# Module 5: Language Models & NLP in Research

- Using Large Language Models (LLMs) to summarize scientific literature
- Tools: ChatGPT, SciSummary, PaperQA
- Applications: literature reviews, data extraction from reports
- 📌 Activity: Try summarizing a scientific paper using an LLM

**IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System**
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 "Education and Research" - Component 2: "From research to business" - Investment 3.1: "Fund for the realisation of an integrated system of research and innovation infrastructures"

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

# Why Summarize Scientific Papers with AI?

- Explosion of publications: thousands of new papers daily
- Manual reading ≠ scalable for literature reviews
- LLMs help extract, condense, and synthesize key findings quickly

# What Are Large Language Models (LLMs)?

- Definition: Trained on massive text corpora to understand and generate natural language
- Examples: ChatGPT, Claude, Gemini
- Strengths: abstraction, summarization, question answering
- Limitations: hallucination, outdated knowledge

# Use Cases in Scientific Research

- Rapid literature reviews
- Extracting methods/results from PDFs
- Building databases from unstructured text
- Assisting grant writing or project scoping

# Tools for Summarizing Papers

- ChatGPT: General-purpose, conversational
- SciSummary: Tailored for academic abstracts & summaries
- PaperQA: Ask targeted questions from a paper (retrieval-augmented)
- Bonus: Semantic Scholar, ScholarAI plugins, Scite Assistant

# Comparison – LLM Summarization Styles

| Tool | Summary Type | Strengths | Notes |
|---|---|---|---|
| ChatGPT | Conversational summary | Versatile, dialog-driven | Customizable prompts |
| SciSummary | Formal abstract-style | Fast, academic tone | Limited interactivity |
| PaperQA | Q&A from PDF chunks | Deep focus, source-linked | Needs good questions |

# Choose a short academic article or abstract (1–2 pages)

- Use ChatGPT or SciSummary to:
  - Generate a plain-language summary
  - Extract the main research question, method, and conclusion
- Discuss: Did the summary match your interpretation?

# Prompt Engineering Tips

- Be specific: “Summarize the methodology section in bullet points.”
- Use roles: “You are a science journalist. Summarize this paper for a lay audience.”
- Ask for structured output: TL;DR, pros/cons, RQ + results

# Automating Literature Reviews

- Build pipelines:
  - Search → scrape → summarize → tag
  - Zotero + API tools + LLM = semi-automated workflows
- Organize summaries into structured formats (e.g., CSV, Notion, Roam)

# Limitations and Ethical Use

- LLMs may miss nuance or misinterpret data
- Don’t rely on summaries for critical decisions without reading the original
- Citing AI output? Follow institutional and journal policies

# Integration with Other Research Tools

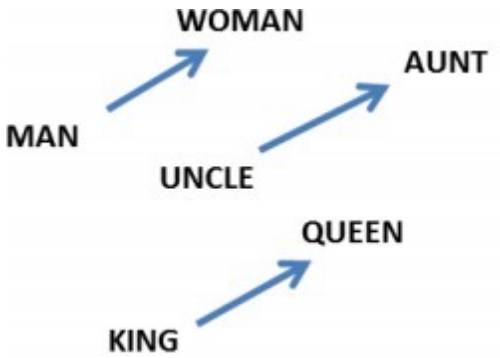
- Reference managers: Zotero, EndNote, Mendeley
- Markdown + GPT for research journaling
- Embedding in Jupyter notebooks for interactive workflows

# Example Outputs

- Show original abstract + ChatGPT summary
- Compare with SciSummary version
- Highlight differences in tone, clarity, and detail

# Word embeddings: properties

- Relationships between words correspond to difference between vectors.




$$W(\text{“woman”}) - W(\text{“man”}) \simeq W(\text{“aunt”}) - W(\text{“uncle”})$$

$$W(\text{“woman”}) - W(\text{“man”}) \simeq W(\text{“queen”}) - W(\text{“king”})$$

http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

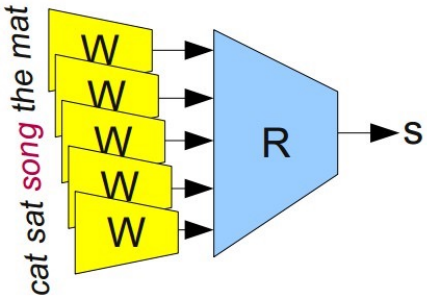
# Word embeddings: questions

- How big should the embedding space be?
  - Trade-offs like any other machine learning problem – greater capacity versus efficiency and overfitting.
- How do we find W?
  - Often as part of a prediction or classification task involving neighboring words.

