Introduction to NLP

Dr Amit Sethi, IIT Bombay

Module objectives

- Revise what is NLP
- Understand some key problems in NLP
- Appreciate earlier frameworks used for NLP
- Some example solutions to NLP problems

Outline

- NLP basics
- Pre-processing in NLP
- Language model with an example
- From words to vectors
- Some applications

What is Natural Language Processing?

- NLP is analysis or generation of natural language text using computers, for example:
  - Machine translation
  - Spell check (autocorrect)
  - Automated query answering
  - Speech parsing (a problem that overlaps with ASL)
- NLP is based on:
  - Probability and statistics
  - Machine learning
  - Linguistics
  - Common sense

Why do NLP?

- Language is one of the defining characteristics of our species
- A large body of knowledge can be organized and easily accessed using NLP
- Original conception of the Turing test was based on NLP

A few types of problems in NLP

- Text classification or regression
- Named entity recognition
- Parse text (syntax)
- Semantics understanding
- Text synthesis
- Reasoning

Example: text classification

- Sentiment analysis – positive or negative
  - “This is a ridiculously priced toothbrush. Seriously, no way to get around it. It is absurdly priced and I'm almost embarrassed to be admitting that I bought it. With that said... Wow, this thing is amazing.”
  - “These pens make me feel so feminine and desirable. I can barely keep the men away when I'm holding one of these in my dainty hand. My husband has started to take fencing lessons just to keep the men away.”

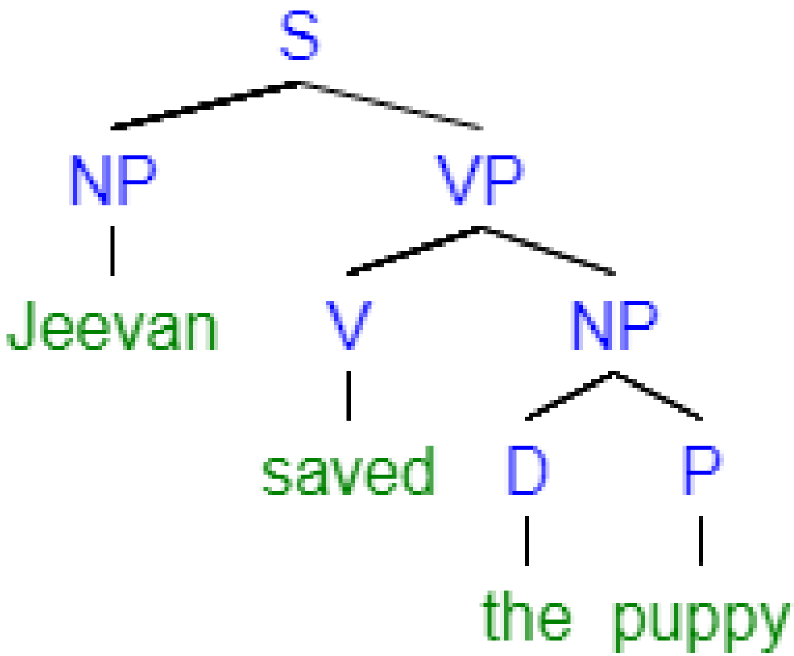
Text source: Amazon.com reviews

Example: Named entity recognition

- A real-world person, place, or object that can be given a proper noun:
  - "**India** posted a score of 256/8 in their allotted 50 overs in the third and deciding ODI of the series. **Virat Kohli** was the top-scorer for men in blue with a classy 71, while **Adil Rashid** and **David Willey** picked up three wickets each."
  - India → Place, Virat Kohli → Person, …

Text source: Reuters

Example Parsing text



Tool Source: http://mshang.ca/syntree/

Example semantics understanding

- "We were on a crash course."
- **Crash** can mean an accident, a percussion strike, or a collapse.
- **Course** can mean a study plan, or a path.

Challenges in NLP

- Large vocabulary
- Multiple meanings
- Many word forms
- Synonyms
- Sarcasm, jokes, idioms, figures of speech
- Fluid style and usage

Basic text classification using ML

Text sample
- Variable number of words

Pre-processing
- Tokenization, normalization, etc.

