ML for Text Analytics in Accounting Research

Dr. Richard M. Crowley
rcrowley@smu.edu.sg

https://rmc.link/ / Slides @ rmc.link/ML2024

Text as data in Accounting Research

Text data that has been studied

- Firms
  - Letters to shareholders
  - Annual and quarterly reports
  - 8-Ks
  - Press releases
  - Conference calls
  - Firm websites
  - Twitter posts
  - CSR reports
- Investors
  - Blog posts
  - Social media posts
- Intermediaries
  - Newspaper articles
  - Analyst reports
  - Hobbies analysts/investors
- Government
  - FASB exposure drafts
  - Comment letters
  - IRS code
  - Court cases

What can we do with text?

Textual analysis

Using textual analysis, we can extract meaningful information from text

- This could be as simple as extracting specific words/phrases/sentences
  - Mentions of dollar amounts, dates, etc. (Hope, Hu, and Lu 2016)
  - Mentions of Brexit (Hassan et al. 2024)
- This could be as complex as extracting latent (hidden) patterns/structures within text
  - Sentiment
  - Content
  - Emotion
  - Writer characteristics
- Often referred to as text mining (in CS) or textual analysis (in accounting)

What about NLP?

> NLP is a field devoted to understanding how to understand human language

- NLP stands for **N**atural **L**anguage **P**rocessing
- It is a very diverse field within CS
  - Grammar/linguistics
  - Conversations
  - Conversion from audio, images
  - Translation
  - Dictation
  - Generation

Why discuss NLP?

Consider the following situation:

> You have a collection of 40 million sentences, and you want to know which are accounting relevant

- Without NLP:
  1. Hire an RA/mechanical turk army…
  2. Use a dictionary: Words/phrases like “earnings,” “profitability,” “net income” are likely to be in the sentences
- With NLP:
  1. We could associate sentences with outside data to build a classifier (supervised approach)
  2. We could ask an algorithm to learn the structure of all sentences, and then extract the useful part *ex post* (unsupervised)

A brief history of text analytics in accounting research

1980s and 1990s

> Manual content analysis

- Read through “small” amounts of text, record selected aspects

Indexes

- Ex.: Botosan (1997 TAR): For firms with low analyst following, more disclosure $\Rightarrow$ Lower cost of equity
  - Index of 35 aspects of 10-Ks
- Covered in detail in Cole and Jones (2004 JAL)
  - Most use small samples
  - Often use select industries

Readability

- Automated starting with Dorrell and Darsey (1991 JTWC) in accounting…
- At least 32 studies on this in the 1980s and early 1990s per Jones and Shoemaker (1994 JAL)
  - Only 2 use full docs
  - Only 2 use >100 docs

2000s

> Automation

- With computer power increasing, two new avenues opened:
  1. Do the same methods as before, at scale
     - Ex.: Li (2008 JAE): Readability, but with many documents instead of <100
       - Or use existing data: Loughran and McDonald (2014)
  2. Implementing statistical techniques (often for tone/sentiment)
     - For instance, sentiment classification with Naïve Bayes, SVM, or other statistical classifiers
       - Antweiler and Frank (2005 JF)
       - Das and Chen (2007 MS)
       - Li (2010 JAR)

Early 2010s

> Dictionaries take the helm

- Loughran and McDonald (2011 JF) points out the misspecification of using dictionaries from other contexts
  - Also provides a sets of positive, negative, modal strong/weak, litigious, and constraining words (available here)
- Subsequent work by the authors provides a critique:

> Applying financial dictionaries “without modification to other media such as earnings calls and social media is likely to be problematic” (Loughran and McDonald 2016)

- A lot of papers ignore this critique, and are still at risk of *misspecification*

Late 2010s

> Fragmentation and new methods

- Loughran and McDonald dictionaries frequently used
- Bog index is perhaps a new entrant in the Fog index vs document length debate (Bonsall, Leone, Miller, Rennekamp 2017)
- LDA methods first published in Accounting/Finance in Bao and Datta (2014 MS), with a handful of other papers following suit.
  - Dyer, Lang and Stice-Lawrence (2017); Huang, Lehavy, Zang and Zheng (2018); Brown, Crowley and Elliott (2020)
- Word2vec and other embedding methods on the fringe

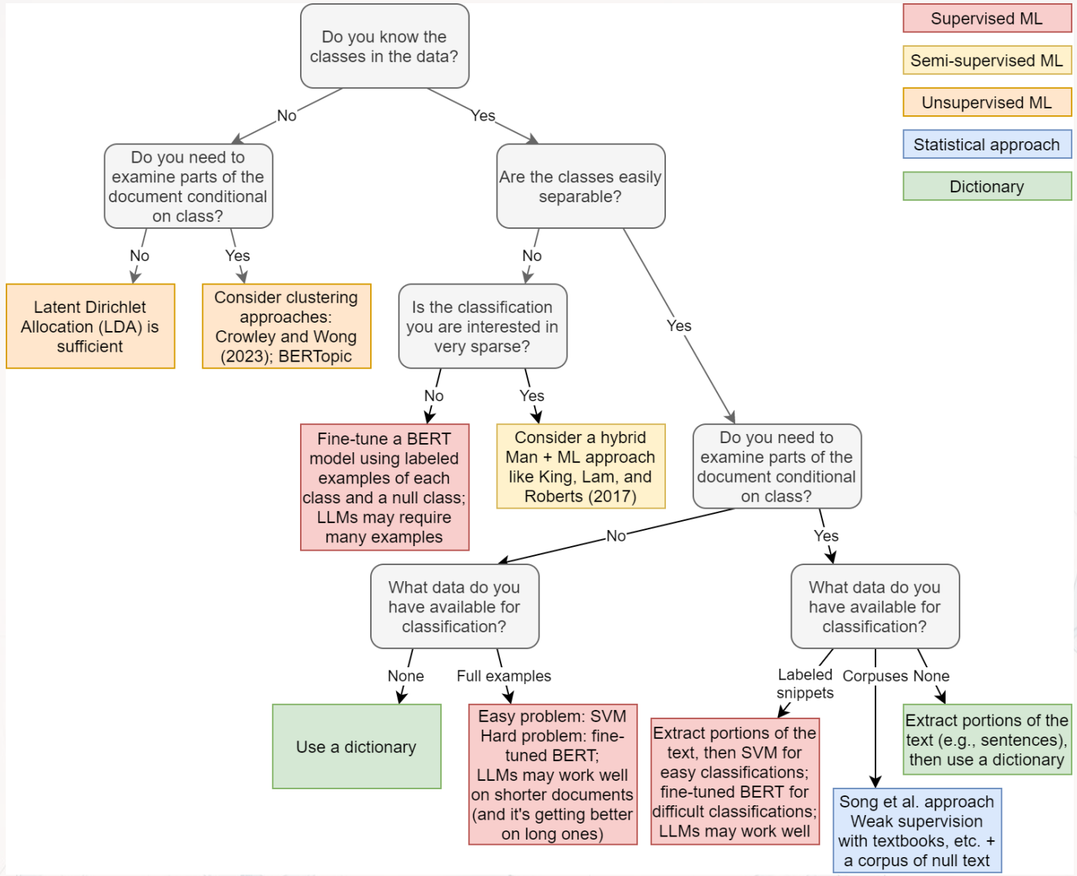
2019 to present

> Transformer neural networks

- These methods help to directly process larger sets of text
  - USE provides a general purpose pretrained model to convert sentences to embedding
  - BERT provides both a general purpose model and the ability to fine-tune to specific tasks
  - FinBERT (Huang et al. 2023) is a BERT trained on financial text
  - GPT provides a fine-tunable model that generally requires fewer examples to train

What method(s) should one use?

A quick guide



The diagram represents my own subjective opinion on what method to use when. It generally aligns with my publicly available work through April 2024.