# Learning word embeddings

- First attempt:
  - Input data is sets of 5 words from a meaningful sentence. E.g., “one of the best places”. Modify half of them by replacing middle word with a random word.
    “one of function best places”
  - W is a map (depending on parameters, Q) from words to 50 dim’l vectors.
    E.g., a look-up table or an RNN.
  - Feed 5 embeddings into a module R to determine ‘valid’ or ‘invalid’
  - Optimize over Q to predict better

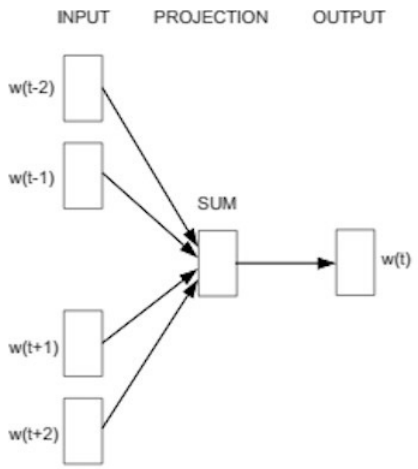
https://arxiv.org/ftp/arxiv/papers/1102/1102.1808.pdf



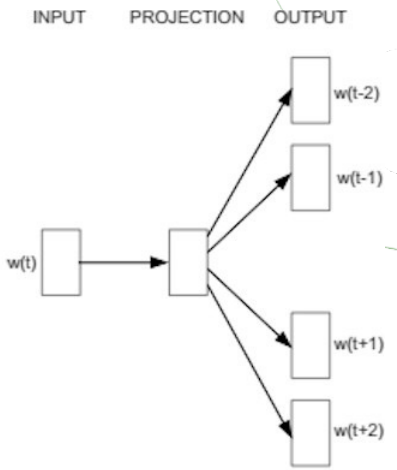

http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

# word2vec

- Predict words using context
- Two versions: CBOW (continuous bag of words) and Skip-gram




CBOW

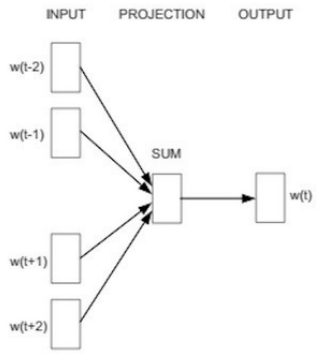



Skip-gram

https://skymind.ai/wiki/word2vec

# CBOW

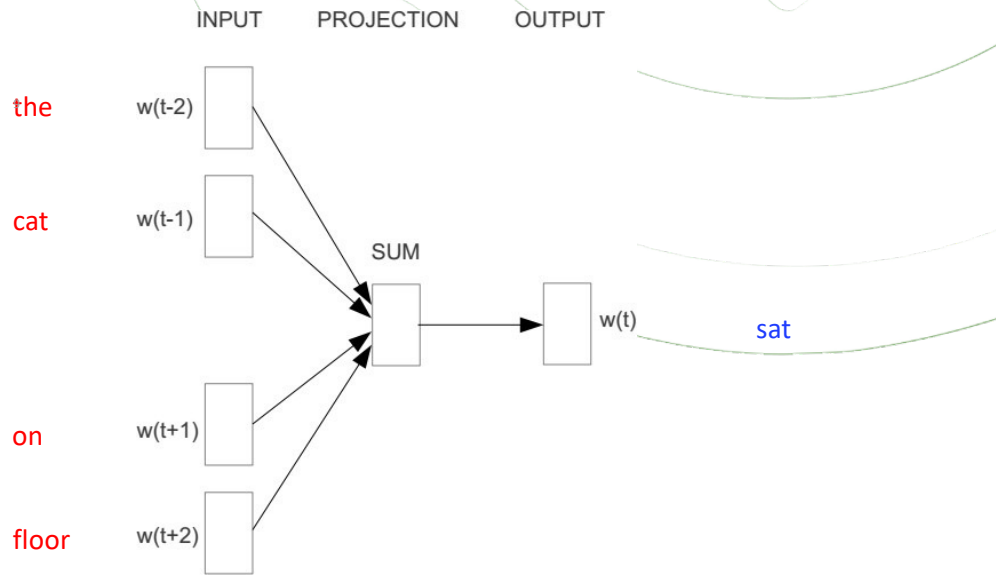
- Bag of words
  - Gets rid of word order. Used in discrete case using counts of words that appear.
- CBOW
  - Takes vector embeddings of n words before target and n words after and adds them (as vectors).
  - Also removes word order, but the vector sum is meaningful enough to deduce missing word.