Feature
- Fixed-length vector

Class
- Discrete set

Some standard terms

- Corpus: A body of text samples
- Document: A text sample
- Vocabulary: A list of words used in the corpus
- Language model: How the words are organized

Outline

- NLP basics
- Pre-processing in NLP
- Language model with an example
- From words to vectors
- Some applications

Tokenization

- Chopping up text into pieces called *tokens*
- Usually, each word is a token
  - Jeevan / saved / the / puppy
- How do you tokenize?
  - Split up at all non-alpha-numeric characters
  - What about apostrophes?
  - What about two-word entities, e.g. “New Delhi”?
- What about compound words in Sanskrit and German?

Stop words

- Words that are common
- Non-selective (excluding negation)
- Examples:
  - Articles: a, an, the
  - Common verbs: is, was, are
  - Pronouns: he, she, it
  - Conjunctions: for, and
  - Prepositions: at, on, with
- Need not be used to classify text

Normalization

- Words appear in many forms:
  - School, school, schools
  - U.S.A, USA, U.S., US
    - But not “us”
  - Windows vs. windows/window
- These need not be considered separate terms
- Normalization is counting equivalent forms as one term

Stemming and Lemmatization

- Stemming – chopping off the end of words
  - *Nannies* becomes *nanni* (Rule: *.ies* → *.i*)
  - *Caresses* becomes *caress* (Rule: *.sses* → *.ss*)
  - This is a heuristic way
- Finding the lemma of a word is the more exact task
  - *Nannies* should become *nanny*
  - *Privatization* should become *private*

greatlearning
Learning for Life

Word vectors

- “India posted a score of 256/8 in their allotted 50 overs in the third and deciding ODI of the series. Virat Kohli was the top-scorer for men in blue with a classy 71, while Adil Rashid and David Willey picked up three wickets each”
- One-hot encoding (or 1-of-N encoding)

$$
\begin{array}{ccc}
Vocab \downarrow & India & posted \ldots \\
\begin{array}{c} a \\ India \\ posted \\ score \end{array} & \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \ldots
\end{array}
$$

Text source: Reuters

Bag-of-words as a feature

- "India posted a score of 256/8 in their allotted 50 overs in the third and deciding ODI of the series. Virat Kohli was the top-scorer for men in blue with a classy 71, while Adil Rashid and David Willey picked up three wickets each"

- Counts
  - $\begin{array}{c} India \\ Posted \\ Score \end{array} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$
- The counts can be normalized
- The words can be standardized
  - Score
  - Scorer
- What about uninformative words?

Text source: Reuters

TF-IDF as a feature

- Term frequency – inverse document frequency
- TF $f_{t,d}$ is the count of term *t* in document *d*
  - Usually normalized in some sense
  - $\mathrm{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$
- IDF penalizes terms that occur often in all documents, e.g. “the”
  - $\mathrm{idf}(t, D) = \log \frac{|D|}{1+|\{d \in D : t \in d\}|}$
- TF-IDF is $\mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$
- Form a vector of TF-IDF for various terms
  - Which terms?

Examples of TF-IDF

- Let us assume that the word ***dog*** appears four times in a document of 1000 words
  - TF = 4/1000 = 4 × 10^{-3} = 0.004
- Let the same word appear 50 times in 1 million documents
  - IDF = log (1000000 / 50) = 4.3
- So, TF-IDF = 0.004 × 4.3 = 0.0172

- Let us assume that the word ***is*** appears 50 times in a document of 1000 words
  - TF = 50/1000 = 50 × 10^{-3} = 0.05
- Let the same word appear 40,000 times in 1 million documents
  - IDF = log (1000000 / 40000) = 1.398
- So, TF-IDF = 0.05 × 1.398 = 0.0699

Without IDF, ***dog*** would not be able to compete with ***is***.

We can then use traditional ML methods

- Text: “India posted a score of 256/8 in their allotted 50 overs in the third and deciding ODI of the series. Virat Kohli was the top-scorer for men in blue with a classy 71, while Adil Rashid and David Willey picked up three wickets each”
- Add word vectors: $\begin{array}{r} cat \\ dog \\ Kohli \\ score \\ zero \end{array} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ 0 \end{bmatrix}$
- Topic: “Cricket”

Text source: Reuters

Outline

- NLP basics
- Pre-processing in NLP
- Language model with an example
- From words to vectors
- Some applications

Language model: predicting words

- Can you predict the next word?

