Introduction to Social Media Analytics (Lec 6)

Hao PENG
Department of Data Science
City University of Hong Kong
https://haoopeng.github.io/

Beyond sentiment analysis

- Emotion classification
- Spam/clickbait detection
- Helpfulness rating on Q&A sites
- Complain/fake/rating of product reviews
- Civility/toxicity/hateful speech
- Language identification of posts
- Fake news / bot detection
- Identify AIGC posts on RedNote

Need advanced methods to deal with complex linguistic tasks.

Feature engineering

- Word-level:
  - Bag of words, N-grams
  - TF-IDF
- Word dictionary:
  - LIWC
  - LDA
- Embeddings:
  - Word2vec
  - Neural nets (CNN/RNN/LSTM)
  - Transformer (LLMs)

LIWC Dictionary (bag of topics)

- Linguistic Inquiry and Word Count (LIWC)
- Closed-vocabulary: fixed word dictionary look up

LIWC Dimensions and Sample Words	
DIMENSION	**EXAMPLES**
I. STANDARD LINGUISTIC DIMENSIONS	
Pronouns	I, them, itself
Articles	a, an, the
Past tense	walked, were, had
Present tense	Is, does, hear
Future tense	will, gonna
Prepositions	with, above
Negations	no, never, not
Numbers	one, thirty, million
Swear words	*****

II. PSYCHOLOGICAL PROCESSES	
Social Processes	talk, us, friend
Friends	pal, buddy, coworker
Family	mom, brother, cousin
Humans	boy, woman, group
Affective Processes	happy, ugly, bitter
Positive Emotions	happy, pretty, good
Negative Emotions	hate, worthless, enemy
Anxiety	nervous, afraid, tense
Anger	hate, kill, pissed
Sadness	grief, cry, sad
Cognitive Processes	cause, know, ought
Insight	think, know, consider
Causation	because, effect, hence
Discrepancy	should, would, could

- Feature engineering: aggregate word count into category count

https://lit.eecs.umich.edu/geoliwc/liwc_dictionary.html

LIWC in python demo

```
with open('./LIWC.pkl', 'rb') as liwc:
    int_cate, liwc = pickle.load(liwc)
```

```
print(liwc)

{'dormancy': [110, 112],
'bronchiarctia': [70, 72],
'naggy': [30, 32, 34],
'disorientates': [50, 54],
'daywrit': [100, 103],
'gentlewomanish': [40, 43],
'fifty': [24],
'undeterminable': [50, 54],
'accomplishment': [80, 82, 110], ...}
```

```
int_cate

{1: 'function (Function Words)',
 2: 'pronoun (Pronouns)',
 3: 'ppron (Personal Pronouns)',
 4: 'i (I)',
 5: 'we (We)',
 6: 'you (You)',
 7: 'shehe (SheHe)',
 8: 'they (They)',
 9: 'ipron (Impersonal Pronouns)',
 10: 'article (Articles)',
 11: 'prep (Prepositions)',
 12: 'auxverb (Auxiliary Verbs)',
 13: 'adverb (Adverbs)',
 14: 'conj (Conjunctions)',
 15: 'negate (Negations)',
 20: 'verb (Verbs)',
 21: 'adj (Adjectives)',
```

LIWC in predicting gender & age

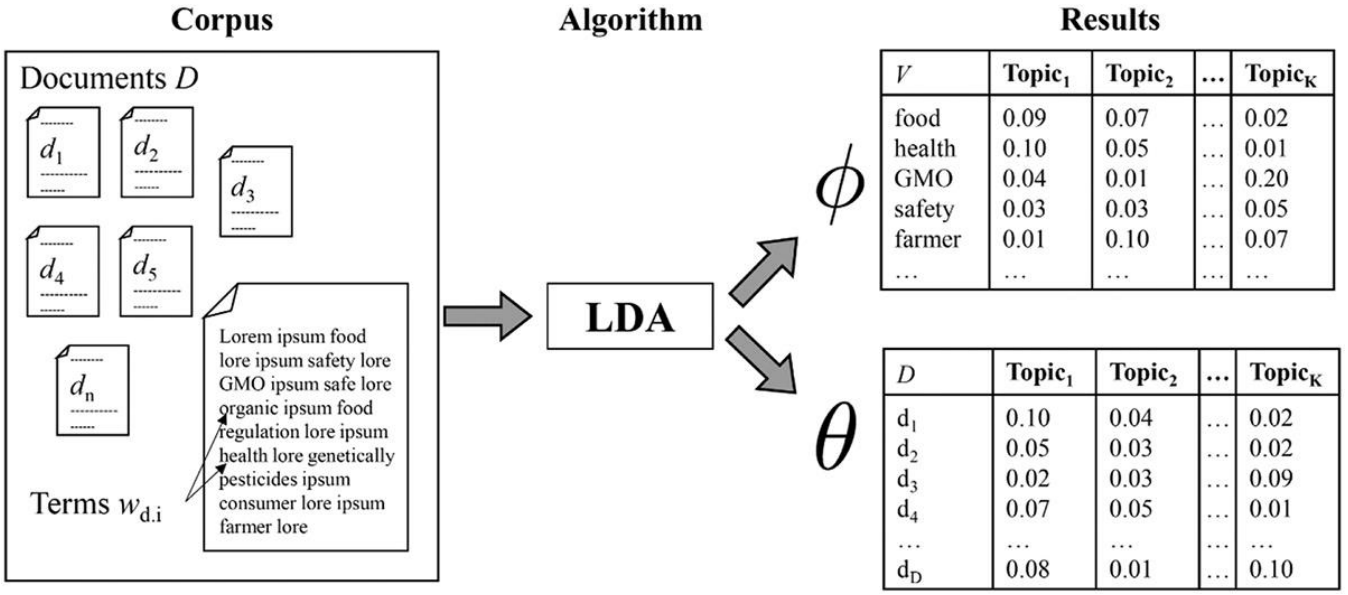
LIWC Category	Gender		Age	
	[34] *d*	our *β*	[30] *β*	our *β*
Total function words	-	-0.04	-	0.16
Total pronouns	0.36	0.07	-	-0.02
Personal pronouns	-	0.14	-	-0.08
1st pers singular	0.17	0.13	-0.14	-0.22
1st pers plural	*ns*	*ns*	-0.13	0.21
2nd person	-0.06	0.05	-	0.04
3rd pers singular	-	0.09	-	0.15
3rd pers plural	-	-0.05	-	0.26
3rd pers overall	0.2	-	-	-
Impersonal pronouns	-	-0.09	-	0.11
Articles	-0.24	-0.24	-	0.28
Common verbs	-	0.04	-	0.02
Auxiliary verbs	-	0.02	-	0.08
Past tense	0.12	-0.03	-0.16	*ns*
Present tense	0.18	0.08	0.04	*ns*
Future tense	*ns*	-0.07	0.14	0.09
Adverbs	-	0.05	-	-0.07
Prepositions	-0.17	-0.13	-	0.27
Conjunctions	-	0.03	-	0.12
Negations	0.11	*ns*	-	-0.12
Quantifiers	-	-0.09	-	0.24
Numbers	-0.15	-0.13	-	0.05
Swear words	-0.22	-0.21	-	-0.17
Social processes	-	0.08	-0.13	0.21
Family	0.12	0.22	-	0.28
Friends	0.09	0.08	-	0.26
Humans	*ns*	0.04	-	0.06

Advantages of LIWC?
Any disadvantages?

https://doi.org/10.1371/journal.pone.0073791

LDA topic modeling

- Latent Dirichlet Allocation (LDA) is a generative statistical model for automatically discovering hidden topics across a set of documents (Unsupervised learning; Open-vocabulary approach)
- Represent documents as prob. distribution of topics
- Represent the topics are prob. distributions of words



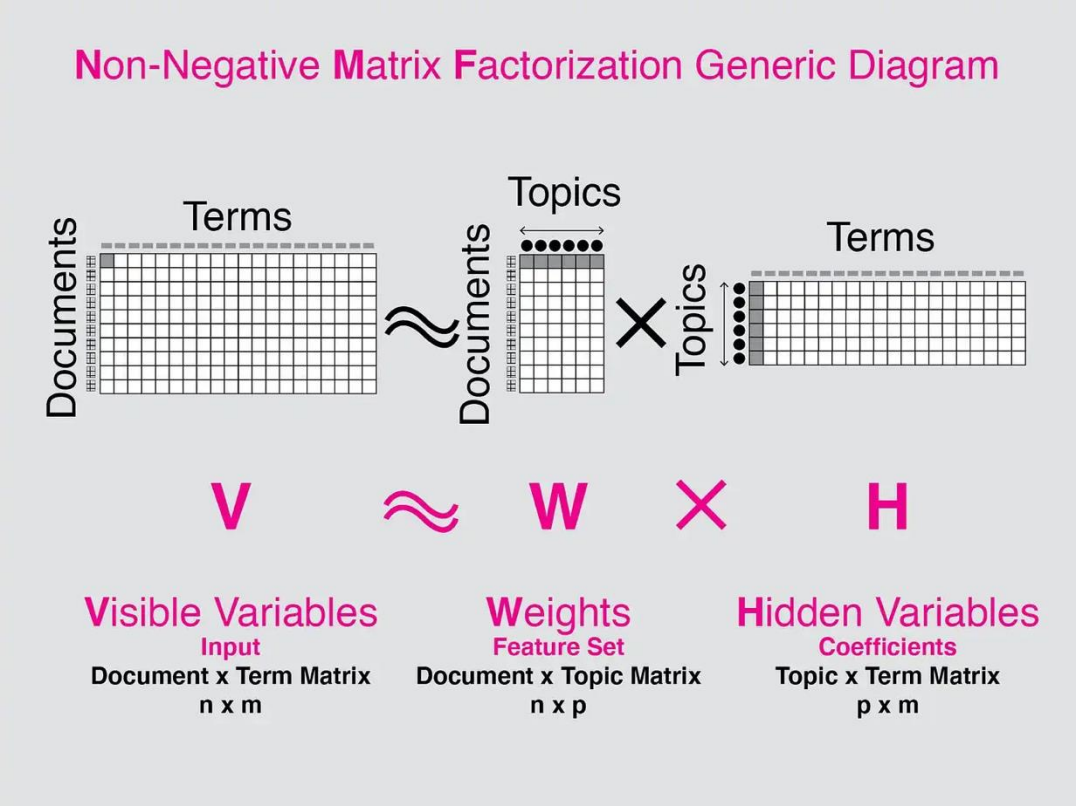
V	Topic$_1$	Topic$_2$	…	Topic$_K$
food	0.09	0.07	…	0.02
health	0.10	0.05	…	0.01
GMO	0.04	0.01	…	0.20
safety	0.03	0.03	…	0.05
farmer	0.01	0.10	…	0.07
…	…	…	…	…

D	Topic$_1$	Topic$_2$	…	Topic$_K$
d_1	0.10	0.04	…	0.02
d_2	0.05	0.03	…	0.02
d_3	0.02	0.03	…	0.09
d_4	0.07	0.05	…	0.01
…	…	…	…	…
d_D	0.08	0.01	…	0.10

https://doi.org/10.1080/19312458.2018.1430754

LDA topic modeling

- LDA is similar with PCA. They both are matrix factorization techniques.
- LDA is applied to text data. It works by decomposing the Document Word Matrix (the larger matrix) into two parts (smaller matrices): the Document Topic Matrix and the Topic Word Matrix.