Overview of methods: Word-based

1. Dictionaries: Researcher-defined sets of words or phrases.
    - A word is either included or not. May include weights.
2. Song and Wu (2008)/Schutze et al. (2008): Use textbooks or other corpuses to supervise classification.
    - Uses words' inclusion in a corpus relevant to the class (like a textbook) to indicate a positive match. Uses words' inclusion in relevant to the documents *but not the class* to remove irrelevant terms.
3. SVM: Classify using a separating hyperplane.
    - A simple ML classifier. Uses word counts to predict a classification. Requires labeled examples to train.
4. *King, Lam, and Roberts (2017)*: Iterate between human and machine coding
    - Start with a dictionary, and hand code correct and incorrect matches. Feed correct matches to an ML algorithm, and let it suggest a broader dictionary to check. Repeat, then hand verify.
    - Other approaches involve determining **similar words** (in context) to those of the initial dictionary. This is noisy but more automated.
5. LDA: Let the machine determine the topics.
    - Uses word counts, Bayesian statistics, and simulation to identify topics, which are represented as weighted dictionaries. Technically an embedding

Methods 1 and 2 are not ML/AI, but everything else is.

Overview of methods: More than words

6. **Clustering approaches**: Create granular measures of topic or context *within-document*
    - First, create an embedding of parts of the document. Then cluster using kmeans or the like + weak supervision through simulation (such as Gap statistic)
7. **Transformer methods** (USE, BERT): Make embeddings and map embeddings nonparametrically to classes
    - The transformer creates machine-understandable embeddings of documents or parts thereof. Fine-tuning allows for converting embeddings into understandable classifications.
8. **GPT methods**: Still transformers under the hood
    - Instead of fine tuning, provide worked out examples as part of the input to the model alongside the data to be labeled.

Words and their representations

Words

- Words are an interim building block of language (See Harris 1954 for a good discussion)
- Words have a [somewhat] unique meaning
  - This is why methods like Naïve Bayes, cosine similarity, and SVM on word counts had some success in the literature
  - This also allows for constructing dictionaries

> But words are not independent

- Words share sounds, syllables, denotations, connotations, linguisitic structure, and more
  - Representing this allows for better measurement on text

Word embeddings

Word embeddings convert words into an N-dim vector capturing facets of the word, primarily its meaning in context.

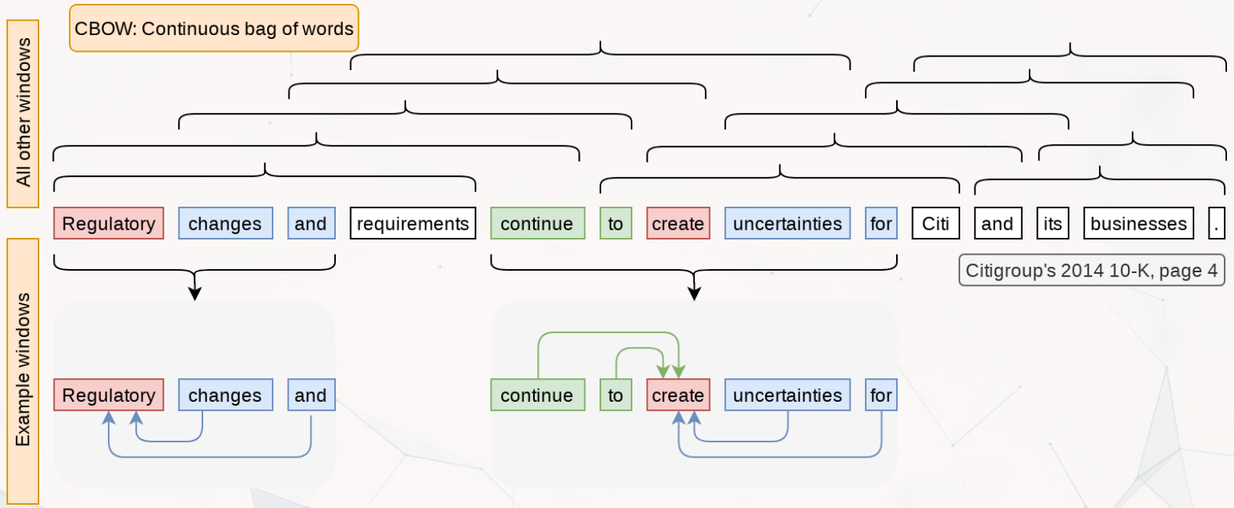
Word embeddings: A simple example

words	f_animal	f_people	f_location
dog	0.5	0.3	-0.3
cat	0.5	0.1	-0.3
Bill	0.1	0.9	-0.4
turkey	0.5	-0.2	-0.3
Turkey	-0.5	0.1	0.7
Singapore	-0.5	0.1	0.8

- The above is a simplified illustrative example
- Notice how we can tell apart different animals based on their relationship with people
- Notice how we can distinguish turkey (the animal) from Turkey (the country) as well

How can we make these embeddings?

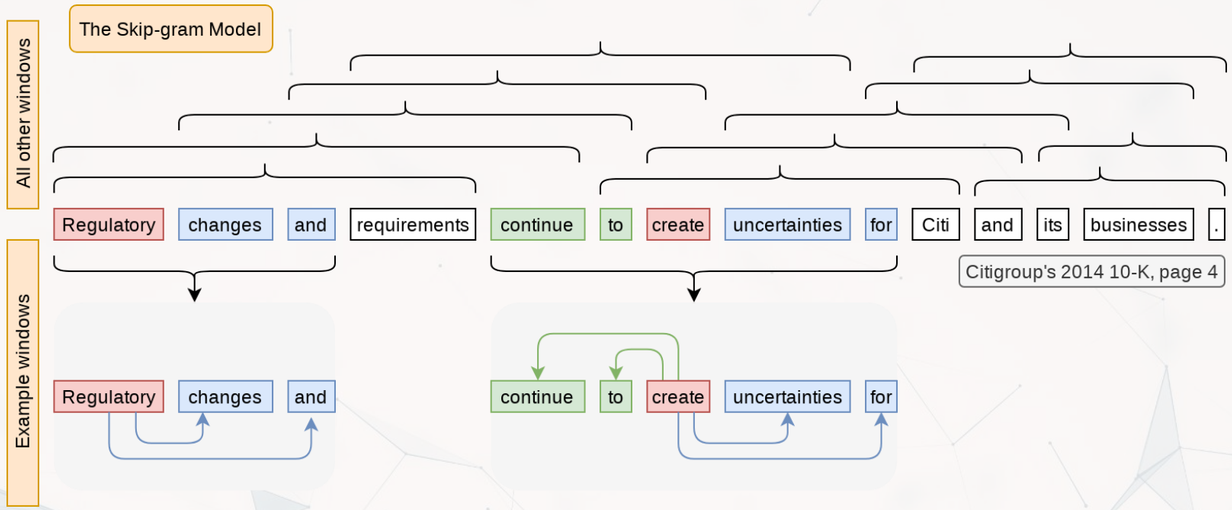
> Infer a word's meaning from the words around it



> Refered to as CBOW (continuous bag of words)

