ECE 1786 Lecture #1

- Welcome in person! Please make sure that you i) view lecture Ø online
  ii) fill out the pre-requisite survey.

on board:

- recap of lecture Ø → classification of language → generation of language | SGD? Debug?
  - new course is about NLP & deep learning
  - incredible progress & capabilities in last 5 years
  - course is lectures, 4 assignments + project
  - believe learning comes from doing! ; text: Jurafsky. | See syllabus
  - must have pre-requisites; will/have contacted you if you don't appear to; can't stay if don't.
  - Questions? - Survey due today

mixture of bottom-up detail + top-down intuition

→ 2 hours time for this course!

- Assignment #1 released; due Sept 26 @ 9pm. → see Quercus website / Assignments
  - covering material today + lecture 2.
  - will see that ML background + coding experience is necessary
  - uses PyTorch; need to come up to speed on Google Colab or install PyTorch on own computer; not using TF

- few slides → chalk + talk

- let's begin: Natural Language Processing

- why 'Natural'? (vs. computer language, it is our human spoken & written language)

- NLP is/was difficult because of the ① ambiguity → define in language ② that there are many ways to say same thing (① different interpretations) ②

- e.g. ③ words have multiple meanings - bank
  - duck

- understanding of possibly ambiguous sentences requires extra knowledge: wrong interp? right interp? how do we know?

  - "boy paralyzed after tumour fights back to gain black belt"

  - "I saw bats" ... in the cave
    ... to help kids in baseball.

- to account for these & many more with procedural programming - i.e if "context = baseball" then bats are wood
  else if "context = cave" then bats are alive.
- is way too difficult. (although the field has worked on this for many years)

Instead - modern, successful NLP is based on the encoding of words/meaning into numbers

- a word (or a concept) can become/be encoded as say, 100 numbers in an "embedding" or "vector"

  such that words that are closer in meaning ~~are~~ have closer vectors. → don't need to deal with specific words

e.g vector size 100

e.g. Word 1 - apple ⇒ $V_{apple} = \{a_0, a_1, \ldots, a_{99}\}$
2 - banana ⇒ $V_{banana} = \{b_0, b_1, \ldots, b_{99}\}$
⋮
10K word vocabulary 10,000 - zebra ⇒ $V_{zebra} = \{z_0, z_1, \ldots, z_{99}\}$

- since apple & banana are similar, → fruits, eaten we expect their vectors to be "close"

- one way to measure closeness is with Euclidean Distance; e.g distance between V_{apple} & V_{banana} is

$$= \sqrt{(a_0-b_0)^2 + (a_1-b_1)^2 + \ldots + (a_{99}-b_{99})^2}$$

- in PyTorch computed with torch.norm

- these vectors form the core of what has been called, for many years the "stastical approach" to natural language processing;
- there have been a number of methods for computing them, as described in Jurafsky Ch 6, Sections 6.5 → 6.7
  - based on counting appearance of words in documents - TF-IDF - PMI.
  - we won't cover these; instead we'll use the modern, neural net method.

- before going into the neural method, let's bring home the power of the ~~vec~~ neural embedding / vector method. (includes cosine)

$\boxed{\text{→ online demo of A1\_Section1\_starter.ipynb}}$

↳ part of assignment 1

ALSO SHOW TENSORFLOW PROJECTOR

break - 5 or 10 mins?

→ second method of measuring distance (well, similarity) between two vectors: cosine similarity
  - gives a number between -1 (not similar) and 1 (very similar) - normalized.

$$\text{cosine similarity} = \frac{V_A \cdot V_B}{\|V_A\|\|V_B\|} = \frac{\sum_{i=0}^{n-1} a_i b_i}{\sqrt{\sum_{i=0}^{n-1} a_i^2}\sqrt{\sum_{i=0}^{n-1} b_i^2}}$$

- considers the direction of the vector, not magnitude

break

- besides the suprising relationship $V_{QUEEN} - V_{KING} = V_{WOMAN} - V_{MAN}$ → this suggests there is more going on in the representation of vectors.
- there are quite a number of other pair-wise relationships:

- e.g. $V_{big} - V_{\cancel{smallest}\ biggest} = V_{small} - V_{smallest}$
  & can use to answer question
  "big is to biggest as small is to ..... ?"

  by computing $V_{ANSWER} = V_{biggest} - V_{big} + V_{small}$
  & search for "nearest" word.

- Mikolov paper + others talk about many relationships; you're asked to do this in Assignment 1.
posted.