Applications of LDA

- LDA can transform data into low dimensional feature space
- Perform predictive tasks using the document-topic matrix:
  - categorize news (sports, politics, tech, etc.)
  - book / music / movie genre classification
  - recommendation system; document search
  - predict user age, gender, ethnicity, sentiment, etc.

Identify topics in song lyrics

```
import spacy
from gensim import models, corpora

# Fetch data
songs_df = pd.read_csv("https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset.csv")
data = list(songs_df["text"]) # clean the data
tokenized_data = list(map(custom_tokenizer, nlp.pipe(clean_data, n_process=4)))

# Build dictionary of (token, ID) mappings, Filter tokens
dct = corpora.Dictionary(tokenized_data)
dct.filter_extremes(no_below=5, no_above=0.5)

# Build Bag-of-Words sparse matrix
bow = [dct.doc2bow(i) for i in tokenized_data]

# Train LDA model
lda_model = models.LdaModel(corpus=bow, num_topics=10, passes=10, alpha='auto', eta='auto', id2word=dct)
```

https://www.kaggle.com/code/samuelcortinhas/nlp6-topic-modelling-with-lda

Song recommendations with LDA

```
# Get word distribution for each topic
topic_words = lda_model.show_topic(topic_id)

# Return topic distribution for a given bow as a list of (topic_id, prob) tuples.
# Note: Probabilities sum to ~1.
topics = lda_model.get_document_topics(bow[song_idx])

# Get topic distributions of new documents.
new_songs = ["Demons on my shoulder Monsters in my head", "Shadow in the water Will you be my friend"]
new_tokens = list(map(custom_tokenizer, [nlp(new_songs)]))
new_bow = dct.doc2bow(new_tokens)
new_topic_vectors = lda_model[new_bow]

# Get topic distributions for the training corpus.
topic_vectors = lda_model[bow]

# Compute consine similarities
def to_dense_vector(sparse_dist, num_topics):
    dense = np.zeros(num_topics)
    dense[[i for i, p in sparse_dist]] = [p for i, p in sparse_dist]
    return dense
```

https://www.kaggle.com/code/samuelcortinhas/nlp6-topic-modelling-with-lda

Visualizing topics

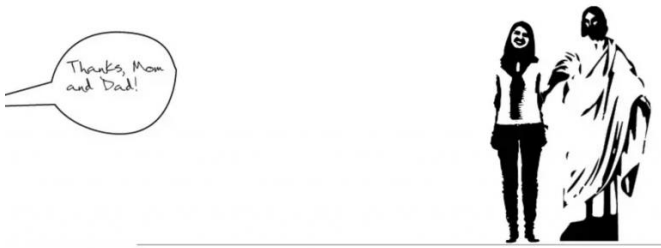
How do we name the topics? How to evaluate topics?

Limitations of LDA

- Doesn’t work well on short document (such as Tweets).
- LDA relies on the assumption that words in each topic are related and meaningful, but this may not always be the case.
- It can produce ambiguous or incoherent topics.
- Not straightforward to evaluate topics’ quality.

Recall weaknesses of BoW

I want to thank my parents, Tiffany and God



I want to thank my parents, Tiffany, and God.

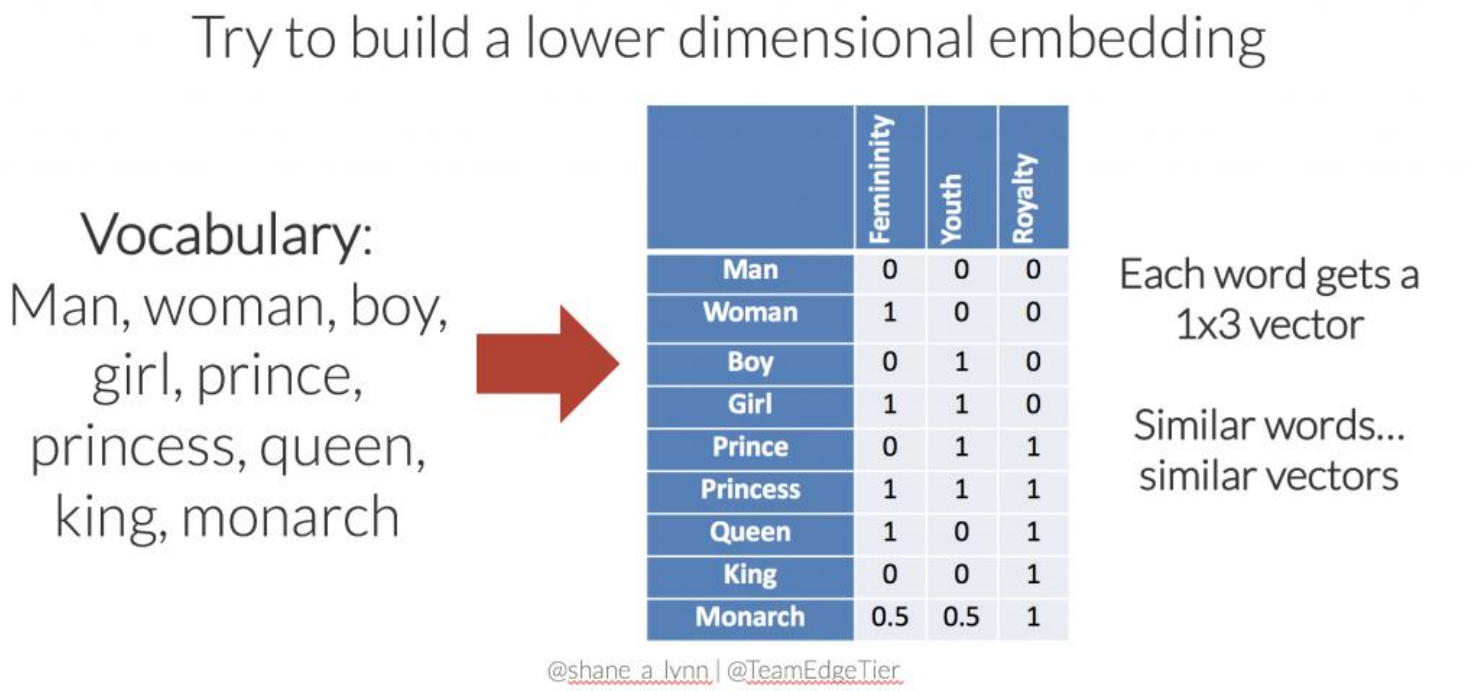
- High dimension, Sparsity
- Ignores order and context
- Lack of semantic nuance

The baseball **pitcher** drank a **pitcher** of water

https://aiml.com/what-are-the-advantages-and-disadvantages-of-bag-of-words-model/

Vector-space model

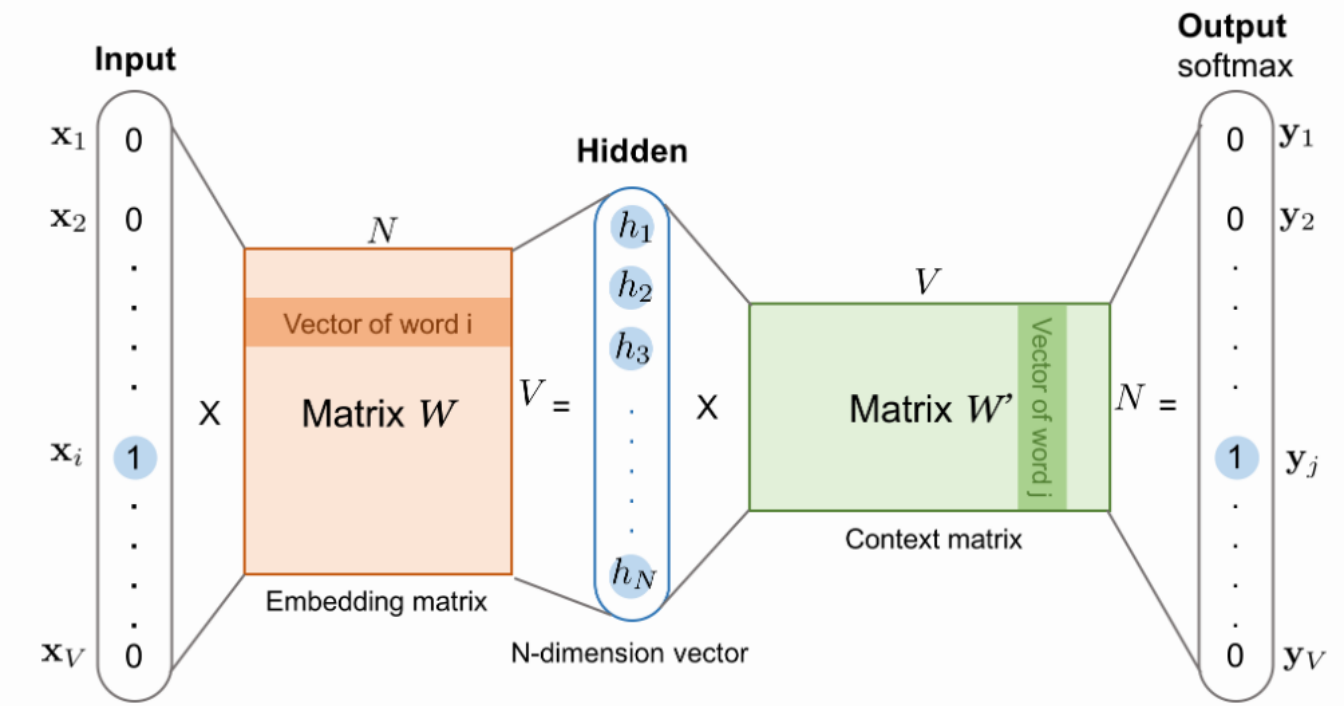
- Captures deeper semantic and linguistic nuances