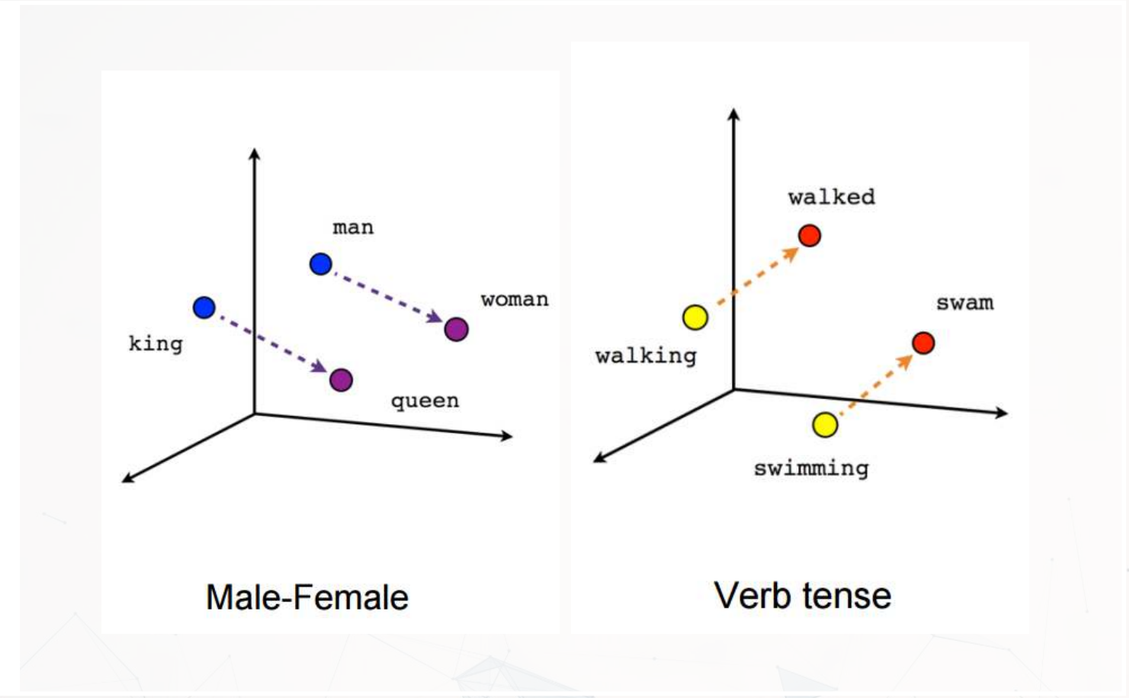
How else can we infer meaning?

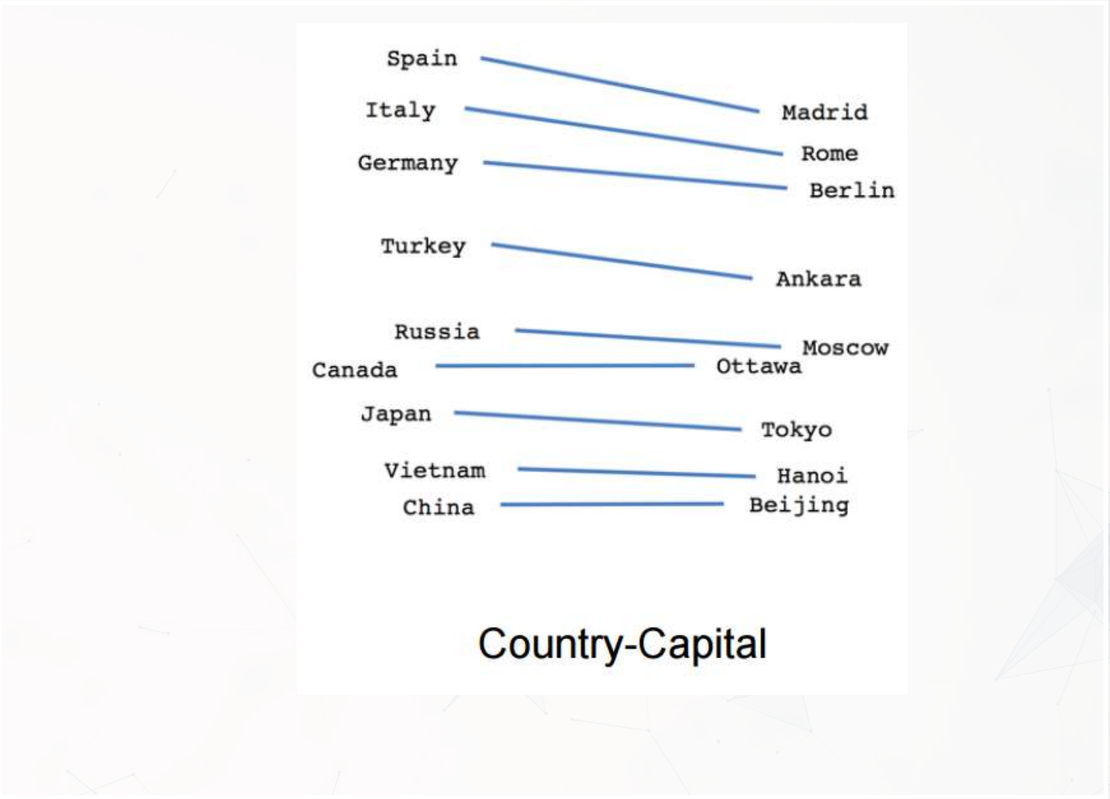
> Infer a word's meaning by *generating* words around it



> Refered to as the Skip-gram model

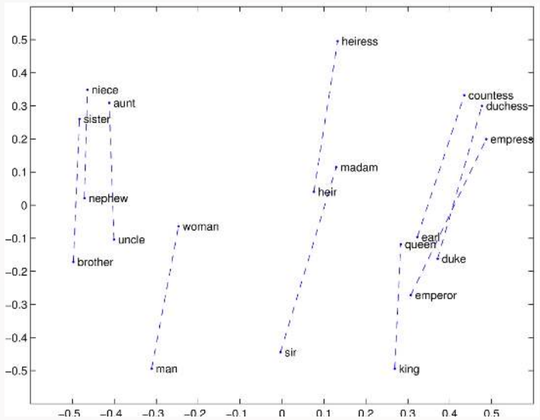
What it retains: word2vec

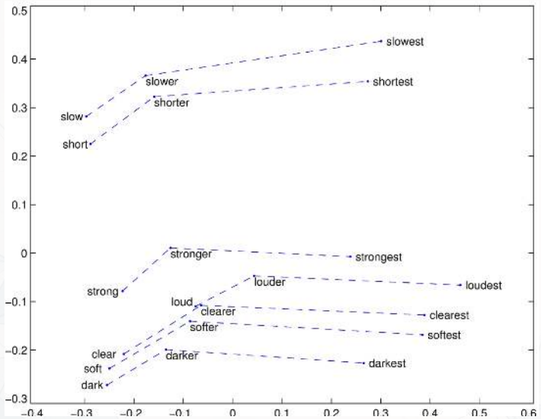


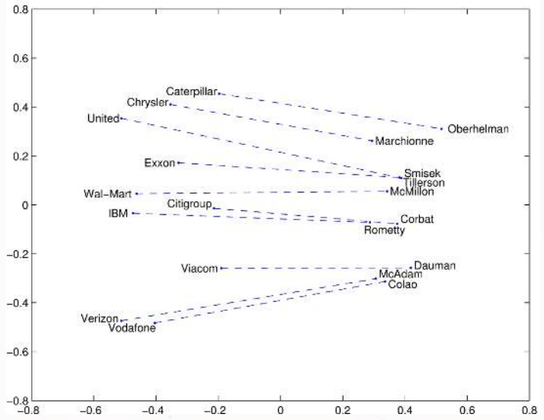


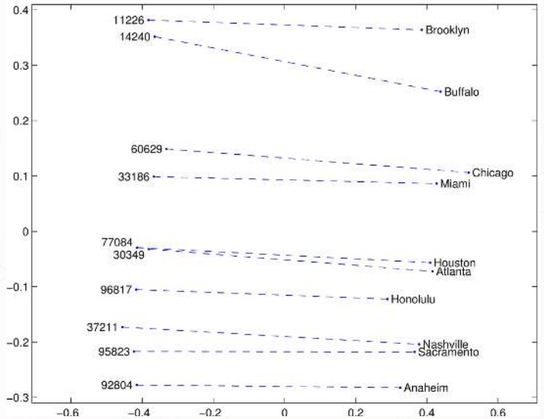
> Relations are retained as vectors between points (distance + direction)

What it retains: GloVe









Examining word2vec: Analogies

> man : King :: woman : ?