The stocks fell again today for a third day in this week.

- Clearly, we can narrow down the choice of next word, and sometimes even get it right.
- How?
  - Domain knowledge: third day vs. third minute
  - Syntactic knowledge: a …<adjective | noun>

A language model is perhaps fundamental to how our mind works

- Even illiterate people can predict the next spoken word with some certainty in their native language
- This comes from experience with lots of conversational sentences
- Can a machine gain such “experience?”
- How would such “experience” be modeled?
- What can it be used for?

A probabilistic model of language

- What is the probability of a word? Which words are highly likely?
  - **A, an, the, he, she, it**
  - What about "**obsequious**?"
  - …

$$P(w_m)$$

- What is the probability of a word given its:
  - previous word?
  - Previous two words?
  - Previous three words?
  - …

$$P(w_m | w_{m-1})$$

$$P(w_m | w_{m-1}, w_{m-2})$$

$$P(w_m | w_{m-1}, w_{m-2}, w_{m-3})$$

An example: Guess the word!

- `*** *** ****** **** ** __?`
- `*** *** ****** **** me __?`
- `*** *** ****** pick me __?`
- `*** *** please pick me __?`
- `*** you please pick me __?`
- `Can you please pick me __?`
- **Can you please pick me up?**

N-gram: Markovian assumption

- The information provided by the immediately previous word(s) is the most useful for prediction
- We need not use more than ***n*** previous words

Unigram: $P(w_m | w_{m-1}, w_{m-2}, \ldots, w_{m-\infty}) = P(w_m)$

Bigram: $P(w_m | w_{m-1}, w_{m-2}, \ldots, w_{m-\infty}) = P(w_m | w_{m-1})$

Trigram: $P(w_m | w_{m-1}, w_{m-2}, \ldots, w_{m-\infty}) = P(w_m | w_{m-1}, w_{m-2})$

n-gram: $P(w_m | w_{m-1}, w_{m-2}, \ldots, w_{m-\infty}) = P(w_m | w_{m-1}, w_{m-2}, \ldots, w_{m-n+1})$

- This simplifies our model

How many *n-grams* are there?

- About 20,000 words (unigrams)
- So, about 400,000,000 bigrams, and
- 8,000,000,000 trigrams

- But, are all the bigrams and trigrams equally likely?
  - ***The*** is a common word.
  - ***The the*** does not even make sense.
- Yet, we want ***n*** to be small

Learn N-grams through examples

- Examples from corpora
  - Shakespeare
  - Wall Street Journal
  - Thomson Reuters
- Depending on the corpus, machine will learn that vocabulary; machine can sound like Shakespeare
  - **Where art thou** ****
  - Where **art thou my** ****
  - Where art **thou my forlorn** ****
  - **Where art thou my forlorn prince?**

How does this help us?

- Automatic speech recognition (ASL)
  - “There was a ***bay-er*** behind the bushes”
  - Did she say **bear** or **bare** or **beer** or **bar**?
  - Noun, adjective, verb?
  - Or simply use the previous words
  - This requires many, many examples such that all n-grams that we are ever likely to encounter are seen with reliable frequencies

It also helps spell check software

- Context for the word being checked
- Two types of spelling mistakes:
  - Non words
    - “There was a **baer** behind the bushes”
  - Wrong words
    - “There was a **bare** behind the bushes”
- Both benefit from a language model

greatlearning
Learning for Life

Typical causes of spelling mistakes

- Exchanging two letters, e.g. **baer**
- Typing the wrong key, e.g. **bwar**
- Missing a letter, e.g. **b_ar**
- Adding an extra letter, e.g. **beear**
- Wrong homophone, e.g. **bare** or **beer**
- OCR errors, e.g. **bcar**

Let us model word distortion

- What is the probability of exchanging two letters?
- What is the probability of typing the wrong key?
  - Does it depend on the distance from the right key on keyboard?
- What is the probability of missing a letter?
- …

The distortion model is called <u>channel model</u>

Inspirational source: https://www.youtube.com/playlist?list=PL4LJlvG_SDpxQAwZYtwfXcQr7kGnl9W93

Channel model example: edit distance

- How many additions, deletions?
  - BEAR: (1) <u>F</u>EAR
  - BEAR: (1) <u>F</u>EAR, (2) F<u>-</u>AR
  - BEAR: (1) <u>F</u>EAR, (2) F<u>-</u>AR, (3) FAR<u>E</u>
- Should additions and deletions have equal weight?
- What about exchange of two letters?
- What about pressing wrong neighboring key?