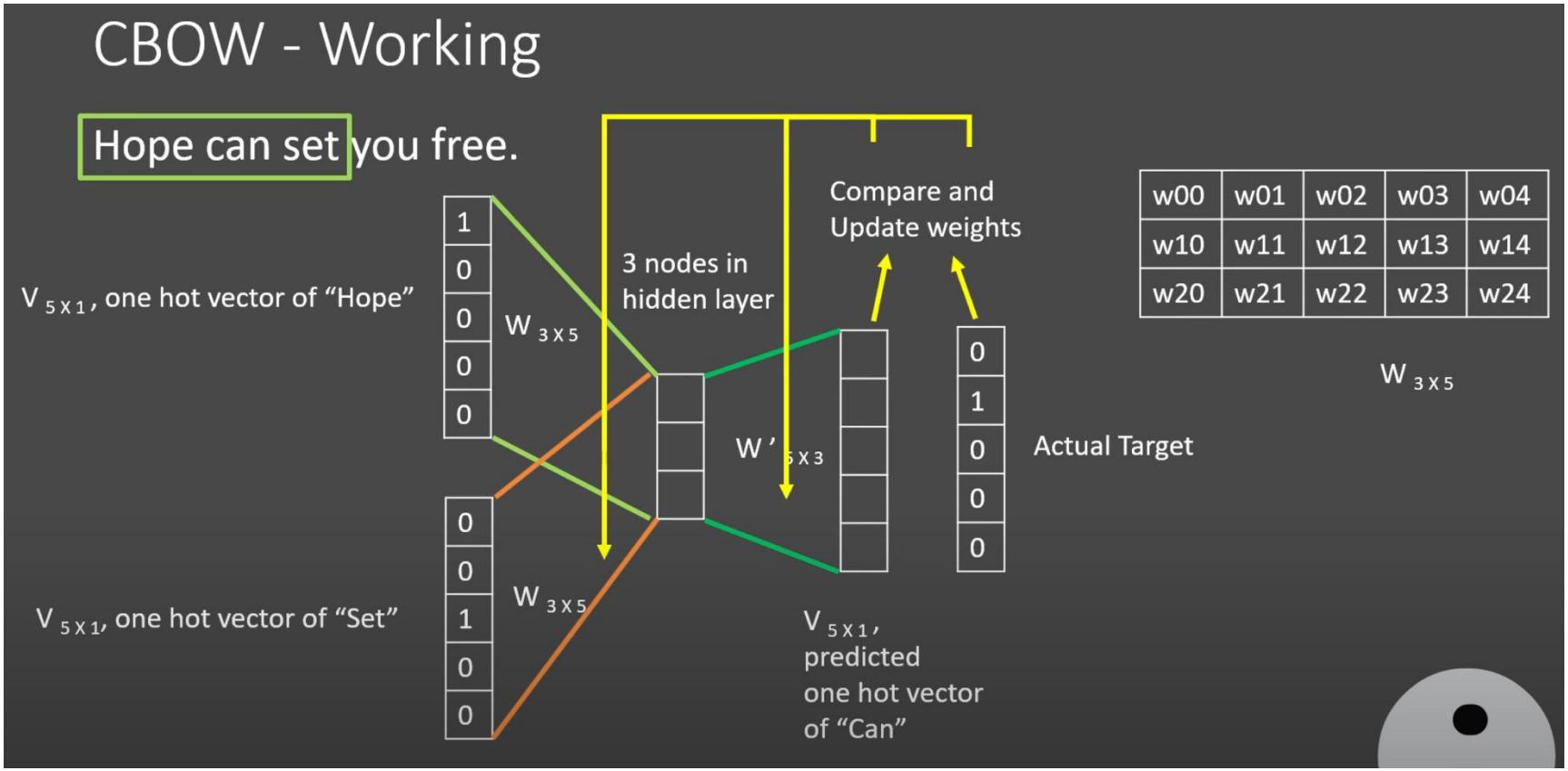
https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Word2vec training setup

- *V* is the vocabulary size; *N* is the num of dimensions
- *W* is (*N* x *V*), *W'* is (*V* x *N*)
- Input is one-hot encoding of each word (*V* x 1)