- Mathematically: $King - man + woman = ?$

```
1  base_w2v.most_similar(positive=['King', 'woman'], negative=['man'])
```

```
('Queen', 0.5515626668930054) ('Oprah_BFF_Gayle', 0.47597548365592957)
 ('Geoffrey_Rush_Exit', 0.4646016657352475) ('Princess', 0.4533674716949463)
 ('Yvonne_Stickney', 0.4507041573524475) ('L._Bonauto', 0.4422135353088379)
 ('gal_pal_Gayle', 0.4408389925956726) ('Alveda_C.', 0.4402790665626526)
 ('Tupou_V.', 0.4373864233493805) ('K._Letourneau', 0.4351031482219696)
```

The sleight of hand behind this

- Word2Vec implementations usually bar a word in the analogy from being an output
  - E.g., it will never report **man : King :: woman : King**
    - But this is actually the mathematical answer

```
analogy = base_w2v['King'] + base_w2v['woman'] + base_w2v['man']
analogy = analogy / np.linalg.norm(analogy)
print('King', np.linalg.norm(analogy - base_w2v['King']))
```

```
King 1.9888592
```

```
print('Queen', np.linalg.norm(analogy - base_w2v['Queen']))
```

```
Queen 2.7364814
```

It’s still pretty good though!

- Note that since word2vec’s original answer was Queen, this implies it was second best
  - If Queen is the closest word to King, then this would be mathematically uninteresting
    - It’s actually 7th though!

```
1  base_w2v.most_similar('King')
```

```
[('Jackson', 0.5326348543167114), ('Prince', 0.5306329727172852), ('Tupou_V.', 0.5292826294898987), ('KIng',
0.5227501392364502), ('e_mail_robert.king_@', 0.5173623561859131), ('king', 0.5158917903900146), ('Queen',
0.5157250165939331), ('Geoffrey_Rush_Exit', 0.49920955300331116), ('prosecutor_Dan_Satterberg',
0.49850785732269287), ('NECN_Alison', 0.4912859499454498 3)]
```

What are word embeddings good for?

1. You care about the words used, by not stylistic choices
    - Abstraction
2. You want to crunch down a bunch of words into a smaller number of dimensions without running any bigger models (like LDA) on the text.
    - E.g., you can toss the 300 dimensions of the Google News model to a Lasso or Elastic Net model
        - This is a big improvement over the past method of tossing vectors of word counts at Naive Bayes
            - 300 << # of unique words
3. You want synonyms for a set of words that are selected in a less-researcher-biased fashion
    - You can even get n-gram synonyms this way
    - A popular method for augmenting small dictionaries

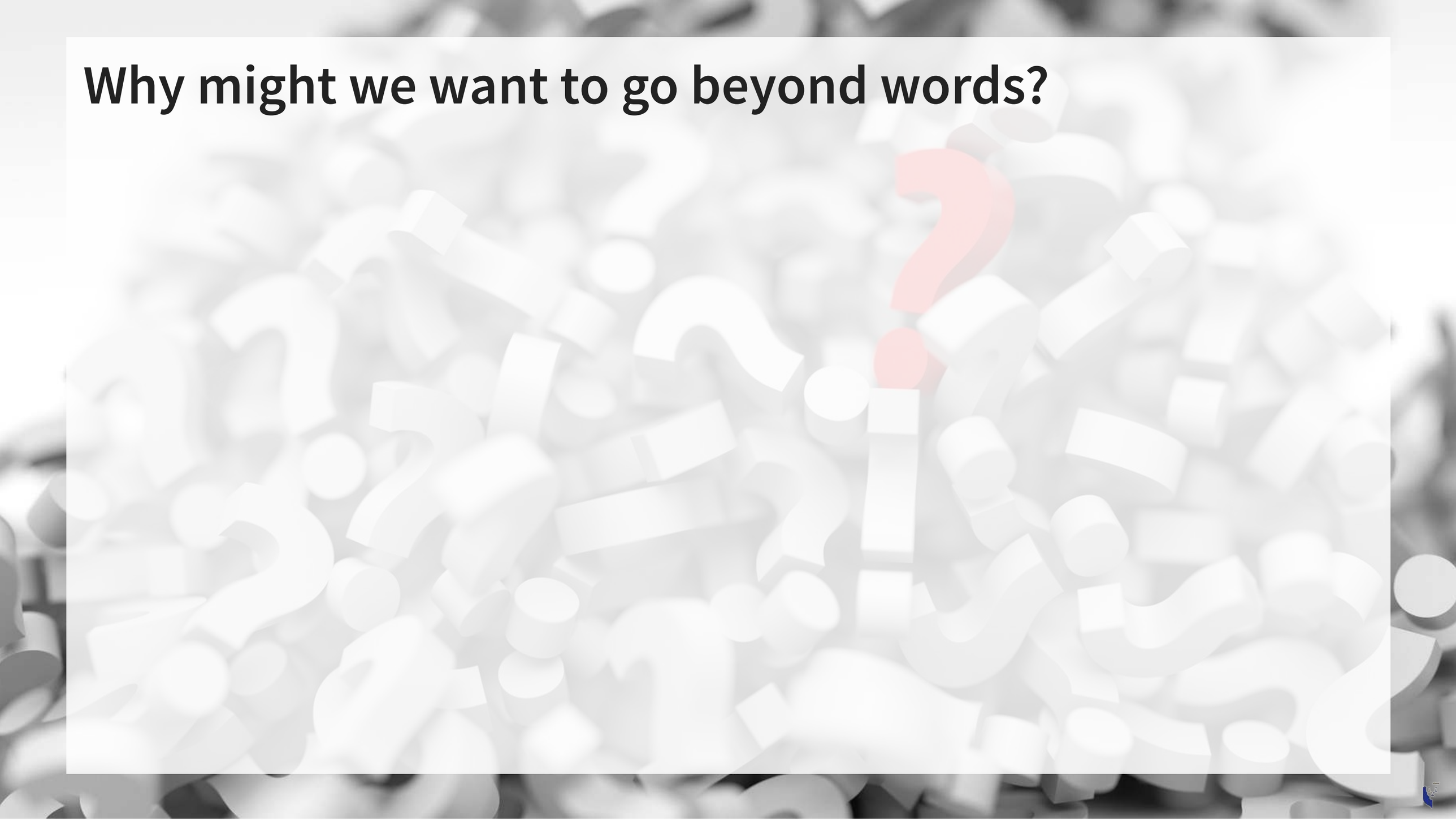
Demo: Trying out word2vec

> Colab file available at https://rmc.link/colab_w2v

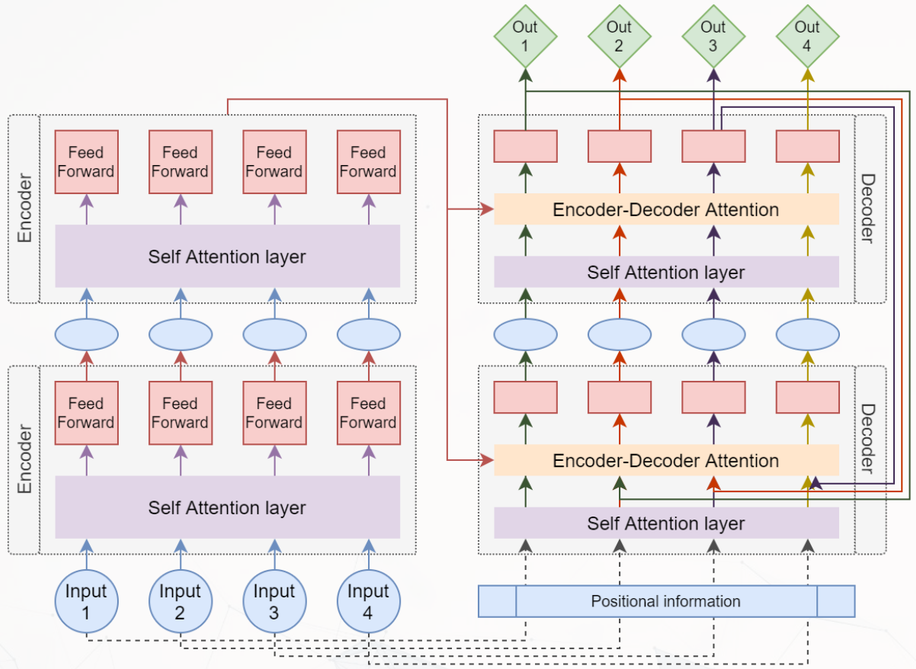
- This exercise is designed to help you understand about what word2vec is good at
  - As well as what it isn't good at

More than words

Why might we want to go beyond words?