CBOW

# Word2vec – Continuous Bag of Word

- E.g. “The cat sat on floor”
  - Window size = 2




www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

# Word Analogies

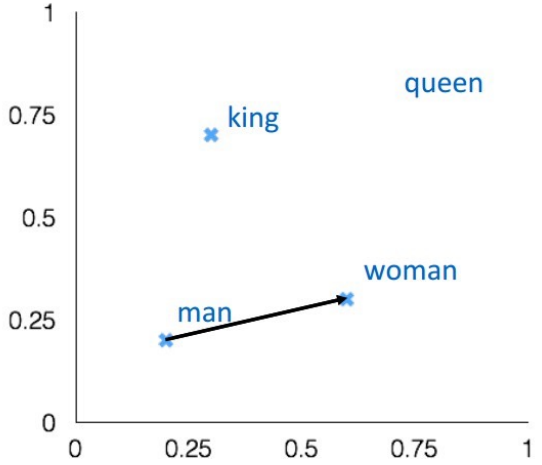
Test for linear relationships, examined by Mikolov et al. (2014)

a:b :: c:?

$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

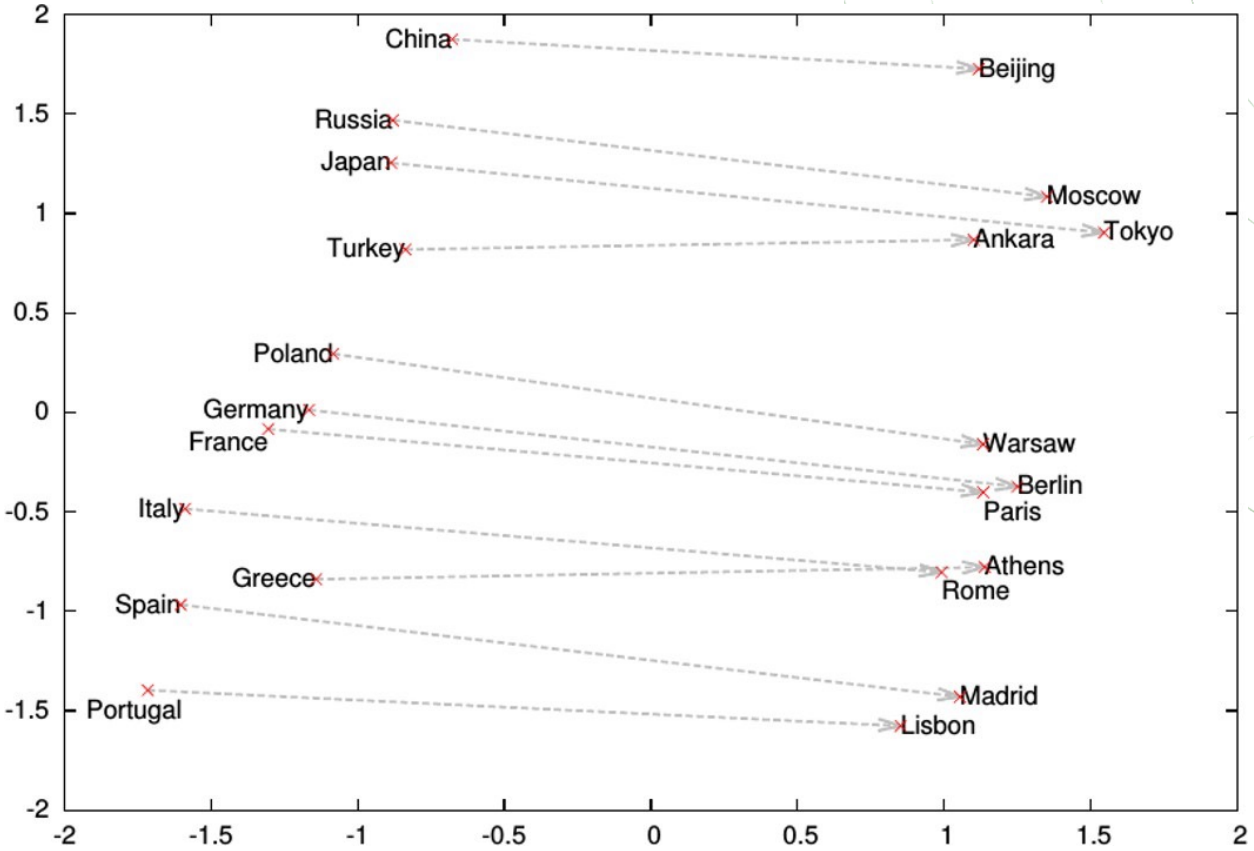
man:woman :: king:?