Channel model: *P(typed word | candidate word)*

Inspirational source: https://www.youtube.com/playlist?list=PL4LJlvG_SDpxQAwZYtwfXcQr7kGnl9W93

Putting the two models together

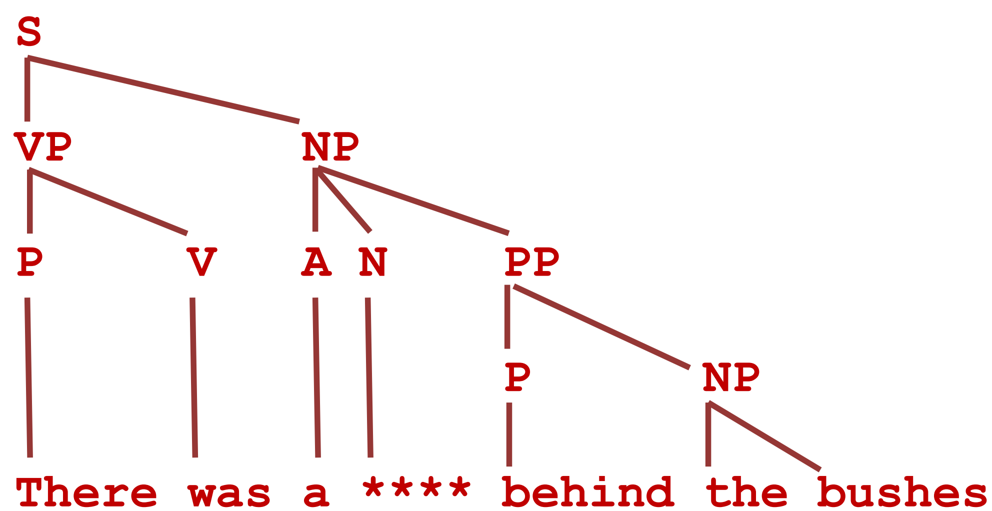
- Bayes theorem and chain rule to the rescue:
  - $P(A,B) = P(A|B) \times P(B) = P(B|A) \times P(A)$
- Let **W** be typed word, **F** be phrase before, **W′** be candidate word
- Find **W′** that maximizes: $P(W'|W,F)$; own probability given data

$$
\begin{aligned}
P(W'|W,F) &= P(W',W,F) \;/\; P(W,F) \\
&\propto P(W',W,F) \\
&= P(W,W',F) \\
&= P(W|W',F) \times P(W',F) \\
&= P(W|W',F) \times P(W'|F) \times P(F) \\
&\propto P(W|W',F) \times P(W'|F) \\
&\approx P(W|W') \times P(W'|F) \\
&= \textit{Channel model} \times \textit{Language model}
\end{aligned}
$$

- That is, it is most likely to have led to the distortion *AND* makes sense language-wise

Role of linguistics in NLP, an example

- What if an ***n-gram*** wasn't in the corpus?
- Knowledge of parts of speech (POS) can help
- Another NLP problem: POS tagging
- Linguistics uncovers language syntax, grammar, and POS patterns
- Now word choices can be limited by POS for ASL or spell check
  - No ***bare***!