Transformers: How we use ML to go beyond words



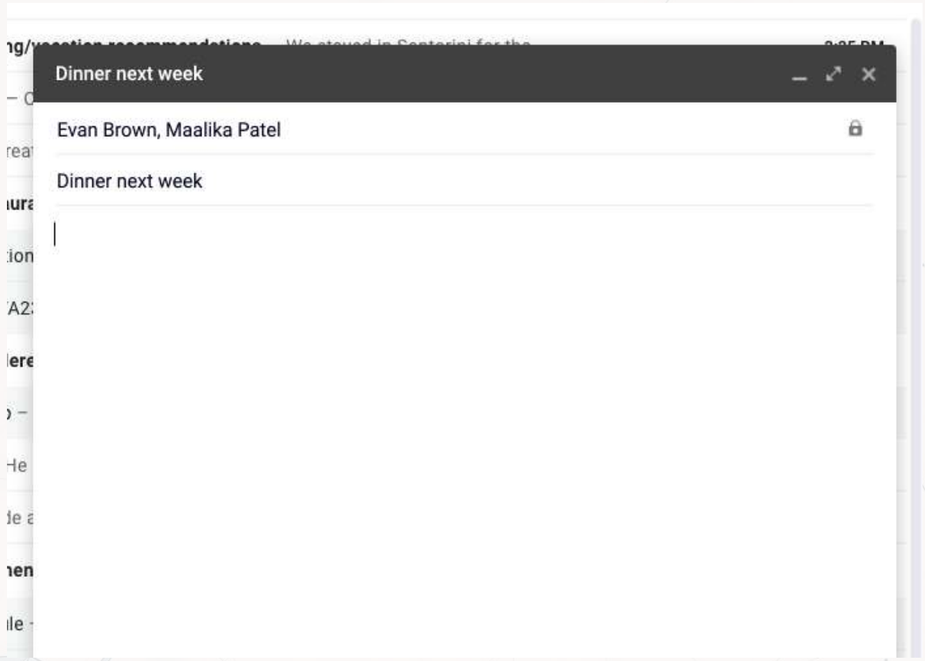
- Note that some models only keep the encoder stack on the left (BERT), and some only keep the decoder stack on the right (GPT). Some use both (T5).
  - Encoder builds embeddings
  - Decoder generates language

A simple approach for sentences: USE

- USE is based on DAN (Deep Averaging Networks) and Transformers
  - There are variants using each
    - DAN is faster
    - Transformer is more accurate
  - USE learns the meaning of sentences via words' implied meanings
- Learn more: Original paper and TensorFlow site
- In practice, it works quite well

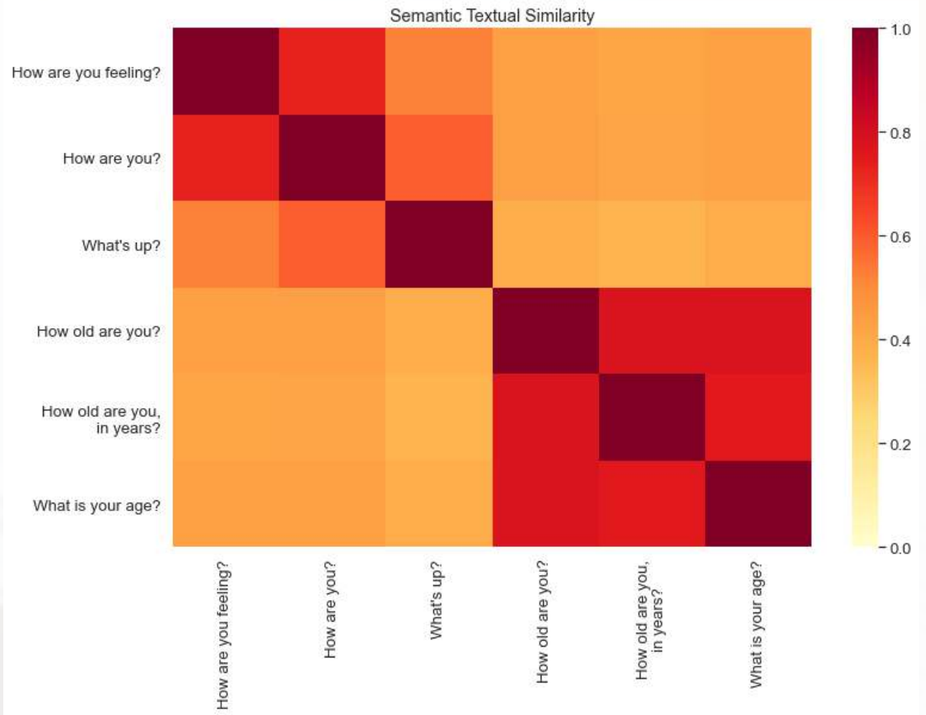
Example of USE in daily life

> Recall that USE focuses on representing sentence-length chunks of text



Demo: Comparing sentence meaning with USE

- Code is available at: https://rmc.link/colab_use
  - This is a hosted environment provided by Google



An approach for slightly longer text: BERT

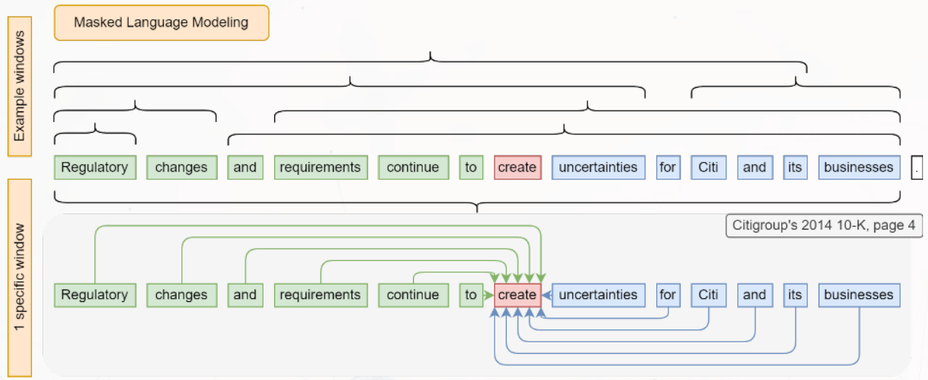
Bidirectional **E**ncoder **R**epresentations from **T**ransformers

- BERT handles text up to 512 *tokens*
  - A good fit for sentences or up to small paragraphs
- It comes pretrained on Wikipedia and a corpus of books
- Many other variants trained on other text exist
  - FinBERT on 10-K and 10-Q filings, Earnings calls, and analyst reports
- Surprisingly easy to use, thanks to platforms like HuggingFace 🤗

> ⓘ **Why use BERT over USE?**
>
> USE is just an embedding. BERT provides the ability to fine-tune the embeddings to map to your own classification through a supervised step.

How are models like this trained?

1. Masked language modeling: Similar to CBOW, but using the whole sentence
2. Next sentence prediction: Feed it pairs of sentences, half correct and half incorrect
3. Do both! BERT is a 50/50 mix



Next Sentence Prediction

- Correct: “Regulatory changes and requirements continue to create uncertainties for Citi and its businesses.” ⇒ “While the U.S. economy continues to improve, it remains susceptible to global events and volatility.”
- Incorrect: “Regulatory changes and requirements continue to create uncertainties for Citi and its businesses.” ⇒ “Citigroup expenses increased 14% versus 2013 to $55.1 billion.”

What can we do with these embeddings?