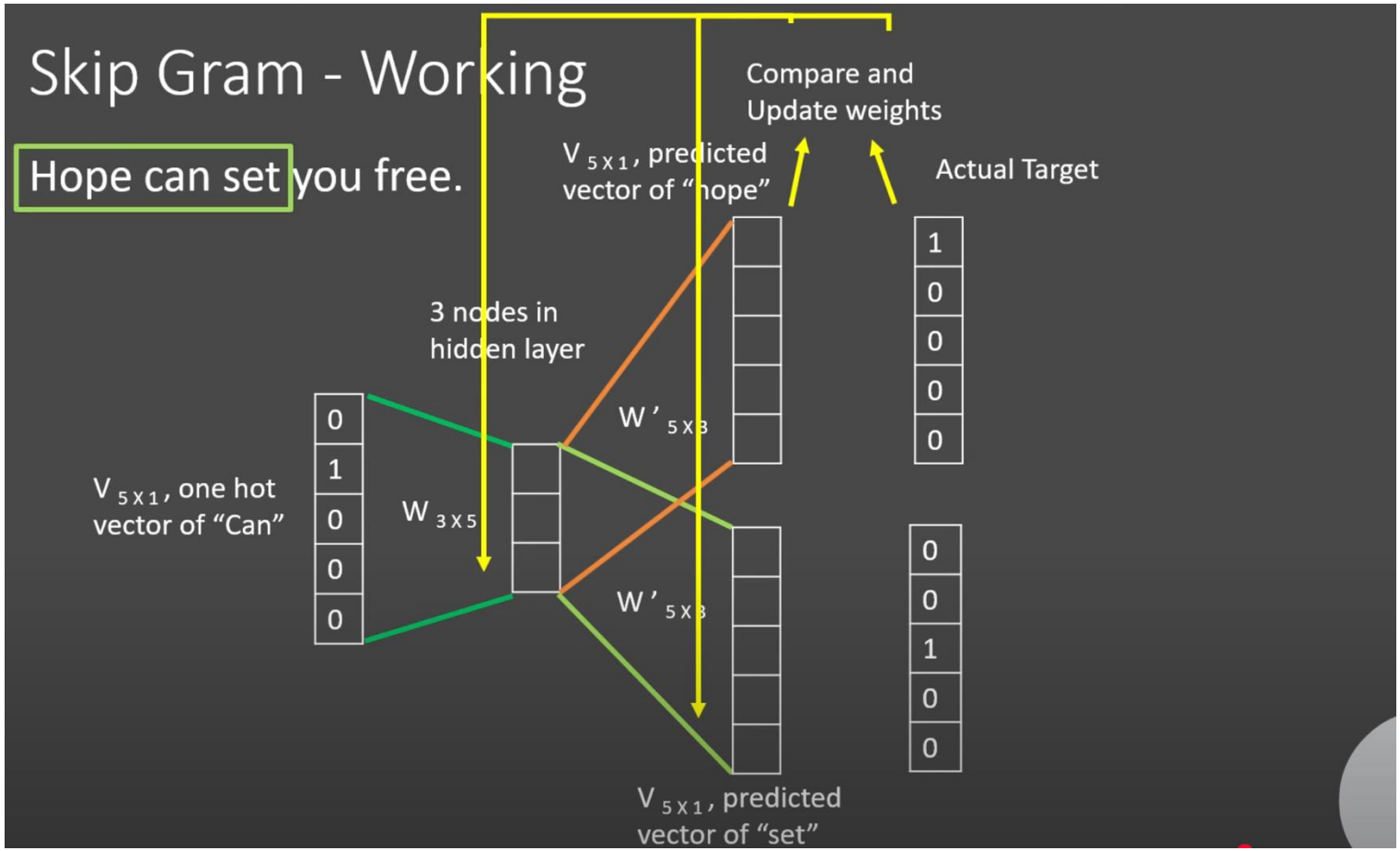


CBOW training architecture



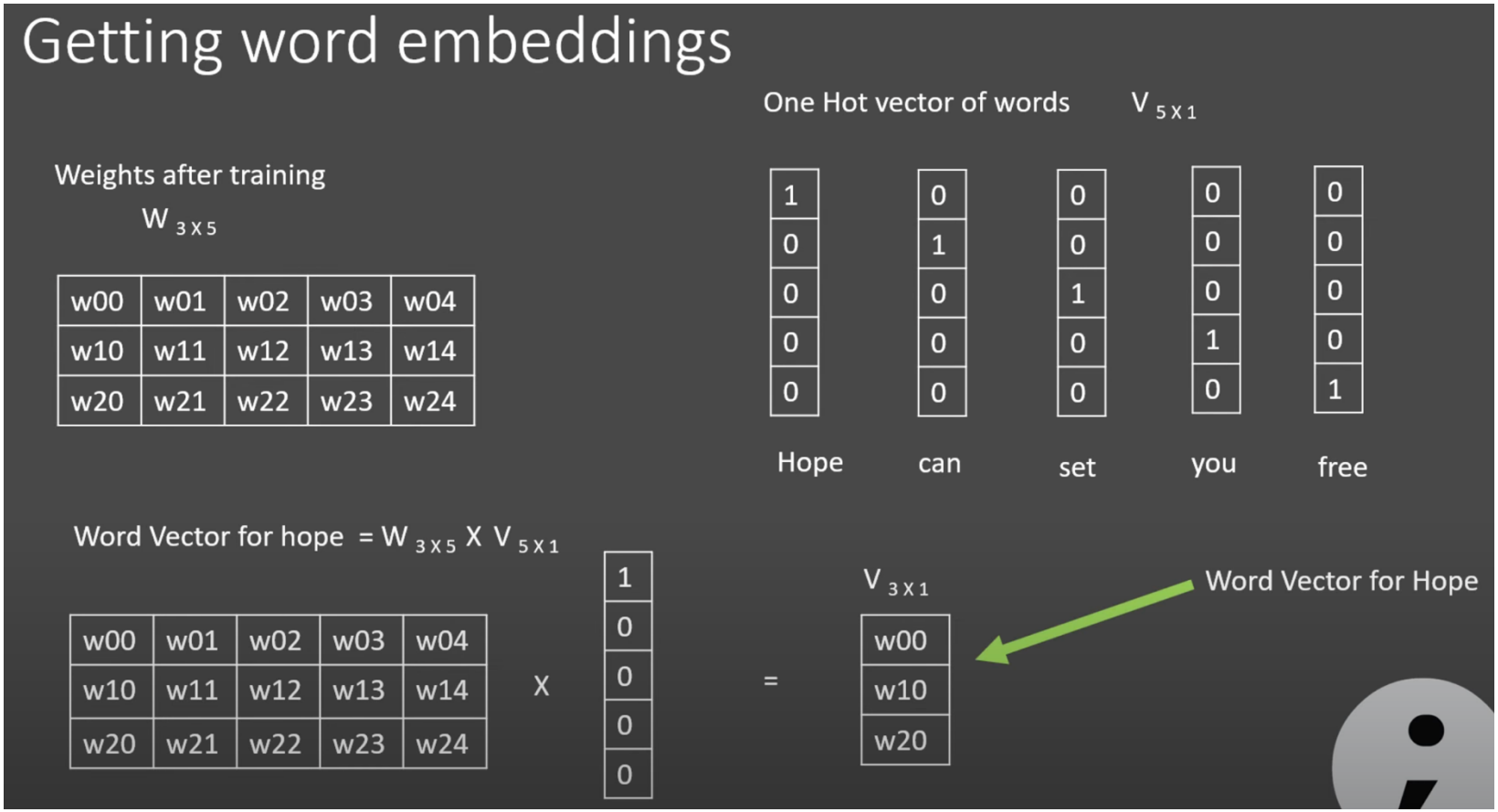
https://www.youtube.com/watch?v=UqRCEmrv1gQ

Skip-gram training architecture



https://www.youtube.com/watch?v=UqRCEmrv1gQ

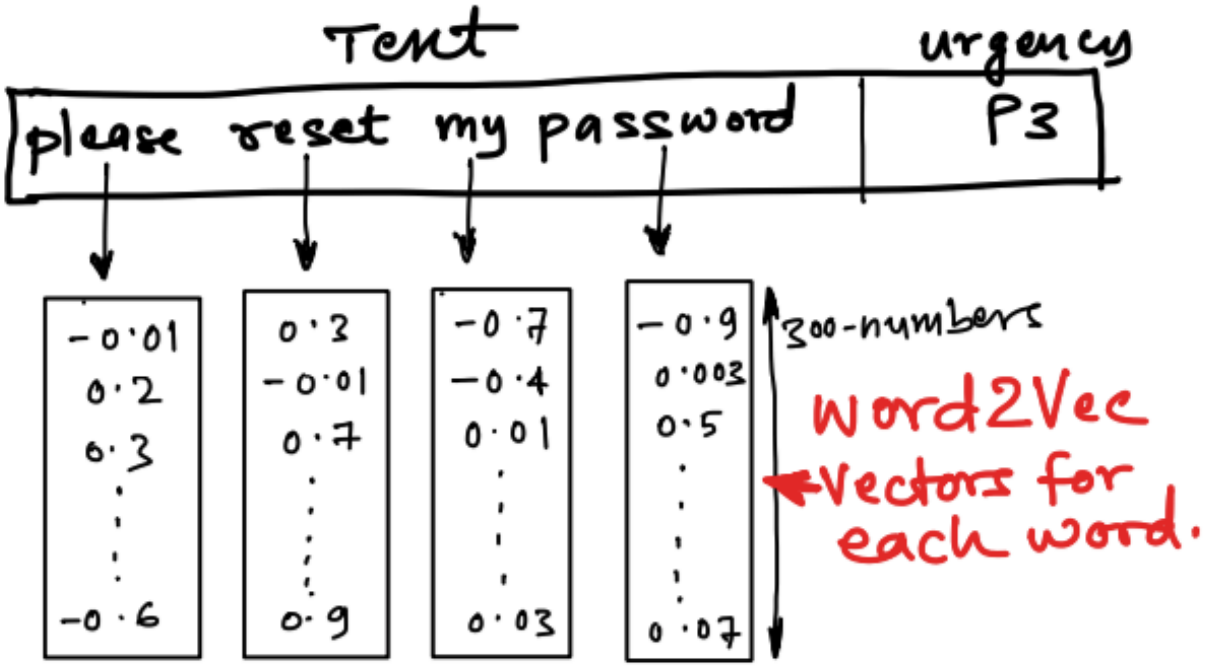
Use input layer W as embeddings



https://www.youtube.com/watch?v=UqRCEmrv1gQ

Word2vec

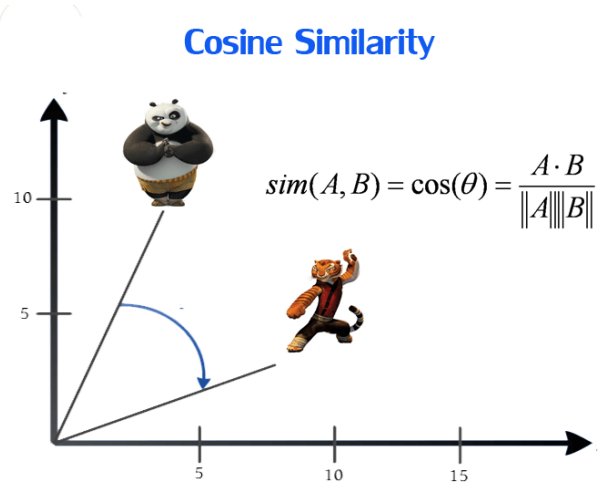
- Learns dense vector representation of each word
  - often 50-500D (vs. 10k in BOW); unsupervised

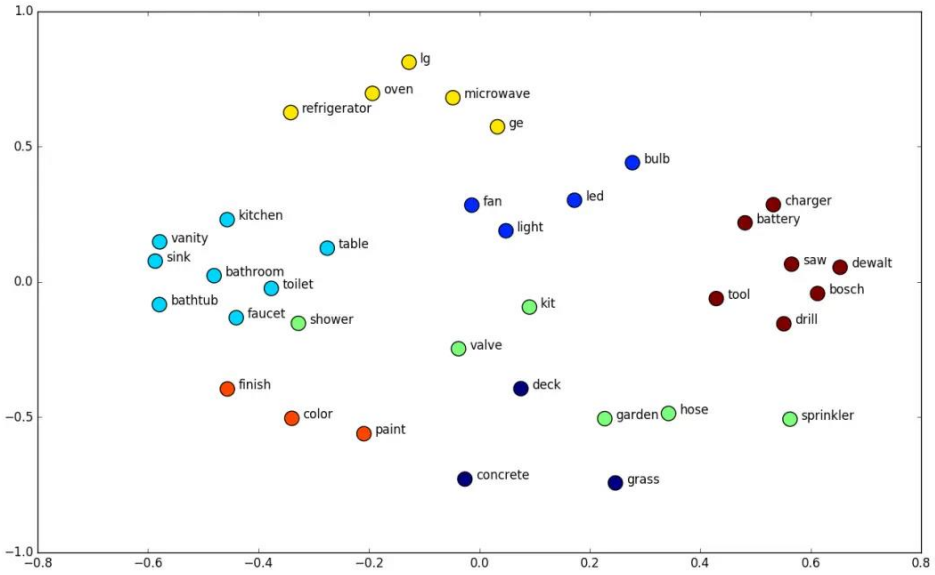


https://thinkingneuron.com/how-to-classify-text-using-word2vec/

Word2vec

- Learns dense vector representation of each word
  - often 50-500D (vs. 10k in BOW); unsupervised
- Words appear in similar context have similar vectors

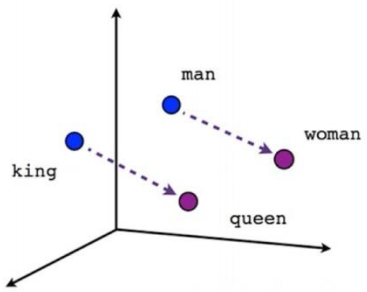




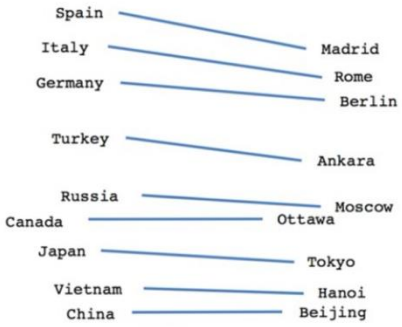
https://www.kaggle.com/code/samuelcortinhas/nlp7-word-embeddings