| | | |
|---|---|---|
| + | king | [ 0.30 0.70 ] |
| - | man | [ 0.20 0.20 ] |
| + | woman | [ 0.60 0.30 ] |
| | queen | [ 0.70 0.80 ] |




www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

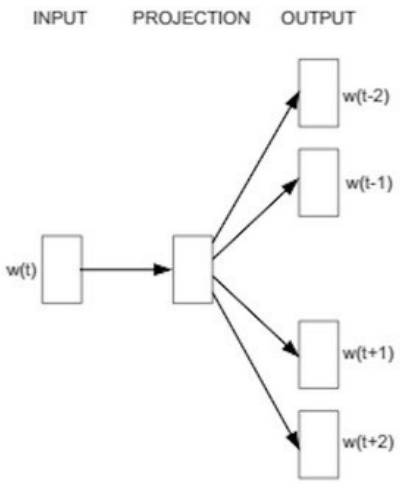
# Word analogies




www.cs.ucr.edu/~vagelis/classes/CS242/slides/word2vec.pptx

# Skip gram

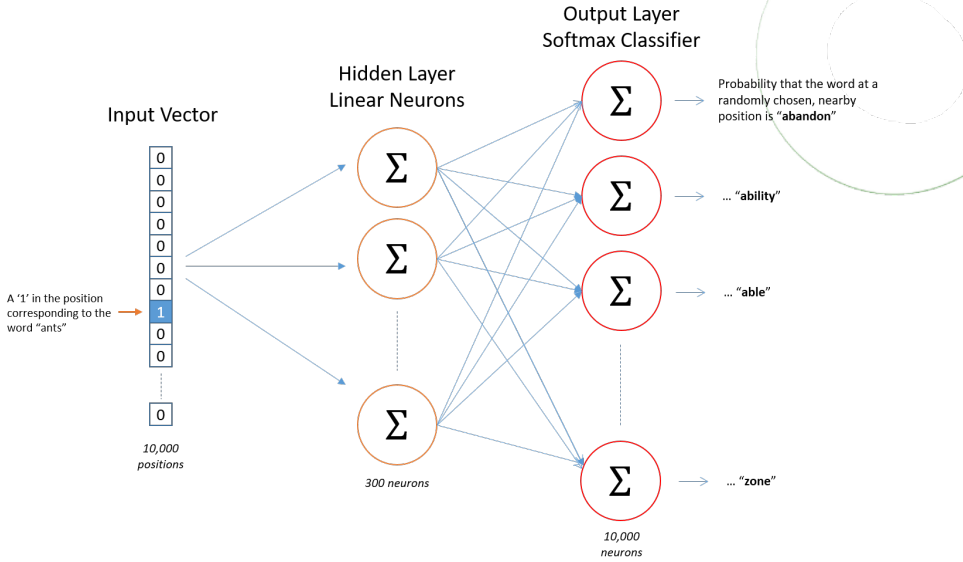
- Skip gram – alternative to CBOW
  - Start with a single word embedding and try to predict the surrounding words.
  - Much less well-defined problem, but works better in practice (scales better).




Skip-gram

# Skip gram

- Map from centre word to probability on surrounding words. One input/output unit below.
- There is no activation function on the hidden layer neurons, but the output neurons use softmax.