1. Direct measurement
    - Similarity of vectors $\Rightarrow$ similarity of meaning
        - See Crowley, Huang and Lu 2024 for an example
2. Include the embeddings in a regression or penalized regression
    - See Brown, Crowley and Elliott (2020) for an example applied to LDA
3. Cluster within the embedding space to determine groupings of meaning
    - Done in Crowley and Wong (2023) using kmeans + USE
    - Also done by BERTopic

Clustering in embedding spaces

- Clustering provides us an interpretable output from our embedding without needing ex-ante classes to be specified

> Example: Crowley and Wong (2023)

- Methods
  - Embeddings: USE applied to 40M+ clauses from 10-K MD&A
  - Cluster using kmeans
  - Supervise using Gap statistic (Tibshirani et al. 2001)
- Use case:
  - Classifying clauses within documents in an automated fashion
  - Applying sentiment classification at the class level instead of document level

Clustering to contexts

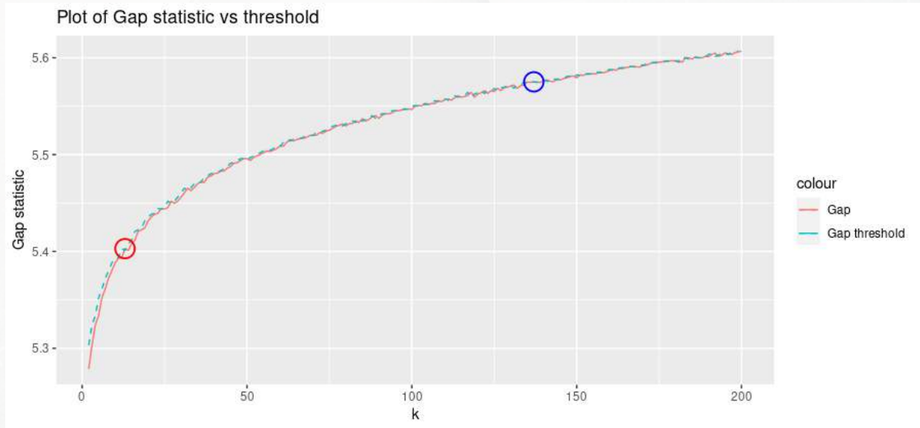
- We cluster within the 512-dim vector space with *Mini-Batch K-means* (Sculley 2010)
  - Mini-Batch K-means is an *online* version of K-means
    - Output is the same as K-means, but the process is more memory-friendly

Optimizing with Gap statistic

- Gap statistic (Tibshirani et al. 2001) is a simulation approach to supervising clustering
- Goal: Select the lowest k by comparing the informativeness of clustering on real data vs. synthetic data
  - Compare informativeness at k vs. at $k+1$, look for a gap < 1 S.D.
  - Caveat: Optimal k may be too small in more varied text; thus we compare k to $k+1$ and $k+2$

- 13 is the lowest k vs $k+1$ (red circle)
- 137 is the lowest k vs $k+1$ and $k+2$ (blue circle)

> 137 contexts in the data



How does the Gap statistic work?

- Let…
  - k be the number of clusters,
  - B the number of simulated samples
  - W_k be the K-Means inertia score on actual data
  - $W^*_{k,r}$ be the K-Means inertia score for iteration r with synthetic data
  - $\bar{l}$ be the average of the $W^*_{k,r}$s

$$Gap(k) = \left(\frac{1}{B}\right)\sum_{r=1}^{B}\log\left(W^*_{k,r}\right) - \log\left(W_k\right) \text{ and}$$

$$s_k = sd_k\sqrt{1+\frac{1}{B}}, \text{ where } sd_k = \sqrt{\left(\frac{1}{B}\right)\sum_{r=1}^{B}\left\{\log\left(W^*_{k,r} - \bar{l}\right)\right\}^2}$$

- Select the lowest k such that $Gap(k) \geq Gap(k+1) - s_{k+1}$

> I.e., select the lowest k s.t. the log-scaled error removed by clustering on real data at k is no worse than 1 SD below the log-scaled error removed at $k+1$

Examples of contexts

Accounting

- *Policies*: Assumptions, Revenue Recognition, Tax, Cautionary Statements
- *Standards*: Standards, New standards
- *General or B/S*: Cash flow, Deferred tax
- *Income statement discussion*: Accounting losses, Depreciation and amortization

Business operations

- *Debt, Equity, and Investment*: Financing, Loans
- *Expectations and future*: Management expectations, Risk factor disclosures
- *Macroeconomics*: Interest rates, Market risk
- *Operations*: Growth, Customers, Products
- *Structure*: Subsidiaries, Partnerships

Changes

- Changes in: sales, expenses, operating measures
- Declines in: value or performance
- Increase in: expenses, income or revenue

Ungrouped

- *Grammatical patterns*
- *Timeframes*
- *Unrelated statements*
- *Unrelated statements with specific words*

What clauses are in the contexts?

Accounting assumptions

1. “Option pricing models require input of highly subjective assumptions particularly for expected stock price volatility”
2. “Weighted average assumptions determine net periodic pension benefit expense”

Growth

1. “Growth was partially offset by closure”
2. “Diamond ’s capital expenditure budget is Diamond ’s highest at approximately $ 250 million with much related to internal growth activities comprised of expansions of facilities”

Deferred tax

1. “Adtalem recognizes future tax benefits associated with tax loss as deferred tax assets”
2. “Company fully impaired deferred tax asset resulting in 5 % effective tax benefit rate”

Market risk

1. “We are exposed to market risk related to interest rate risk on investment of cash in securities with original maturities”
2. “Currency gains related to market risk”

GPT models

What is a GPT model?

> A GPT model is a type of *Large Language Model* (LLM)

- Large: many parameters in the model (usually >1 billion)
- Language: the models are trained by seeing a large amount of written text
  - They infer everything from language
- Model: It's just an algorithm like everything else

> What does GPT mean? Generative Pre-trained Transformers

- Generative: It provides answers by generating an answer based on some latent space, as opposed to selecting answers it has previously seen
- Pre-trained: It's seen a lot of data already. That does not preclude it from seeing more.
- Transformer: A specific neural network architecture (which will talk about in Session 11)

What can ____-GPT do?

What can they do

- Classify data based on a small number of examples
  - “Few shot learning”
- Provide answers in flexible/trainable formats
- Encode and decode language
- Pattern matching
- Images as language

What can they not do

- Unless you train it yourself, it won’t have much domain-specific knowledge
- Beat single-purpose SOTA algorithms on most tasks
  - Validation is always needed to compare performance

How do different GPT models vary?

> Context length

- GPT-2: 2,048 tokens
- GPT-3: 4,096 tokens
- GPT-3.5: 4,096
- Chat-GPT: 4,096 tokens
- GPT-4: 8,096 or 32,384 tokens

Demo: Let’s build one!

> Go to: rmc.link/colab_gpt

- A simple one
  - 12,656 parameters
  - 2 possible tokens
  - A context length of 3
- As a comparison, GPT-2 has:
  - 1.5 billion parameters
  - 50,257 possible tokens
  - a context length of 2,048

How to interpret the network

The arrows show transition from a set of 3 characters to the next. In this process, the left-most character is dropped, the remaining two characters shift left, and a new character is added to the right side.

What to look for in the GPT Colab

1. We can see that it encodes simple patterns in the data well
2. We can see that answers are effectively probabilistic
3. We can see why hallucination occurs

> Additionally, things you can play around with:

1. How does adding more training iterations (epochs) change the output of the model?
2. How does the length of the input data (`seq` in the file) change the output of the model?

Fine tuning text classifiers

Fine-tuning

> Fine tuning allows us to convert embeddings into the classes we want to obtain

1. Hand annotate data with the classes you want
2. Add a couple more NN layers onto the already-trained BERT model
   - *Dropout* for regularization
   - *Dense* for interactions
   - *Output* to get probabilities for each class
3. Train this new model on your hand annotated data

Fine-tuning vs. zero-shot

Fine-tuning (BERT)

- +Tune once, apply repeatedly
- -Requires a GPU to train
  - +Trained model can run on CPU (slowly)
- -Needs many examples
  - +But less than older methods (SVM)
- -Short context length

Zero-shot (GPT)

- +Few examples needed for easy problems
- +Longer context length
- -Need to provide examples for each batch
- -Need to stay within context length
- -Hard problems may not be tractable
- -Many universities don’t have the hardware for this

> Both approaches yield something similar, just with different costs

Demo: Classification using fine-tuned FinBERT

- FinBERT is available via huggingface:
  - Via the `transformers` package
- A demo of the pretrained model is available via my colab shares:
  - rmc.link/colab_finbert
    - Supports tone, ESG, and forward looking statement classification
- Code to fine-tune FinBERT: Sample code available here
  - Note: Colab lacks the computational power needed for this task

Summary

Summary

- There are a lot of different methods available
  - Simple methods
  - Complex methods
  - Purely hand-coded methods
  - Purely automated methods
  - Anything in between
- Since the early 2010s accounting researchers have looked at a lot
  - The amount of data and possible measures dwarfs what we have examined to date

What should you do?

