD

Problem

Can you see how this is not computationally efficient?

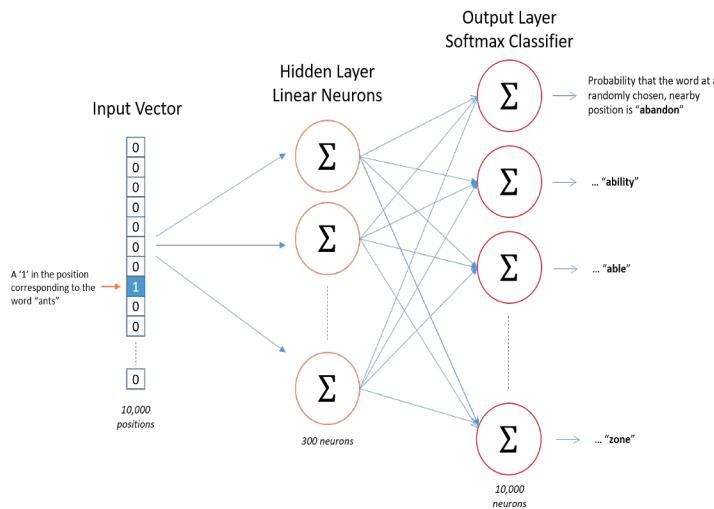
$$\text{probability word } w_i \text{ in context} = \hat{y}_i = P(w_i | \mathbf{v}_c, \mathbf{U}) = \frac{\exp(\mathbf{u}_{w_i}^T \mathbf{v}_c)}{\sum_{j=1}^V \exp(\mathbf{u}_{w_j}^T \mathbf{v}_c)}$$

That's a lot of inner products: 100,000 on each update
(and there will be millions of updates in a serious implementation)

Solution

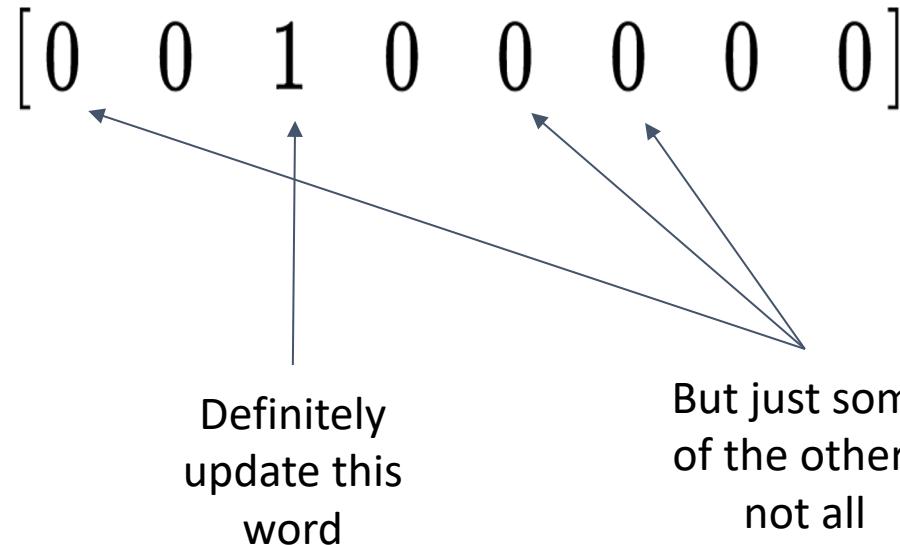
We don't actually care about the output probability distribution being precise - our objective is the **word embeddings**, not a language model

We can therefore focus on ensuring the model can distinguish targets (**context words**) from noise (**non-context words**)



Negative Sampling

Instead of updating the vectors for every word in the vocabulary for each word pair seen in training, just update the target and a few negative samples (randomly sampled non-context words)



Negative Sampling

Choose negative samples by accounting for word frequency, for which we use the function $f(w_i)$. The 3/4th power is an empirical result.

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^n (f(w_j)^{3/4})}$$

How much more efficient is this?

- Assume $V = 100,000$, $d = 300$ (quite normal)
- Take 5 negatives + the target per iteration = 1,800 parameters to update
- That's 0.06% of the original 3 million parameters

Sub-Sampling Frequent Words

- Words like “the” occur very frequently and carry little meaning
- Define a probability for not considering a word
- Apply this in pre-processing (when selecting training pairs)

$$P(w_i) = 1 = \sqrt{\frac{t}{f(w_i)}}$$

- The parameter t is often set to 0.001

How is Similarity Learned?