Word2vec

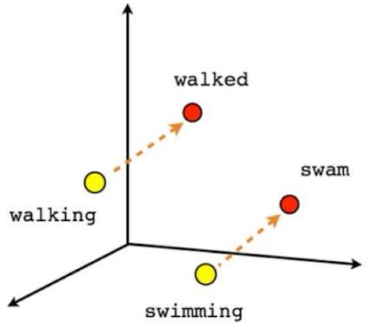
- Learns dense vector representation of each word
  - often 50-500D (vs. 10k in BOW); unsupervised
- Words appear in similar context have similar vectors
- Latent dimensions capture syntactic/semantic meaning



Male-Female



Country-Capital

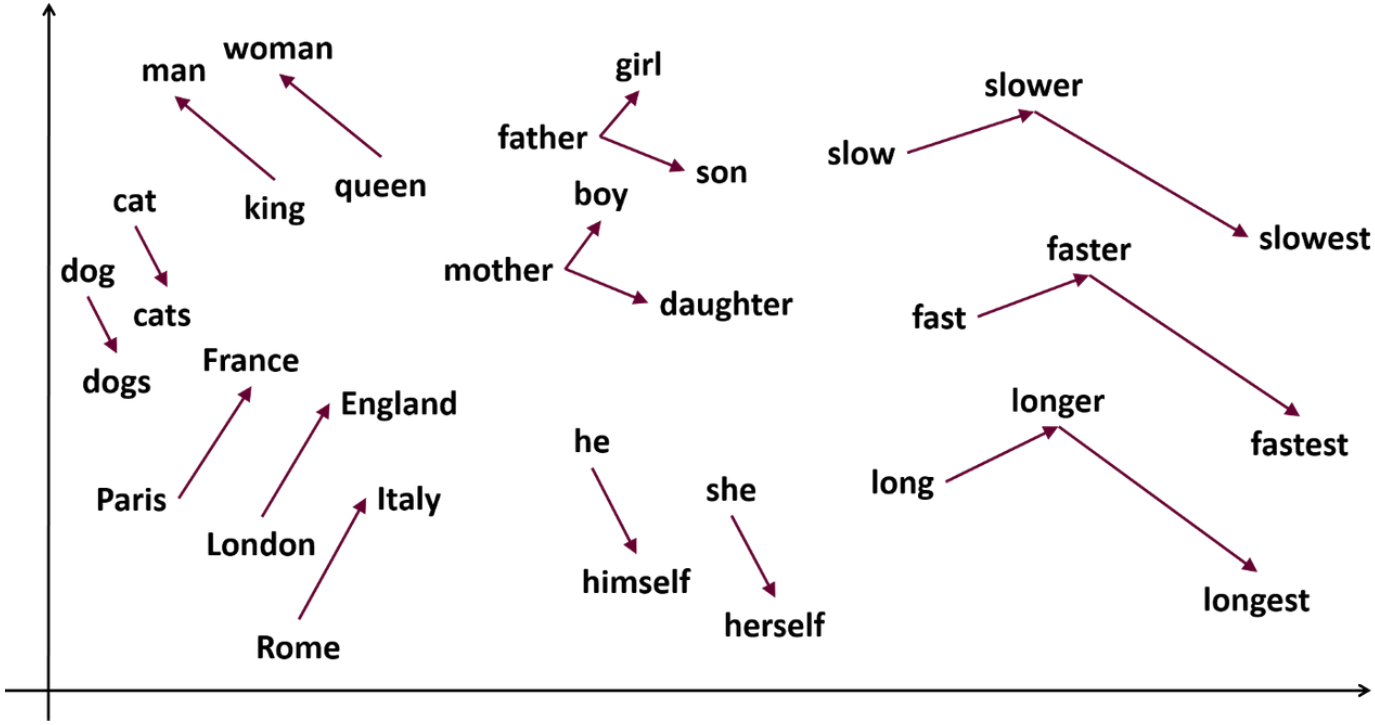


Verb tense

https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Word2vec

- Word analogy using vector operations
- E.g., “king” - “man” + “woman” = "queen"



https://www.kaggle.com/code/samuelcortinhas/nlp7-word-embeddings

Practical considerations for W2V

- Select the model training architecture
  - Use CBOW for small corpus
  - Use Skip-gram for large corpus, more dimensions
- Increase the training size, vector dimensions

```
model = Word2Vec(
    sentences=tokenized_sentences,
    vector_size=100,
    window=5,
    min_count=1,
    sg=0,
    workers=4
)

# Access word vectors
word_vector_apple = model.wv['apple']
print(f"Vector for 'apple':\n{word_vector_apple}\n")
```

Python implementation

```
# Find most similar words
similar_words_apple = model.wv.most_similar('apple', topn=3)
print(f"Words most similar to 'apple':\n{similar_words_apple}\n")

# Compute similarity between two words
similarity_apple_banana = model.wv.similarity('apple', 'banana')
print(f"Similarity between 'apple' and 'banana': {similarity_apple_banana}\

# Perform word analogies
result = model.wv.most_similar(positive=['king', 'woman'], negative=['man']
print(f"Analogy: king - man + woman = {result}\n")

# Save the trained model
model.save("word2vec_model.model")
```

Pre-trained word2vec models

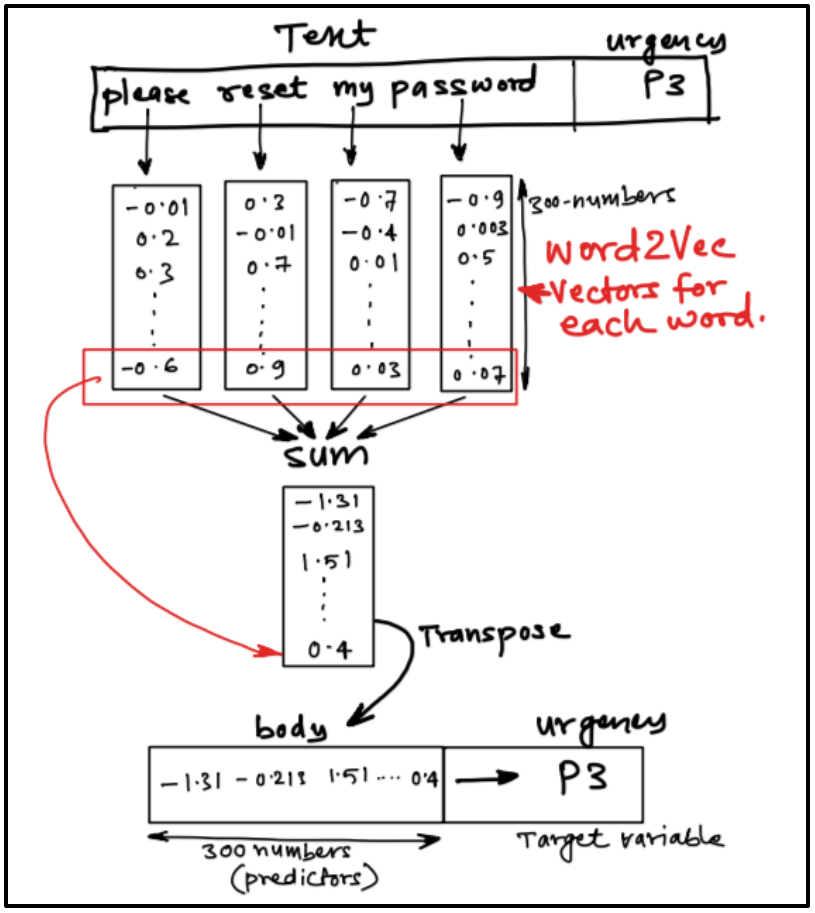
- Use pre-trained models on Facebook, Wikipedia, etc.
  - https://wikipedia2vec.github.io/wikipedia2vec/pretrained/
  - https://github.com/mmihaltz/word2vec-GoogleNews-vectors
- Fine-tuning the model using your own (small) data.
- Training w2v from scratch is computational heavy.
- Only do it for a specific task that pretrained embeddings don't perform well on and you have lots of training data.
- What if there are many words in our data which we could not find word vectors for in the pre-trained model?
  - Our corpus of app reviews might have a very different vocabulary than a news corpus that was used for the pre-trained w2v model.

Word2vec applications

- Semantic analysis
  - finding similar words; identify topics/themes
- Document classification/clustering
  - represent a document as the sum or average of all words' vectors
  - represent a document as a matrix (often used in Neural Net models)
- Social network analysis
  - treat nodes/users as words and obtain "user" vectors using w2v
  - obtain "sentences" of users using random walk on the network
- Linguistic bias analysis
  - by analyzing word associations via cosine similarity
  - E.g., gender stereotypes in job descriptions