Use the method that best fits your problem. Some problems require complex solutions; others are well suited to simple approaches.

Thanks!

Dr. Richard M. Crowley
rcrowley@smu.edu.sg
@prof_rmc
rmc.link/
Slides @ rmc.link/ML2024

More resources

1. Dictionaries: ML for SS Session 4
2. Song and Wu (2008)/Schutze et al. (2008): ML for SS Session 4
3. SVM: ML for SS Session 2
4. King, Lam and Roberts (2017): My GitHub paper, Replication code
5. LDA: ML for SS Session 6
6. Clustering on embeddings: Wong and Crowley 2023 Slides
7. Transformers
    - USE: ML for SS Session 6
    - FinBERT: ML for SS Session 11
8. GPT: See Kim, Muhn, and Nikolaev (2024) Working for an interesting use case.

References

- Antweiler, Werner, and Murray Z. Frank. 2005. “Is All That Talk Just Noise? The Information Content of Internet Stock Message Boards.” The Journal of Finance 59 (3): 1259–94. https://doi.org/10.1111/j.1540-6261.2004.00662.x.
- Bao, Y., and Datta, A. 2014. “Simultaneously Discovering and Quantifying Risk Types from Textual Disclosures.” Management Science 60 (6): 1371–1391.
- Bonsall, Samuel B., Andrew J. Leone, Brian P. Miller, and Kristina Rennekamp. 2017. “A Plain English Measure of Financial Reporting Readability.” Journal of Accounting and Economics 63 (2): 329–57. https://doi.org/10.1016/j.jacceco.2017.03.002.
- Botosan, C. A. 1997. “Disclosure level and the cost of equity capital.” The Accounting Review 72 (3), 323–349.
- Brown, Nerissa C., Richard Crowley, and W. Brooke Elliott. 2020. “What Are You Saying? Using Topic to Detect Financial Misreporting.” Journal of Accounting Research.
- Cole, C. J. and C. L. Jones. “Management Discussion and Analysis: A Review and Implications for Future Research.” Journal of Accounting Literature 24, 135–174.
- Crowley, Richard, Wenli Huang, and Hai Lu. 2024. “Executive Tweets.” Working Paper
- Das, Sanjiv R., and Mike Y. Chen. 2007. “Yahoo! For Amazon: Sentiment Extraction from Small Talk on the Web.” Management Science 53 (9): 1375–88. https://doi.org/10.1287/mnsc.1070.0704.
- Dorrell, J. T., and N. S. Darsey. 1991. “An analysis of the readability and style of letters to stockholders.” Journal of Technical Writing and Communication 21: 73–83.

References

- Dyer, Travis, Mark Lang, and Lorien Stice-Lawrence. 2017. “The Evolution of 10-K Textual Disclosure: Evidence from Latent Dirichlet Allocation.” Journal of Accounting and Economics 64 (2): 221–45. https://doi.org/10.1016/j.jacceco.2017.07.002.
- Harris, Zellig S. “Distributional structure.” Word 10, no. 2-3 (1954): 146-162.
- Hassan, T. A., Hollander, S., LENT, L. V., & Tahoun, A. (2024). The global impact of Brexit uncertainty. The Journal of Finance, 79(1), 413-458.
- Hope, Ole-Kristian, Danqi Hu, and Hai Lu. “The benefits of specific risk-factor disclosures.” Review of Accounting Studies 21 (2016): 1005-1045.
- Huang, Allen H., Reuven Lehavy, Amy Y. Zang, and Rong Zheng. “Analyst information discovery and interpretation roles: A topic modeling approach.” Management science 64, no. 6 (2018): 2833-2855.
- Huang, Allen H., Hui Wang, and Yi Yang. “FinBERT: A large language model for extracting information from financial text.” Contemporary Accounting Research 40, no. 2 (2023): 806-841.
- Jones, M. J. and P. A. Shoemaker. “Accounting Narratives: A Review of Empirical Studies of Content and Readability.” Journal of Accounting Literature 13, 142.
- Li, Feng. 2008. “Annual Report Readability, Current Earnings, and Earnings Persistence.” Journal of Accounting and Economics, Economic Consequences of Alternative Accounting Standards and Regulation, 45 (2): 221–47. https://doi.org/10.1016/j.jacceco.2008.02.003.

References

- Li, Feng. 2010. “The Information Content of Forward-Looking Statements in Corporate Filings—A Naïve Bayesian Machine Learning Approach.” Journal of Accounting Research 48 (5): 1049–1102. https://doi.org/10.1111/j.1475-679X.2010.00382.x.
- Loughran, Tim, and Bill McDonald. 2011. “When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks.” The Journal of Finance 66 (1): 35–65. https://doi.org/10.1111/j.1540-6261.2010.01625.x.
- Loughran, Tim, and Bill McDonald. 2014. “Measuring Readability in Financial Disclosures.” The Journal of Finance 69 (4): 1643–71. https://doi.org/10.1111/jofi.12162.
- Loughran, Tim, and Bill McDonald. 2016. “Textual Analysis in Accounting and Finance: A Survey.” Journal of Accounting Research 54 (4): 1187–1230. https://doi.org/10.1111/1475-679X.12123.
- Tibshirani, Robert, Guenther Walther, and Trevor Hastie. “Estimating the number of clusters in a data set via the gap statistic.” Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63, no. 2 (2001): 411-423.

Packages used for these slides

- `kableExtra`
- `knitr`
- `quarto`
  - `material-icons`
- `revealjs`
- `reticulate`

Appendix: Workflow

High level overview: Tools

Hardware

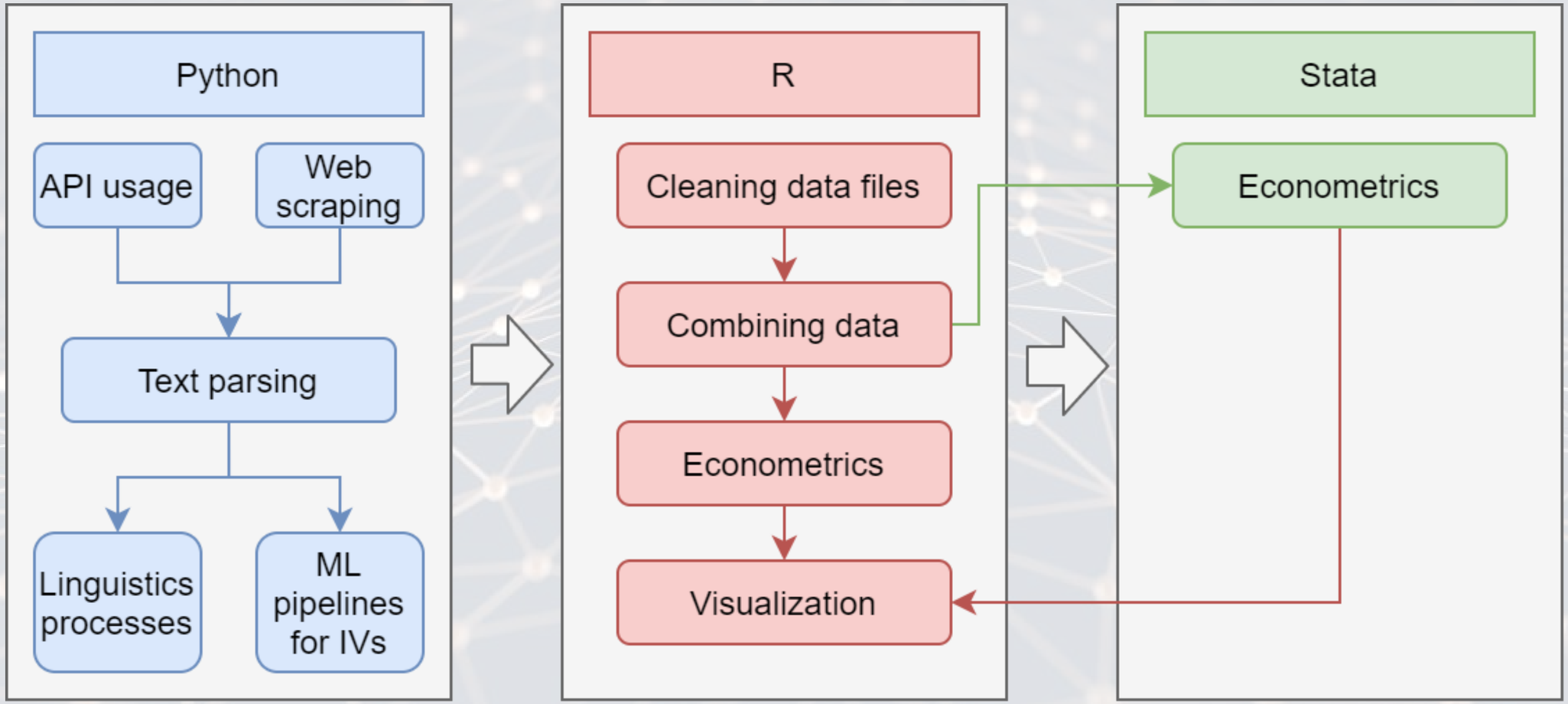
- Data stored in RAID 1/5/6 arrays (redundant disks)
  - Duplicated across machines
  - HDDs for large data, SSDs for temporary storage
- The more CPU cores, the better (so long as memory scales to match)
- Large memory for text analytics
  - 64-128GB is good for most tasks
  - 512GB for large matrix problems
- **Nvidia** GPU for training and inference
  - 10-100x speedup for most algorithms
  - Nvidia is needed for CUDA
  - CUDA is needed for most ML libraries
  - The more VRAM, the better