Remember that one of our goals was for similar words to be close in vector space

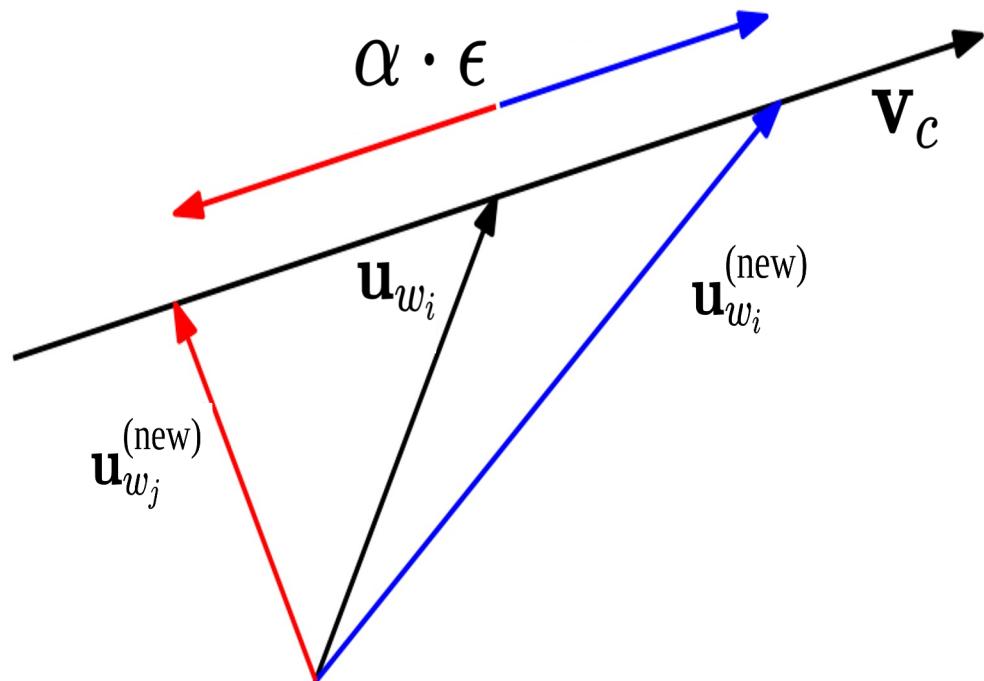
$$\text{probability word } w_i \text{ in context} = \hat{y}_i = P(w_i | \mathbf{v}_c, \mathbf{U}) = \frac{\exp(\mathbf{u}_{w_i}^T \mathbf{v}_c)}{\sum_{j=1}^V \exp(\mathbf{u}_{w_j}^T \mathbf{v}_c)}$$

- The inputs to our softmax function are dot products
- The dot product measures the similarity between vectors
- Therefore optimizing this objective will maximize the similarity of vectors in shared contexts, and minimize the similarity of other vectors



Learning Similarity

A geometric view of output vector updates* shows visually how the vectors for words in shared contexts are made closer through training, and non-context vectors are made more dissimilar



* similar but more complicated for input vectors - image from <https://arxiv.org/pdf/1705.10359.pdf>

Starter Code and Instructions

Go to <https://github.com/IKMLab/skipgram> to access the starter code

```
: import numpy as np
import torch
from torch import nn
from torch.autograd import Variable
import nltk
import pandas as pd
from data import toxic
import os
import glovar
```

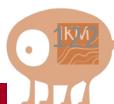
Skip-Gram Implementation Practice

Your goal is to implement the Skip-Gram model in PyTorch, including pre-processing. Pre-processing is an important step in deep learning with text, and you should learn it now. This tutorial assumes you are familiar with the basics of PyTorch. If not, you can review some introductory tutorials such as:

<https://github.com/cjohansson/pytorch-examples>

Steps in this tutorial:

1. Import the data
2. Tokenize the data
3. Build the vocab dictionary
4. Prepare training pairs
5. Implement negative sampling
6. Code the model
7. Train the model
8. Visualize the word embeddings



GloVe (Global Vectors)

- word2vec is excellent at learning similarity and linear regularities
- But it doesn't make full use of co-occurrence statistics
- GloVe improves word2vec by considering global statistical information

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

Computer Science Department, Stanford University, Stanford, CA 94305

jpennin@stanford.edu, richard@socher.org, manning@stanford.edu



FastText

FastText is much newer and has also seen good results.

Considers character-level features, and can thus take a guess and generate vectors for out-of-vocabulary words, taking advantage of patterns in morphology

Pre-trained vectors are available in many languages:

<https://fasttext.cc/>

fastText

Library for efficient text classification and representation learning



What Vectors Should I Use?

--It depends.

GloVe is very widely used with good results.

FastText is new and adoption may be slow, but can also achieve good results.

Training your own with word2vec may be indicated when you have many “out-of-vocabulary” words - i.e. your dictionary has lots of words for which there are no pre-trained vectors.

This can happen in special areas - for example when dealing with classical Chinese text.

A good idea is to try different vectors on specific tasks.

Evaluation: Example dataset: WordSim353

<http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/>