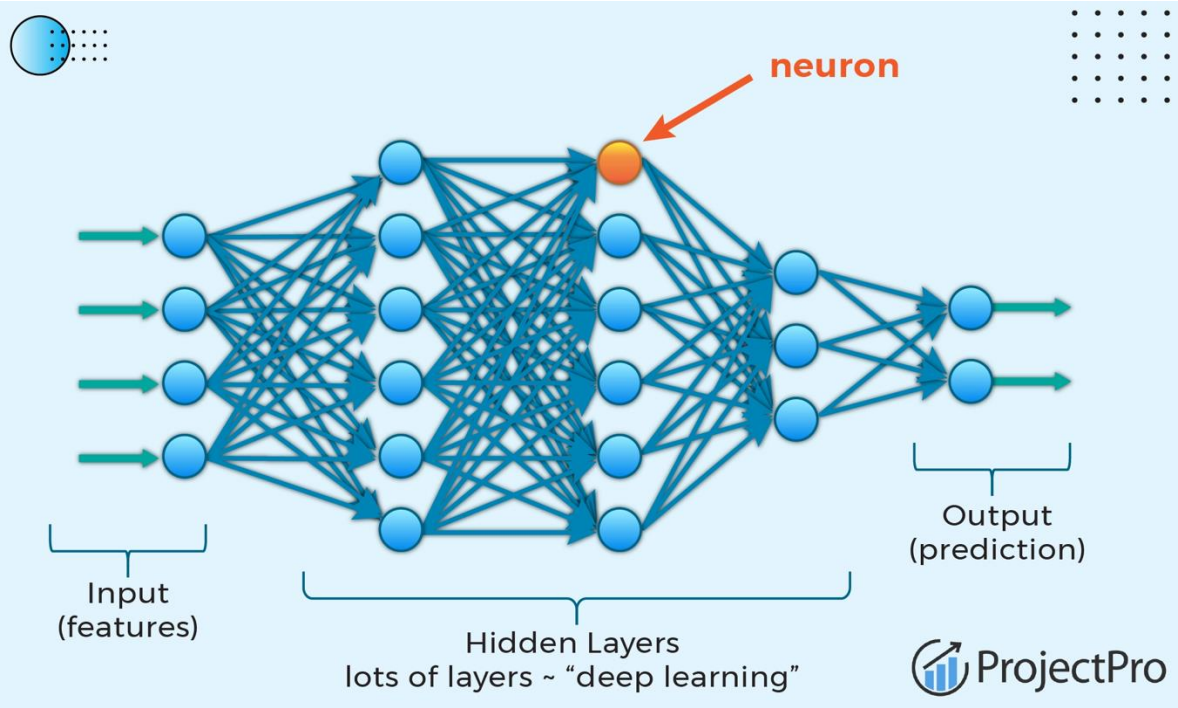
Word2vec for sentiment analysis

- Represent a text as its sum/avg word vector:
  - Apply logit model



Word2vec for sentiment analysis

- Represent a text as its sum/avg word vector:
  - Or use Neural Network model



https://www.projectpro.io/article/deep-learning-architectures/996

Word2vec for sentiment analysis

- Represent a text as a matrix (like an image):
  - Use padding to convert all texts to the same length. Longer sequences are truncated and shorter sequences are padded with 0's at the beginning. E.g. set "max_length = 250".

```
# Vectorize
X_train_ids = tokenizer.texts_to_sequences(X_train)
X_valid_ids = tokenizer.texts_to_sequences(X_valid)
X_test_ids = tokenizer.texts_to_sequences(X_test)

# Example
print(X_train_ids[0][:10])
```

```
[1, 116, 8, 29, 28, 944, 25, 10, 211, 22]
```

```
# Example
print(X_train_pad[0])
```

```
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   1 116   8  29  28 944  25
  10 211  22   5 265  74 187  22   8  27 108  89  28 944 226   5   5  98
 273 187  22  45 944 294  30  80  39 205  40  75   5  89  75   5]
```

- If N = 300, this text will be transformed to a 250 x 300 matrix.

https://www.kaggle.com/code/samuelcortinhas/nlp7-word-embeddings

Word2vec for sentiment analysis

- Then the task is like using deep learning models (CNN) to do image classification on a 250 x 300 matrix.
  - 250 is the text length after padding
  - 300 is the embedding length

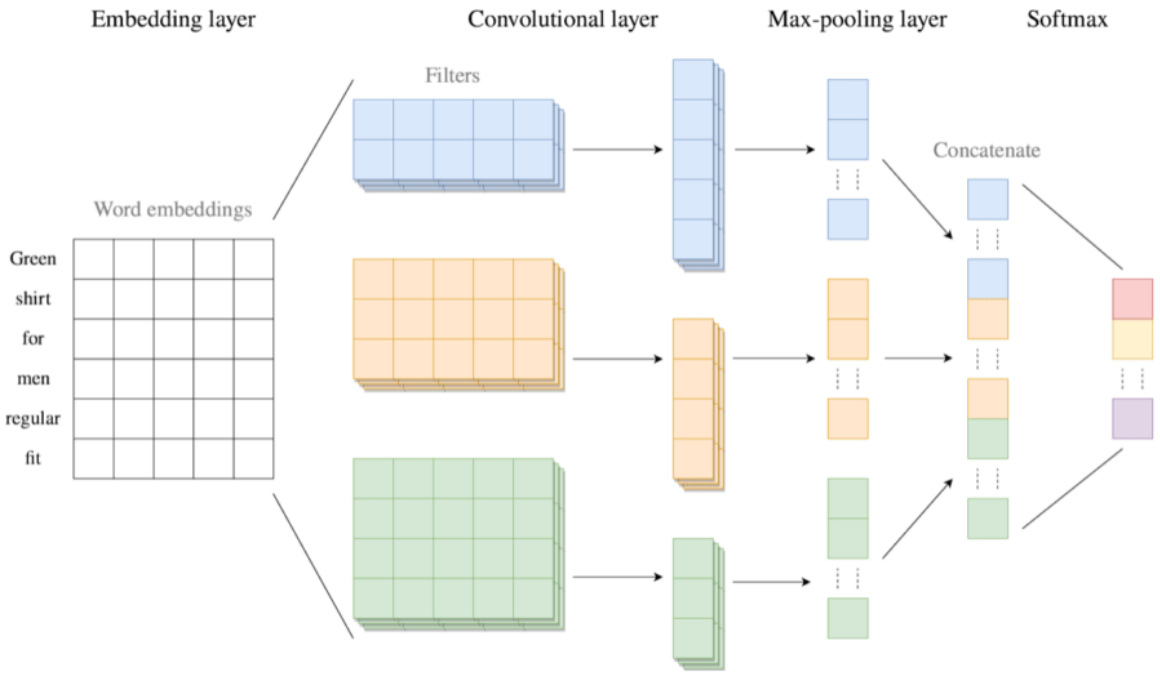
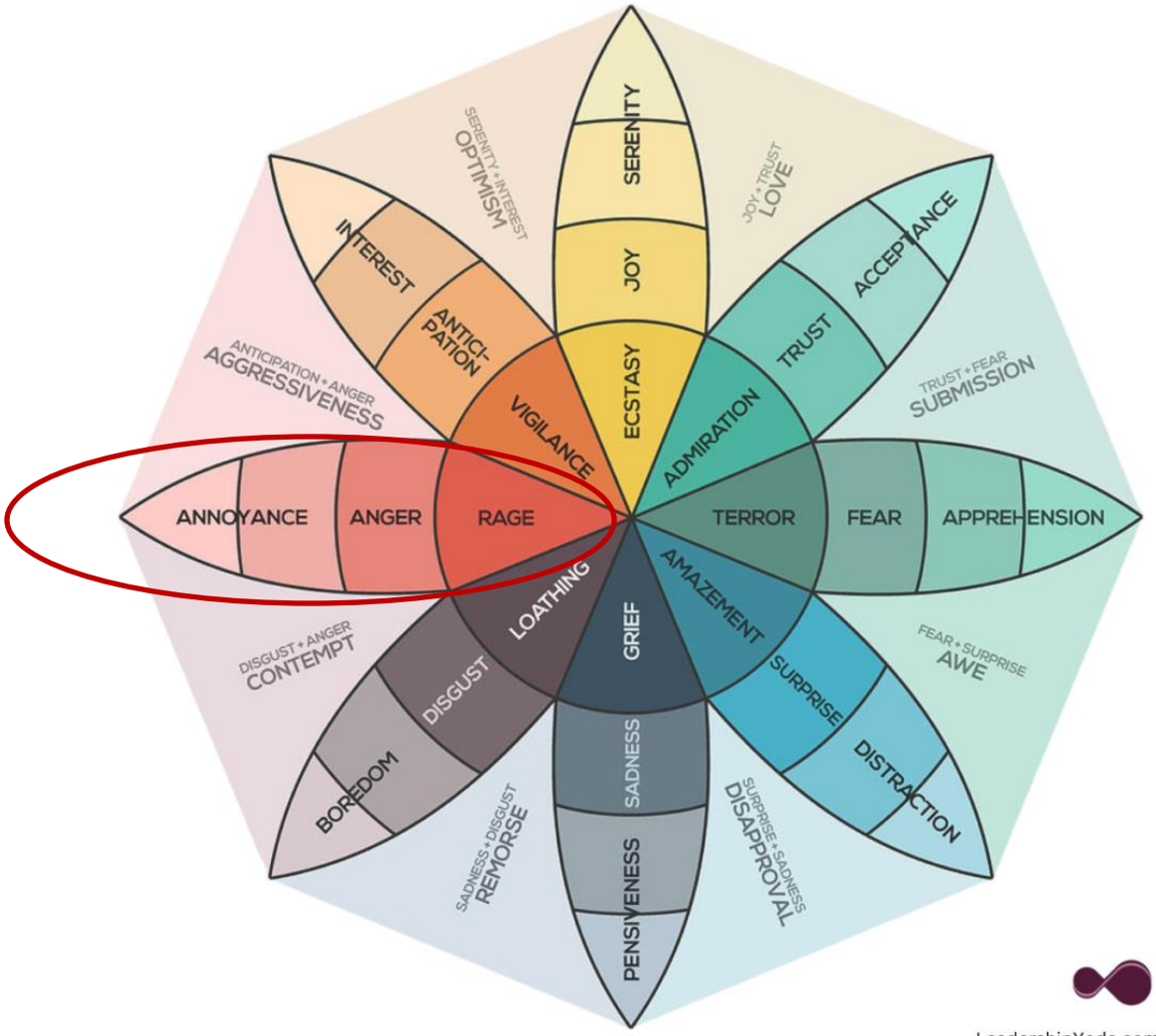


Figure 3: General CNN architecture for text classification

Plutchik's Wheel of Emotions



LeadershipYoda.com

Plutchik's Wheel of Emotions

- 8 primary emotions arranged in opposing pairs (2nd circle)
  - *joy, trust, fear, surprise, sadness, disgust, anger, and anticipation*
- Each emotion has a spectrum of intensity
  - **annoyance** is a less intense form of **anger** (outer circle)
  - while **rage** is a more intense form of **anger** (inner circle)
- Combinations of primary emotions produce new emotions
  - joy + trust -> love; joy + fear -> guilt

Primary		
<	<<	<<<
Admiration	Trust	Acceptance
Terror	Fear	Apprehension
Amazement	Surprise	Distraction
Grief	Sadness	Pensiveness

LeadershipYoda.com

Primary		
>>>	>>	>
Loathing	Disgust	Boredom
Rage	Anger	Annoyance
Vigilance	Anticipation	Interest
Ecstasy	Joy	Serenity

LeadershipYoda.com

https://leadershipyoda.com/understanding-emotions-using-the-wheel-of-emotions/

Plutchik's Wheel of Emotions

Can model different categories:

- 8 primary emotions (4 +/- pairs)
- 8 intensified primary emotions
- 8 weakened primary emotions
- 24 mixed emotions --->

No mixture of opposite ones

- joy + sadness = ?
- anger + fear = ?