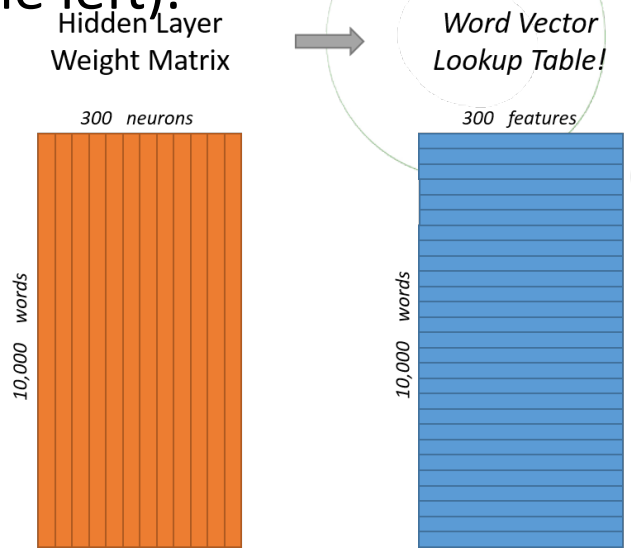



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

# Skip gram example

- Vocabulary of 10,000 words.
- Embedding vectors with 300 features.
- So the hidden layer is going to be represented by a weight matrix with 10,000 rows (multiply by vector on the left).

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$




http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

# Skip gram/CBOW intuition

- Similar “contexts” (that is, what words are likely to appear around them), lead to similar embeddings for two words.
- One way for the network to output similar context predictions for these two words is if the word vectors are similar. So, if two words have similar contexts, then the network is motivated to learn similar word vectors for these two words!

http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

# word2vec

- word2vec, or skip-gram, is an algorithm for training real-valued vectors to represent each word.
- If word $w1$ is represented by vector $\vec{v}1 = v11, \ldots, v1D$, we say that $\vec{v}1$ is the D-dimensional embedding of word $w1$.
- The general area of vector semantics (representthe meaning of a word as a vector) goes back to the 1950s, in the field of information retrieval (more about that in the next lecture).
- word2vec is an algorithm for learning those vectors using a one-layer neural network, in such a way that similar words are close together in the vector space.

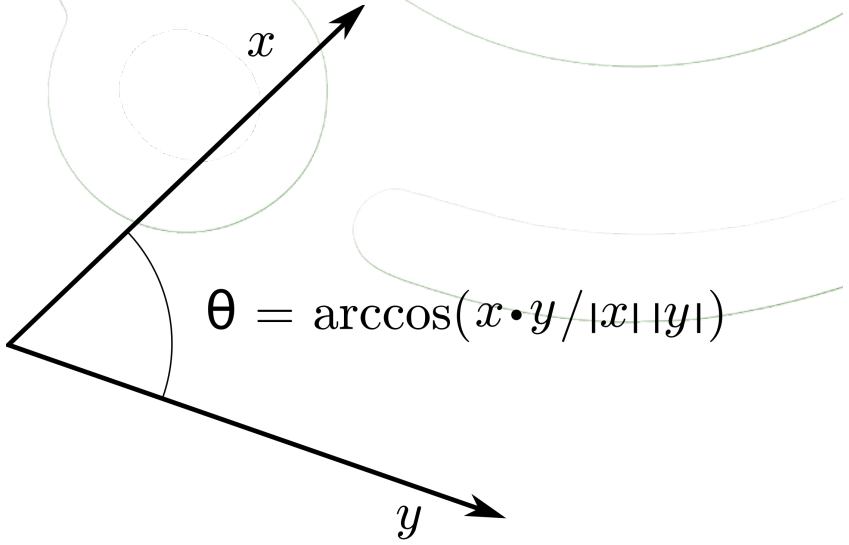
# cosine similarity

- If words $w_1$ and $w_2$ are similar, $w_1$is represented by vector $\vec{v}_1$, and $w_2$ by vector $\vec{v}_2$, then the angle between the two vectors should be small.
- Angle between two vectors can be measured by their dot product

where

$$\cos\theta = \frac{\vec{v}_1 \,)\, \vec{v}_2}{|\vec{v}_1||\vec{v}_2|}$$

$$\vec{v}_1 \,)\, \vec{v}_2 = \sum_{d=1}^{D} v_{1d} v_{2d}\,, \qquad |\vec{v}_1| = \sqrt{\sum_{d=1}^{D} v_{1d}^2}$$




By BenFrantzDale at the English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=49972362