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

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

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

Meaning & Vectors

- in lecture 2 & Assignment 1 Sections 3 & 4 you will explore how these remarable vector/embeddings are created - it is a very clever method that leverages the training loop/optimization of a NN

- in one sentence: the vectors are a ~~by-product~~ side-product of a neural network that is trained to predict ~~the~~ words that are related to a given word

- however, as you saw with glove["apple"] the numbers in a dim = 50 vector have no apparent meaning to us humans → L2 will make the dra
  - I find this annoying & would like it to be possible to have explainable elements of vectors [might lose the pairwise relationships]

- how might this work, if we were creating vectors by hand?

ask for suggestions of categories

- suppose that we wanted to create vectors for the words; we might create categories of meaning like

nouns	Colour	Temperature	Plants	Human	
Mountain	0.2	0.3	0.4	0.0	- ask
Ocean	0.3	0.3	0.5	0.0	
Sun	0.4	0.8	0.0	0.0	
(noun) Judge	0.3	0.1	0.0	0.9	
Radiator	0.1	0.6	0.0	0.0	
Grass	0.6	0.2	0.8	0.0	

- we could try to fill in the numbers, between -1 & 1

- I wonder, how many "dimensions" do we need to be able to cover all meanings! 50 → 300 → 1000 looks like enough!

- but we don't get these "understandable" vectors from the NN training process

- is there a way to compute understandable meaning from the NN-trained vectors? yes
  not understandable vectors.

How? Method 1:

e.g. ~~te~~colour

- for each "category of meaning" come up with several words that represent that category
  e.g for category colour: colour, red, green, blue, brown
  ⇒ why might just colour not be sufficient?

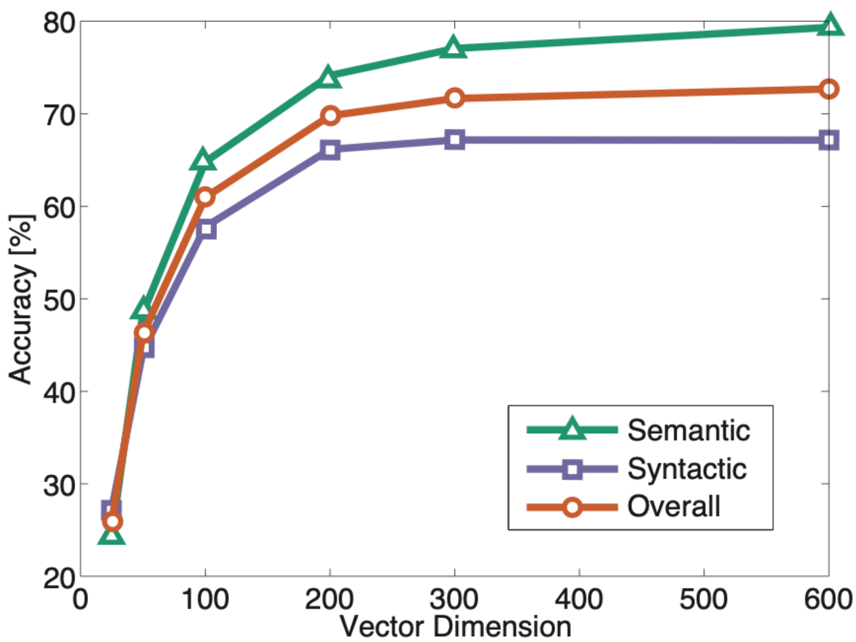
- Compute cosine similarity between a word of interest (e.g. grass, mountain) and each of these words. Average the results, i.e.

  Σ cosine_dist ("grass", "colour")
  " ("grass", "red")
  " ("grass", "green")
  " ⋮
  ("grass", "brown") / 6.

- Method 2
  - first, average vectors of all words in category
    - what might this do? i.e. average = $\frac{\Sigma \text{ wordsvecs}}{6}$
  - compute cosine_dist ("grass", average_vec
- this is the task more in Assignment 1, Section 2

On that point of 'wondering how many dimensions would be enough' to cover the meaning of all words?

When training these word vectors (the GloVe vectors), the Pennington paper uses them for various tasks and comes to the conclusion that maybe a dimension of size 300 is enough, based on this graph:



I thought this was true for some time, until transformer models came along (the model used in the GPT-x and chatGPT). For the very large language models, they employ word vector sizes of over 12,000, much larger. Some significant part of the knowledge of the big models is in those vectors.