Outline

- NLP basics
- Pre-processing in NLP
- Language model with an example
- From words to vectors
- Some applications

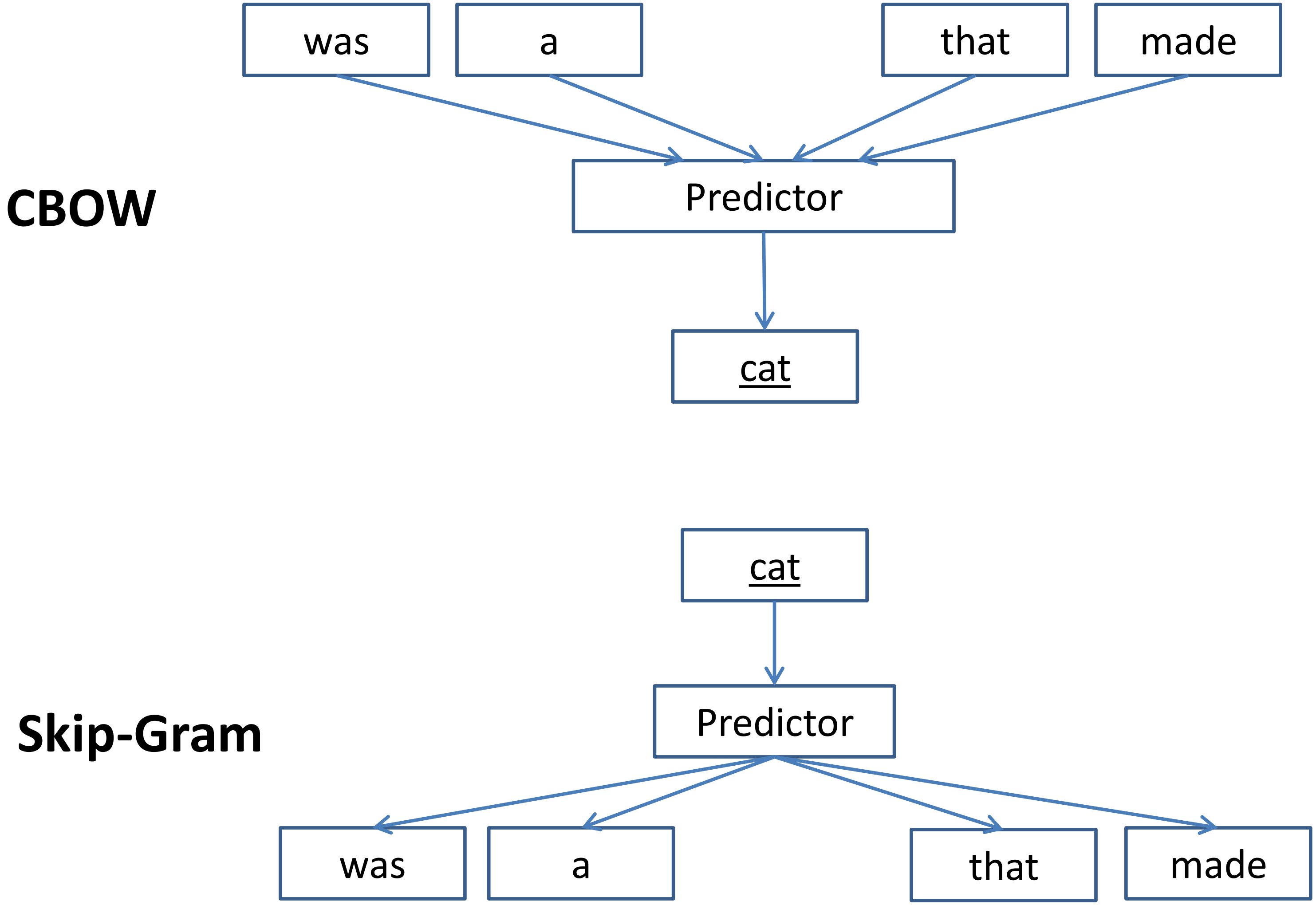
Encoding

- Moving from sparse (e.g. one-hot) to dense vectors
- $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0.221 \\ 0.578 \\ 0.091 \end{bmatrix}$
- Each dimension could represent attributes such as geography, gender, POS etc.
- A very successful model to do so is Word2Vec

CBOW and Skip-Gram

- Example: *It was a cat that made all the noise*
- In continuous bag-of-words (CBOW), we try to predict a word given its surrounding context (e.g. location ± 2)
  - (*was* → *cat*), (*a* → *cat*), (*that* → *cat*), (*made* → *cat*)
- In a skip-gram model, we try to model the contextual words (e.g. location ± 2) given a particular word
  - (*cat* → *was*), (*cat* → *a*), (*cat* → *that*), (*cat* → *made*)