Emotion 1	+	Emnotion 2	=	Emotions
Anger	+	Anticipation	=	Aggression
Anger	+	Joy	=	Pride
Anger	+	Trust	=	Dominance
Anticipation	+	Fear	=	Anxiety
Anticipation	+	Joy	=	Optimism
Anticipation	+	Trust	=	Hope
Disgust	+	Anger	=	Contempt
Disgust	+	Anticipation	=	Cynicism
Disgust	+	Joy	=	Morbidness
Fear	+	Disgust	=	Shame
Fear	+	Sadness	=	Despair
Fear	+	Surprise	=	Awe
Joy	+	Fear	=	Guilt
Joy	+	Surprise	=	Delight
Joy	+	Trust	=	Love
Sadness	+	Anger	=	Envy
Sadness	+	Anticipation	=	Pessimism
Sadness	+	Disgust	=	Remorse
Surprise	+	Anger	=	Outrage
Surprise	+	Disgust	=	Unbelief
Surprise	+	Sadness	=	Disappointment
Trust	+	Fear	=	Submission
Trust	+	Sadness	=	Sentimentality
Trust	+	Surprise	=	Curiosity

Predict emotion with Neural Nets

Identifying emotions using large-scale social media data:

- Created 665 emotion words by expanding the 24 emotion types with synonyms
  - Joy: *{"happy", "happiness", "joy", "joyful", "joyfully", "delighted", "feel-ingsunny", "blithe", "beatific", "exhilarated", "blissful", "walkingonair", "jubilant"}*
- Use each word as a hashtag seed to extract tweets using API
- Collected a dataset of 1/4 billion tweets in [2009, 2017].

EmoNet: Fine-Grained Emotion Detection with Gated Recurrent Neural Networks

Muhammad Abdul-Mageed
School of Library, Archival & Information Studies
University of British Columbia
muhammad.mageed@ubc.ca

Lyle Ungar
Computer and Information Science
University of Pennsylvania
ungar@cis.upenn.edu

Predict emotion with Neural Nets

Dimension	prec	rec	f-score
anger	97.40	97.72	97.56
anticipation	91.18	89.95	90.56
disgust	96.20	93.94	95.06
fear	94.97	94.38	94.68
joy	94.61	96.40	95.50
sadness	95.52	95.25	95.39
surprise	94.99	91.62	93.27
trust	96.36	97.58	96.96
All	**95.68**	**95.68**	**95.68**

Table 5: GRNNs results across 8 emotion dimensions. Each dimension represents three different emotions. For example, the *joy* dimension represents *serenity, joy* and *ecstasy*.

Gated recurrent neural net. VS. PAC:

- predicting the 8 inner circle emotions
- average f-score is over 0.9

	PAC	GRNNs		
Emotion	**f-score**	**prec**	**rec**	**f-score**
admiration	79.86	94.53	95.28	94.91
amazement	46.69	90.44	89.02	89.73
ecstasy	53.53	83.49	90.01	86.62
grief	48.10	85.07	81.13	83.05
loathing	2.99	83.87	54.17	65.82
rage	17.04	80.00	75.11	77.48
terror	47.00	91.15	84.01	87.44
vigilance	8.42	71.93	70.69	71.30
plutchik-1	**64.86**	**91.26**	**91.24**	**91.21**

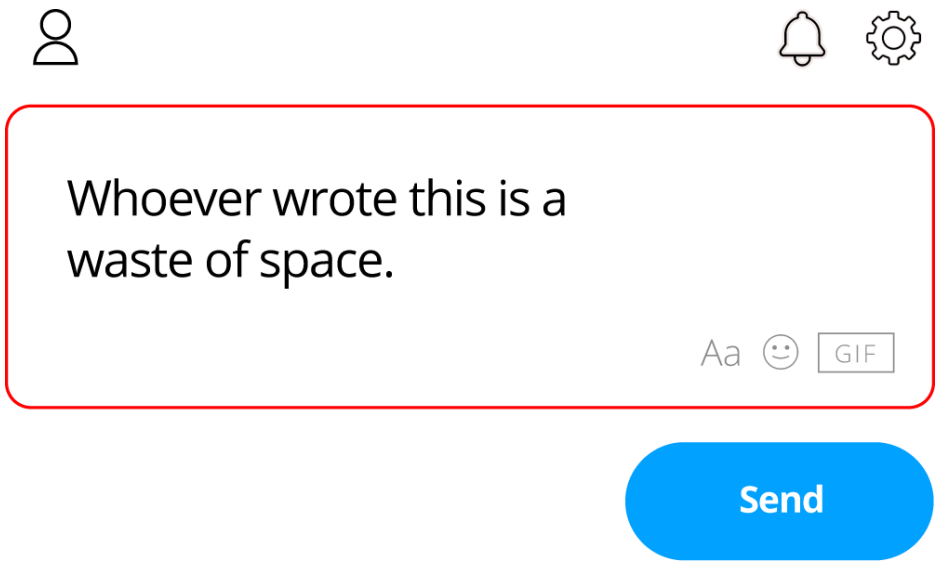
Mageed & Ungar. EmoNet: Fine-grained emotion detection with gated recurrent neural networks. ACL 2017.

Other text classification tasks

- Spam/clickbait detection
- Helpfulness rating on Q&A sites
- Complain/fake/rating of reviews
- Civility/toxicity/hateful speech
- Language identification of posts
- Fake news / bot detection
- Identify AIGC posts on RedNote

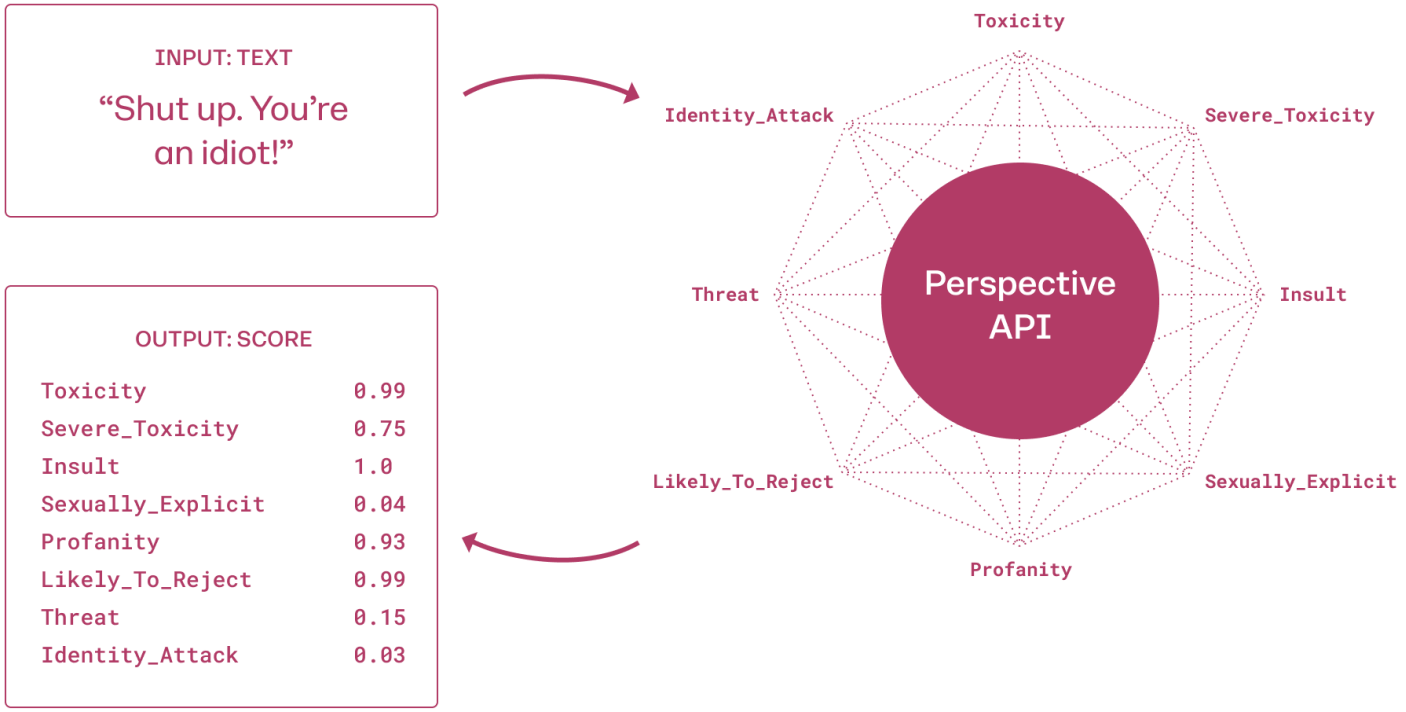
Toxicity prediction

- Good for content moderation for online platforms
- Help to maintain healthy / civic conversations
- Prevent online abuse and harassment



Perspective API by Google

- Python library:
  - https://developers.perspectiveapi.com/s/docs-sample-requests
- Other Github package:
  - https://github.com/baishalidutta/Comments-Toxicity-Detection



POLICY FORUM

SOCIAL SCIENCE

The science of fake news

Addressing fake news requires a multidisciplinary effort

By **David M. J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, Jonathan L. Zittrain**

The rise of fake news highlights the erosion of long-standing institutional bulwarks against misinformation in the internet age. Concern over the problem is global. However, much remains unknown regarding the vulnerabilities of individuals, institutions, and society to manipulations by malicious actors. A new system of safeguards is needed. Below, we discuss extant social and computer science research regarding belief in fake news and the mechanisms by which it spreads. Fake news has a long history, but we focus on unanswered scientific questions raised by the proliferation of its most recent, politically oriented incarnation. Beyond selected references in the text, suggested further reading can be found in the supplementary materials.