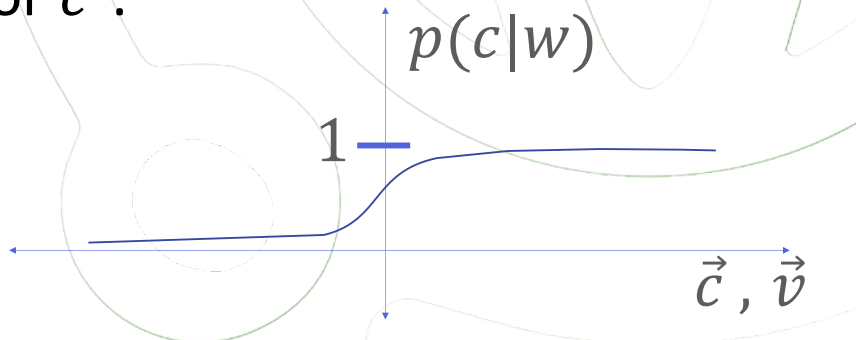
# Word2vec: context probability

- The key innovation of word2vec is the idea of representing similarity as the probability that words $w_1$ and $w_2$ could occur in the same context, and of estimating the probability using a sigmoid.
- Consider the "…hot although iced **coffee** is a popular…". Define the target word to be $w$ =coffee.
- Define the context words $c_{-3}$ =hot, $c_{-2}$ =although, …, $c_3$ =popular. Use a naïve Bayes model of the context probability:

$$p(c_{-3}, \dots, c_3 | w) = \prod_{\substack{i=-3 \\ i \neq 0}} p(c_i | w)$$

# word2vec: context probability

- Now suppose we want to embed $w$ =coffee with a vector $\vec{v}$.
- …and we want to embed $c$–3 =hot with a vector $\vec{c}$.
- Define the probability that “hot” occurs within
- +/-N words of “coffee” to be just a sigmoid:




$$p(c|w) = \frac{1}{1 + e^{-c \cdot \vec{v}}}$$

# word2vec: training

- We train the neural network by listing, as positive examples, the words that occur in the context of "$w$ =coffee," e.g.,
  - $\mathcal{D}+(w) = \{$hot, although, iced, moderate, the, hot, consumption, ... $\}$
- Create a negative database by selecting words at random from the vocabulary, each word in proportion to its frequency in the whole dataset:
  - $\mathcal{D}-(w) = \{$aardvark, dog, gazebo, the, precipitates, ... $\}$

# word2vec: training

- The coefficients $\vec{v}i = vi1, \ldots, viD$ for each vector are then learned in order to maximize the log probability of the dataset:

$$\mathcal{L} = \ln p(\text{Data}) = \sum_{w \in \mathcal{V}} \ln p(\mathcal{D}_+(w)|w) + \sum_{w \in \mathcal{V}} \ln p(\mathcal{D}_-(w)|w)$$

$$= \sum_{w \in \mathcal{V}} \sum_{c \in \mathcal{D}_+(w)} \ln p(c|w) + \sum_{w \in \mathcal{V}} \sum_{c \in \mathcal{D}_-(w)} \ln\left(1 - p(c|w)\right)$$

$$= \sum_{\vec{v} \in \mathcal{V}} \sum_{c \in \mathcal{D}_+(w)} \ln \frac{1}{1 + e^{-c \cdot \vec{v}}} + \sum_{\vec{v} \in \mathcal{V}} \sum_{c \in \mathcal{D}_-(w)} \ln\left(1 - \frac{1}{1 + e^{-c \cdot \vec{v}}}\right)$$

$$\mathcal{L} = \sum_{\vec{v} \in \mathcal{V}} \sum_{c \in \mathcal{D}_+(w)} \ln \frac{1}{1 + e^{-c \cdot \vec{v}}} + \sum_{\vec{v} \in \mathcal{V}} \sum_{c \in \mathcal{D}_-(w)} \ln \frac{1}{1 + e^{c \cdot \vec{v}}}$$

# word2vec: training

The coefficients $\vec{v}_i = [v_{i1}, \ldots, v_{iD}]$ for each vector are then learned in order to maximize the log probability of the dataset:

$$v_{id} \leftarrow v_{id} + \eta \frac{d\mathcal{L}}{dv_{id}}$$

$$= v_{id} + \eta \frac{d}{dv_{id}} \left( + \sum_{\vec{v} \in \mathcal{V}} \sum_{\vec{c} \in \mathcal{D}_+(\mathrm{w})} \ln \frac{1}{1 + e^{-\vec{c}/\vec{v}}} + + \sum_{\vec{v} \in \mathcal{V}} \sum_{\vec{c} \in \mathcal{D}_-(\mathrm{w})} \ln \frac{1}{1 + e^{\vec{c}/\vec{v}}} \right)$$

There's one more issue to consider here: if the word coffee occurs as a center word (w=coffee) or a context word (c=coffee), should those vectors ($\vec{v}$ and $\vec{c}$, respectively) be the same vector, or different vectors? The results are slightly different; which one is better depends on the application for which you're training word2vec.