Visualizing CBOW and Skip-Gram



Adapted from: https://arxiv.org/pdf/1301.3781.pdf

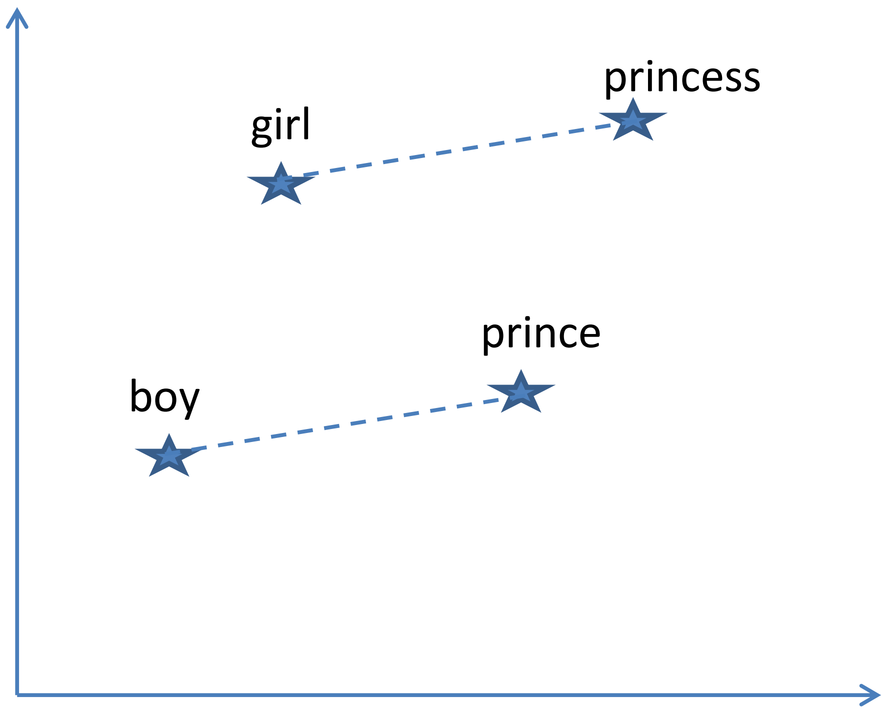
How it is trained

- The objective is to maximize the probability of actual skip-grams, while minimizing the probability of non-existent skip-grams
- $\underset{\theta}{\arg\max} \prod_{w,c \in D} p(D=1|w,c;\theta)$
- $\prod_{w',c' \in D'} p(D=0|w',c';\theta)$
- $\underset{\theta}{\arg\max} \sum_{w,c \in D} \log \frac{1}{1+e^{-v_w . v_c}} + \sum_{w',c' \in D'} \log \frac{1}{1+e^{v_{w'} . v_{c'}}}$

Adapted from: https://arxiv.org/pdf/1402.3722.pdf

The new vectors can directly be used to find analogs

- E.g. $v_{prince} - v_{boy} + v_{girl} = v_{princess}$



Word2Vec example results

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

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

Source: https://arxiv.org/pdf/1301.3781.pdf

Word2vec design choices

- Dimension of the vector
  - Large dimension is more expressive
  - Small dimension trains faster
  - No incremental gain after a particular dimension
- Number of negative samples
  - Increases the search space
  - Gives better models
- Neural network architecture
  - Hidden units to convert 1-hot-bit into a vector

GloVe : Global Vectors

- GloVe captures word-word co-occurrences in the entire corpus better
  - Let X_{ij} be the co-occurrence probability of words indexed with i and j
  - Let X_i be $\sum_j X_{ij}$
  - And, let $P_{ij} = P(j|i) = X_{ij} / X_i$
  - What GloVe models is $F((w_i - w_j)^{\mathrm{T}} w_k) = P_{ik} / P_{jk}$
  - Let X_{ij} be the co-occurrence probability of words indexed with i and j