WHAT IS FAKE NEWS?

We define "fake news" to be fabricated information that mimics news media content in form but not in organizational process or intent. Fake-news outlets, in turn, lack the news media's editorial norms and processes for ensuring the accuracy and credibility of information. Fake news overlaps with other information disorders, such as misinformation (false or misleading information) and disinformation (false information that is purposely spread to deceive people).

Lazer et al. 2018, *Science*

Fake news

- Definition:
  - Purposefully crafted, sensational, emotionally charged, misleading or totally fabricated information that mimics the form of mainstream news (Zimdars & McLeod, 2020).
- Three varieties:
  - Disinformation: Content that is intentionally false and designed to deceive people and cause harm.
  - Misinformation: Misleading content but the person sharing doesn't realize that it is false or misleading.
  - Malinformation: Genuine information that is shared out of context with an intent to cause harm.

Fake news

Misinformation

The spread of false information without malicious intent.

Example: Your mom shares an outdated video of a natural disaster in your town, believing it happened recently.

Impact: Often leads to public confusion or manipulation.

Disinformation

Spreading false or misleading info to deceive or gain an advantage.

Example: During the pandemic, multiple disinformation campaigns spread misleading information about COVID-19 – many of which led to actual harm.

Impact: Can cause significant harm, like health risks, and societal disruption.

TRUSTLAB

Fake news example

Soldiers saving lives or killing people?

Fakey game

- Play the game: https://fakey.osome.iu.edu/login

Hard for humans to identity

Trump tries, fails to circumvent court ruling on unnecessary Portland troop deployment

Donald Trump sees an imaginary crisis unfolding in Portland, which he intends to resolve by deploying Guard troops from other states, against Oregon's wishes.

375 people liked or shared this

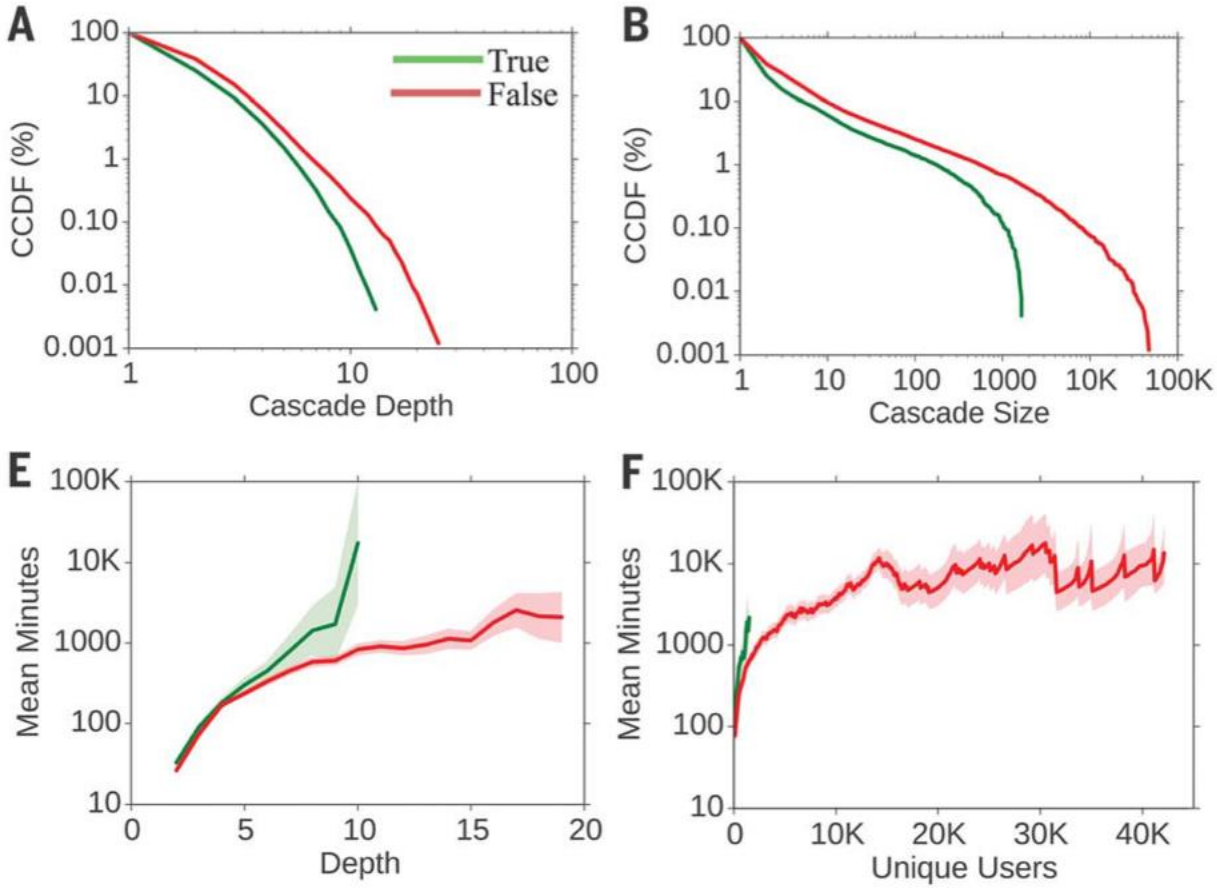
Hard for humans to identity

Georgian opera singer charged with coup attempt

Georgia has charged five politicians with attempting to overthrow the government

22,219 people liked or shared this

Lies spread faster than the truth



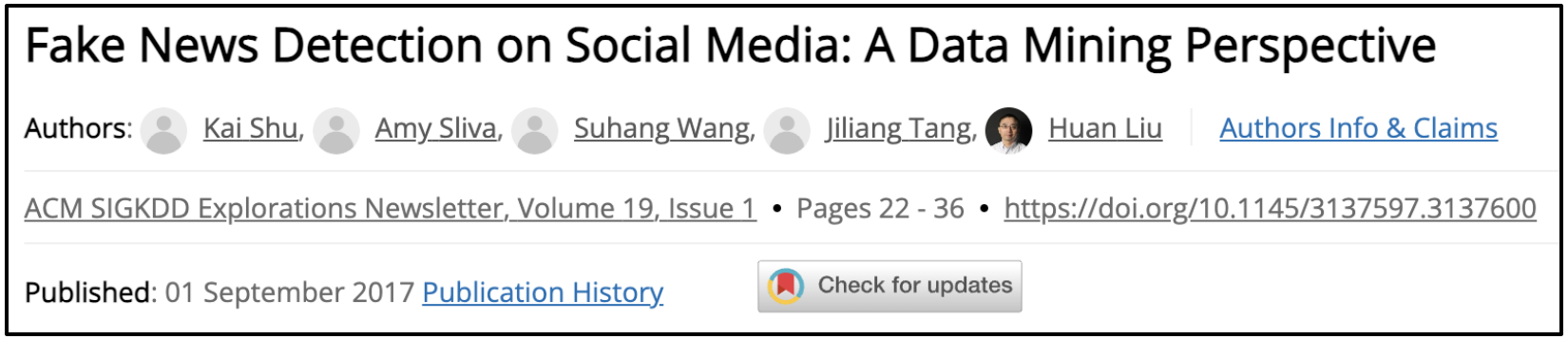
https://www.science.org/doi/10.1126/science.aap9559

Fake news impact

- Undermines online trust, democracy, and public health (e.g., the COVID-19 misinformation).
- Increase polarization and echo chamber effects.
- Erosion of public trust in news media.

Fake news detection

- A classification task (document-level or source-level)
- High-quality data: https://www.snopes.com/fact-check/
- Prediction using three sets of features:
  - Source-level
  - Content-level
  - Spreading pattern (network-level)



Fake News Detection on Social Media: A Data Mining Perspective

Authors: Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, Huan Liu | Authors Info & Claims

ACM SIGKDD Explorations Newsletter, Volume 19, Issue 1 • Pages 22 - 36 • https://doi.org/10.1145/3137597.3137600

Published: 01 September 2017 Publication History

Check for updates

Characteristics of fake news

- Fake news uses distinct linguistic features for virality.
- Examples: “NEVER Trust This Candidate!”.
- **Stylistic and Syntactic**: Sensational headlines, high adjective ratios, and all-caps usage (e.g., clickbait patterns in BuzzFeed).
- **Psycholinguistic**: Negative emotions (anger, fear) and persuasive language, often detected via LIWC tools (e.g., higher negation).
- **Readability**: Simpler language, lower Flesch-Kincaid scores, and repetitive structures for broad appeal and sensational tone.
- **Lexical/Semantic**: Low vocabulary richness and emotional words.

Combating fake news

- **Individual actions:**
  - Verify sources (e.g., check domain credibility via Media Bias).
  - Cross-reference with reputable outlets and fact-checking sites.
  - Be skeptical of emotionally charged or simplistic language.
- **Technological solutions:**
  - Browser extensions (e.g., NewsGuard) for real-time detection.
  - Importance of open datasets for research and model training.
- **Policy and education:**
  - Media literacy programs to teach critical thinking.
  - Platform regulations to reduce bot activity and algorithmic bias.

Bot detection

- Social media platforms and bots amplify the spread of fake news via algorithms prioritizing engagement.
- Construct linguistic features on aggregated posts.
- Key linguistic characteristics associated with bots:
  - Bots display uniform sentiment with low polarity and high consistency; with low complexity in emoji usage.
  - Bots show low emotional diversity, amplifying negative emotions like anger and anxiety in polarizing topics.
  - Bots underuse hedge words, showing higher certainty.
  - Bots exhibit low lexical diversity, high repetition.
  - Posts use uniform tweet lengths.

Course notes

- Upload paper presentation slides on Canvas
- Midterm exam next week (W7)
  - Be on time!
  - Minimal coding
  - 2-hour in class
  - 25% extra time for students with SEN
- User modeling in Week 8
- User modeling V2 in Week 9