# Word2vec shortcomings

- Problem: 10,000 words and 300 dim embedding gives a large parameter space to learn. And 10K words is minimal for real applications.
- Slow to train, and need lots of data, particularly to learn uncommon words.

# Word2vec improvements: word pairs and phrases

- Idea: Treat common word pairs or phrases as single “words.”
  - E.g., Boston Globe (newspaper) is different from Boston and Globe separately. Embed Boston Globe as a single word/phrase.
- Method: make phrases out of words which occur together often relative to the number of individual occurrences. Prefer phrases made of infrequent words in order to avoid making phrases out of common words like “and the” or “this is”.
- Pros/cons: Increases vocabulary size but decreases training expense.
- Results: Led to 3 million “words” trained on 100 billion words from a Google News dataset.

# Word2vec improvements: subsample frequent words

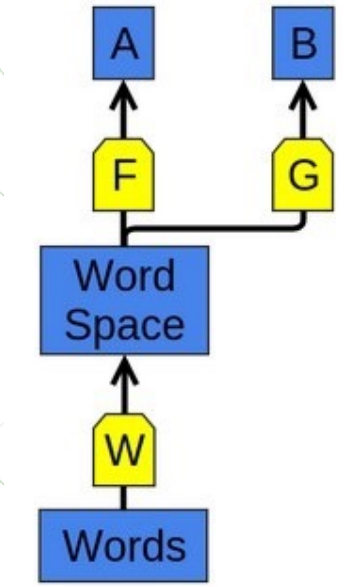
- Idea: Subsample frequent words to decrease the number of training examples.
  - The probability that we cut the word is related to the word's frequency. More common words are cut more.
  - Uncommon words (anything < 0.26% of total words) are kept
  - E.g., remove some occurrences of "the."
- Method: For each word, cut the word with probability related to the word's frequency.
- Benefits: If we have a window size of 10, and we remove a specific instance of "the" from our text:
  - As we train on the remaining words, "the" will not appear in any of their context windows.

# Word2vec improvements: selective updates

- Idea: Use “Negative Sampling”, which causes each training sample to update only a small percentage of the model’s weights.
- Observation: A “correct output” of the network is a one-hot vector. That is, one neuron should output a 1, and all of the other thousands of output neurons to output a 0.
- Method: With negative sampling, randomly select just a small number of “negative” words (let’s say 5) to update the weights for. (In this context, a “negative” word is one for which we want the network to output a 0 for). We will also still update the weights for our “positive” word.

# Word embedding applications

- The use of word representations… has become a key "secret sauce" for the success of many NLP systems in recent years, across tasks including named entity recognition, part-of-speech tagging, parsing, and semantic role labeling. (Luong et al. (2013))
- Learning a good representation on a task A and then using it on a task B is one of the major tricks in the Deep Learning toolbox.
  - Pretraining, transfer learning, and multi-task learning.
  - Can allow the representation to learn from more than one kind of data.

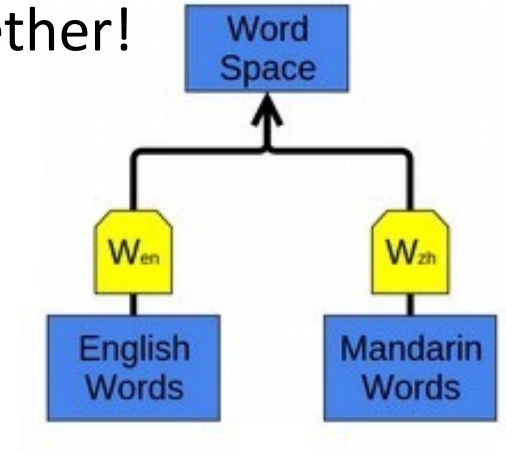



$W$ and $F$ learn to perform task A. Later, $G$ can learn to perform B based on $W$.

http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

# Word embedding applications

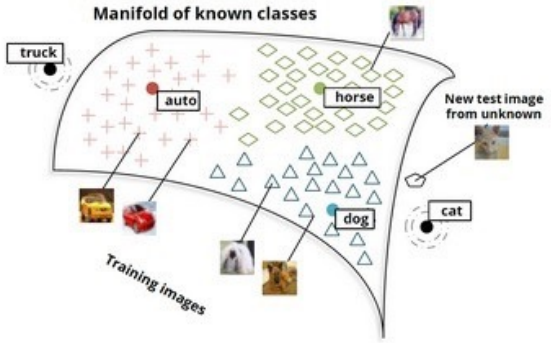
- Can learn to map multiple kinds of data into a single representation.
- E.g., bilingual English and Mandarin Chinese word-embedding as in Socher et al. (2013a).
- Embed as above, but words that are known as close translations should be close together.
- Words we didn't know were translations end up close together!
- Structures of two languages get pulled into alignment.