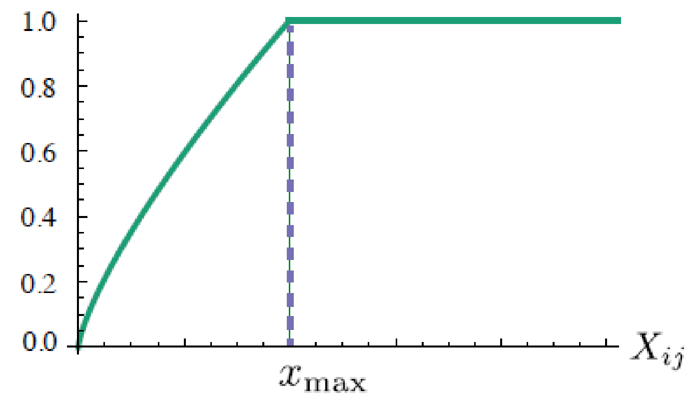
"GloVe: Global Vectors forWord Representation" by Jeffrey Pennington, Richard Socher, Christopher D. Manning

GloVe explanation

- Cost function:

$$J = \sum_{i,j} f(X_{ij})(w_i{}^T \widetilde{w}_j - \log X_{ij})^2$$

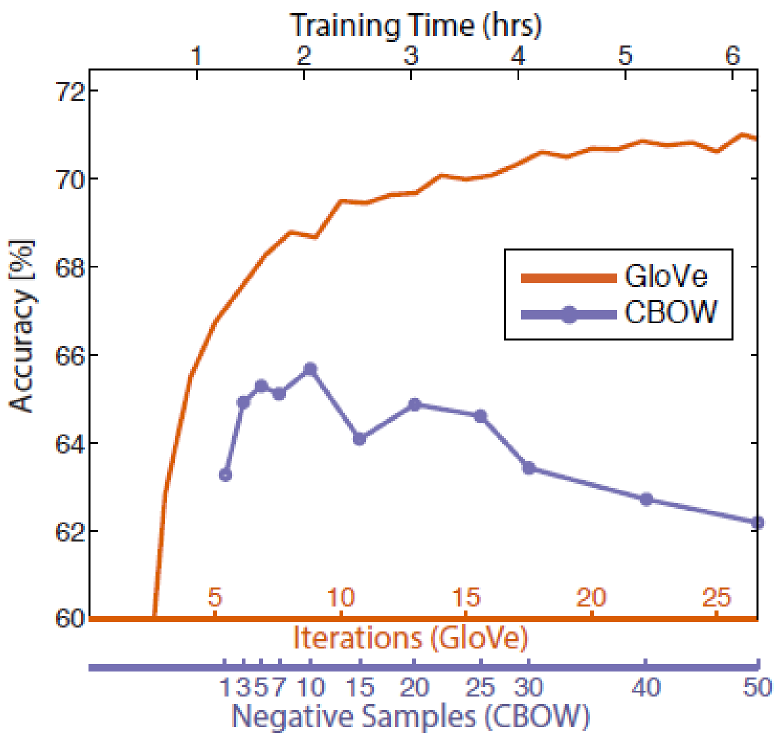
  - For words i, j co-occurrence probability is X_{ij}
  - And, a weighing function f



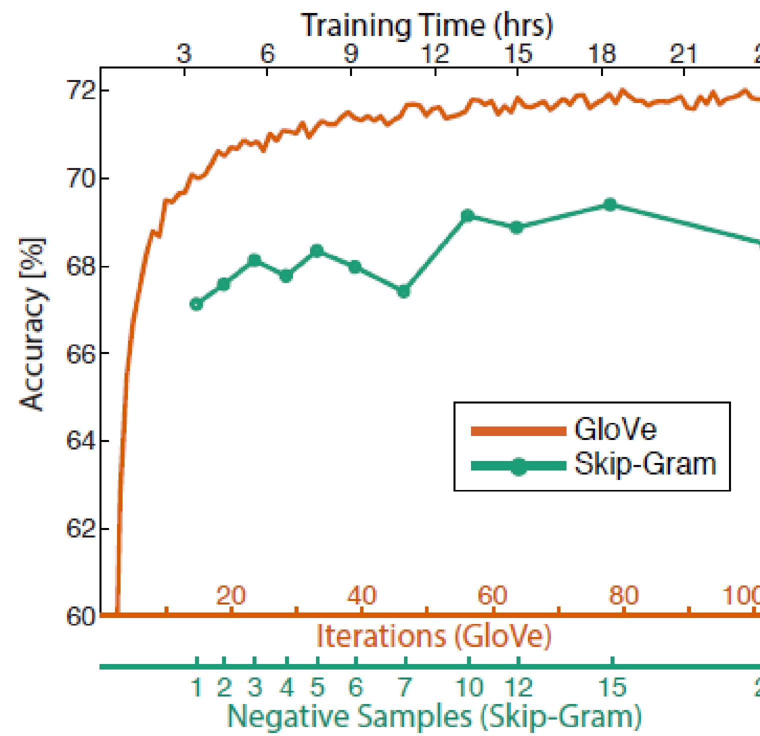
    - Suppresses rare co-occurrences
    - And prevents frequent co-occurrences from taking over