Software

- Python via miniconda for data collection and processing
  - Pycharm and JupyterLab for GUIs
- R for data manipulation, econometrics, and visualization
  - RStudio for GUI
- Stata for econometrics
- sftp for data transfer
- Nomachine for remote access

I run everything under Linux – a bit more stable for long computations, better multithreading in python, and easier to set up servers on

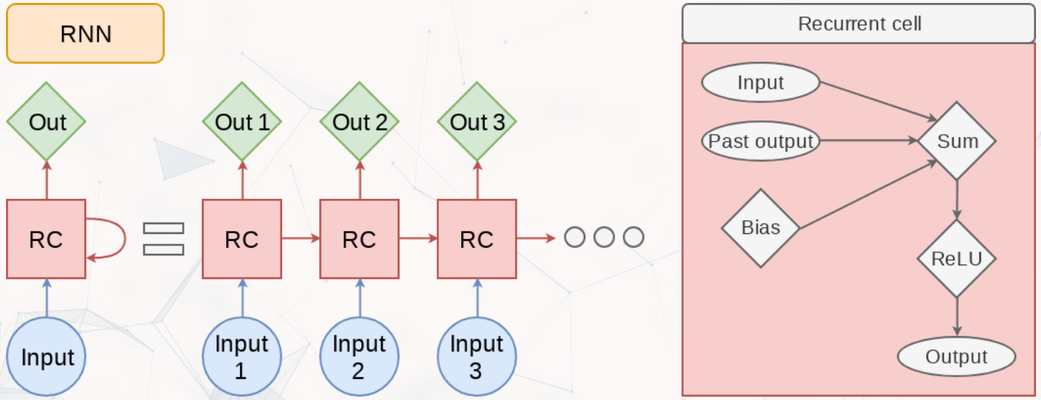
My workflow


Python
API usage
Web scraping
Text parsing
Linguistics processes
ML pipelines for IVs
R
Cleaning data files
Combining data
Econometrics
Visualization
Stata
Econometrics

Appendix: Neural Networks

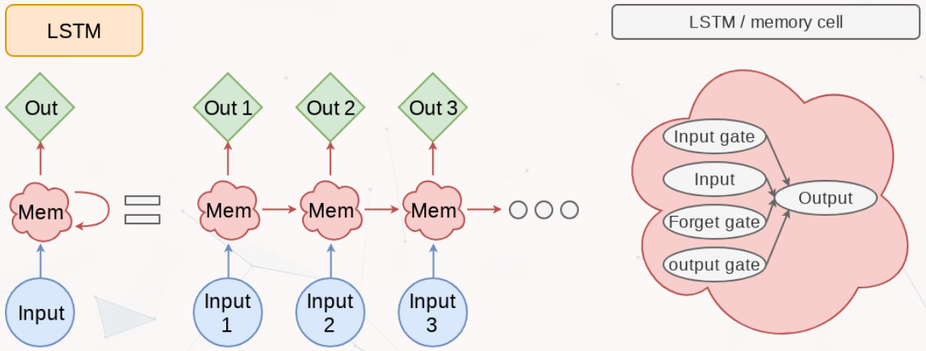
NN Methods: RNN ,Recurret Neural Network

- Recurrent neural networks embed a history of information in the network
  - The previous computation affects the next one
  - Leads to a *short term memory*
- Used for speech recognition, image captioning, anomaly detection, and many others
  - Also the foundation of LSTM
  - SketchRNN (live demo)



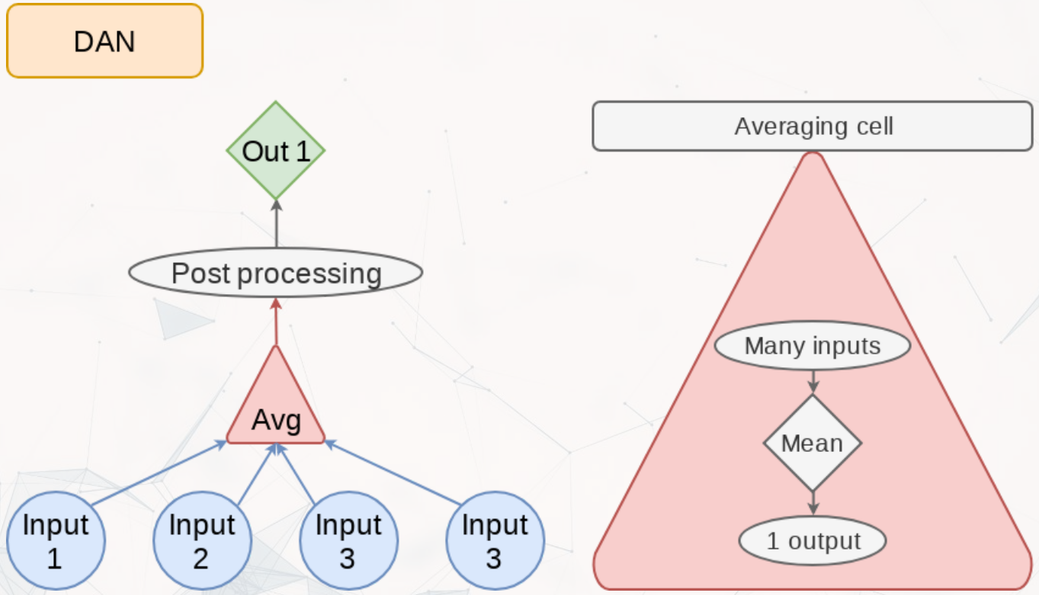
NN Methods: LSTM, Long Short Term Memory

- LSTM improves the *long term memory* of the network while explicitly modeling a *short term memory*
- Used wherever RNNs are used, and then some
  - Ex.: Seq2seq (machine translation)



NN Methods: DAN, Deep Averaging Network

- DANs are simple networks that simply average their inputs
- Averaged inputs are then processed a few times
- These networks have found a home in NLP
  - Ex.: Universal Sentence Encoder



NN Methods: Transformer

- Shares some similarities with RNN and LSTM: Focuses on attention
- Currently being applied to solve many types of problems
- Examples: BERT, GPT-3, XLNEt, RoBERTa, ChatGPT