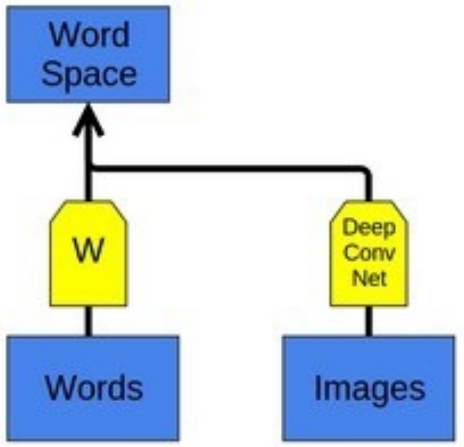

http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

# Word embedding applications

- Can apply to get a joint embedding of words and images or other multi-modal data sets.
- New classes map near similar existing classes: e.g., if ‘cat’ is unknown, cat images map near dog.




(Socher *et al.* (2013b))




http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

# Key Benefits Recap

- Save time scanning papers
- Improve comprehension for non-native readers
- Make science more accessible to collaborators and the public

# Resources & Tools

- Tools: scisummary.com, paperqa.xyz
- Plugins: ScholarAI (ChatGPT), Scite
- Datasets: arXiv, PubMed, Semantic Scholar Open Research Corpus

# Q&A and Feedback

- Prompt
  - “What are the pros and cons of using LLMs in your research field?”
- Invite discussion
- Optional: quick survey poll (Mentimeter or Slido)

# Final Activity: Case Studies & Group Work

- Group work: each team chooses a case (e.g., water monitoring, deforestation, urban heat islands)
- Define the problem, select relevant data and AI approach
- Share ideas with the class (5-min lightning pitch per group)

ITINERIS

# THANKS!


**Francesco Iarlori**

https://uk.linkedin.com/in/thefrankie/

@thefrankie

Francesco@Iarlori.com



**IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System**
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani

# 🐦 Teachable Machine: Bird Species Recognition

- Topic: Biodiversity Monitoring
- Tool: Google Teachable Machine
- Exercise:
  - Students upload or record audio clips (e.g. birdsong samples from Xeno-Canto or AudioSet).
  - Train a basic model to recognize different bird species.
  - Discuss performance and real-world limitations.
- Goal: Show the simplicity of model creation and limitations of generalization with small datasets.

# Google Earth Engine: Land Cover Classification

- Topic: Remote Sensing
- Tool: Google Earth Engine
- Exercise:
  - Use a public script (or provide one) that classifies land cover (forest, urban, water) in a chosen region.
  - Adjust parameters and visualize classification results.
  - Optional: Compare results from different years.
- Goal: Understand how satellite imagery and classification algorithms support environmental monitoring.

# Colab Notebook: Air Quality Anomaly Detection

- Topic: Big Environmental Data
- Tool: Google Colab
- Exercise:
  - Load a subset of OpenAQ data (PM2.5 or NO2 from Milan or another local city).
  - Clean the dataset using pandas.
  - Use scikit-learn to detect anomalies with Isolation Forest or Z-score method.
  - Visualize results with matplotlib.
- Goal: Learn how to process time-series data and detect anomalies using ML.

# 🔁 Time-Series Forecasting with RNNs (Opt. Adv.)

- Topic: Climate Modeling
- Tool: Colab + TensorFlow/Keras
- Exercise:
  - Load simplified climate data (e.g., monthly temperature anomalies).
  - Train an LSTM model to predict future values.
  - Evaluate MAE and plot prediction vs actual.
- Goal: Understand how deep learning can model temporal patterns in climate data.

# LLM Comparison: Literature Summarization

- Topic: LLMs in Research
- Tools: ChatGPT, Elicit, SciSummary
- Exercise:
  - Provide each group with a paragraph from a scientific abstract or paper (environment-related).
  - Ask them to summarize using ChatGPT or another LLM.
  - Compare the quality, tone, and usefulness of summaries across tools.
- Goal: Explore how LLMs can speed up literature review and knowledge extracv on.