"GloVe: Global Vectors forWord Representation" by Jeffrey Pennington, Richard Socher, Christopher D. Manning

GloVe is more accurate than word2vec



(a) GloVe vs CBOW



(b) GloVe vs Skip-Gram

The accuracy show above is on word analogy task

"GloVe: Global Vectors forWord Representation" by Jeffrey Pennington, Richard Socher, Christopher D. Manning

Outline

- NLP basics
- Pre-processing in NLP
- Language model with an example
- From words to vectors
- Some applications

Application: POS tagging

- Goal: Find *part-of-speech* of each word
- Application: Use in language model to structure sentences better
- Example:

Amit	found	the	tray	and	started	to	bring	it	to	the	guest
NNP	VBD	DT	NN	CC	VBD	TO	VB	PRP	IN	DT	NN

- Certain regular expressions can be helpful
  - For example, words ending with **ing* are usually verbs
- Corpora with tagged words can be used
  - For example, Brown corpus

Examples of tags

- Nouns
  - Singular noun → NN (Cat)
  - Plural noun → NNS (Cats)
  - Proper noun → NNP (Garfield)
  - Personal pronouns → PRP (He)
- Verb
  - Base verb → VB (sleep)
  - Gerund → VBG (sleeping)
- Preposition → IN (over)
- Adjective
  - Basic → JJ (bad)
  - Comparative → JJR (worse)
- Adverb
  - Basic → RB (quickly)
- Determiner
  - Basic → DT (a, an, the)
  - WH → WDT (which, who)
- Coordinating conjunction → CC (and, or, however)

Some POS Tagging Challenges

- Ambiguity that needs *context*
  - It is a quick **read** (NN)
  - I like to **read** (VB)

- Differences in numbers of tags
  - Brown has 87 tags
  - British National Corpus has 61 tags
  - Penn Treebank has 45 tags (several merged)

Approaches to POS Tagging

- Learn from corpora
- Use regular expressions
  - Words ending with *‘ed’* or *‘ing’* are likely to be of a certain kind
- Use context
  - POS of preceding words and grammar structure
  - For example, n-gram approaches
- Map untagged words using an embedding
- Use recurrent neural networks

Application: Named entity recognition

- Something which has a name:
  - Person, place, thing, time
- Example:
  - Thereafter, Amit went to the supermarket

    Name place
- Application:
  - Tag texts for relevance and search

Some challenges with NER

- Different entities sharing the same name
  - Manish *Jindal* → Person
  - *Jindal* Steel → Thing (company)
- Common words that are also names
  - Do you want it with *curry* or dry
  - Tyler *Curry*
- Ambiguity in the order, abbreviation, style
  - Jindal, Manish
  - Dept. of Electrical Engineering
  - De Marzo, DeMarzo

Approaches to NER

- Match to an NE in a tagged corpus
  - Fast, but cannot deal with ambiguities
- Rule based
  - E.g. capitalization of first letter
  - Does not always work, especially between different types of proper nouns
- Recurrent neural network based
  - Learn from a NE tagged corpus

Other applications

- Sentiment analysis
  - Is a given product review positive or negative?
  - Which are the most significant reviews?

- Text generation
  - Question answering, e.g. chatbots
  - Language translation, e.g. English to Telugu