Experiment 3: Embeddings

Prof. Dr. Nicolas Meseth

In this experiment you explore how a computer can represent the *meaning* of words as numbers. You start by placing words intuitively on paper with no rules at all, then assign explicit dimensions by hand, discovering how much structure can be captured in just three numbers, then investigate how real models discover their own dimensions automatically from text, and finally use those representations for semantic search in a way that no keyword search could match.

Step 1: Arranging Words by Hand

Before touching a computer, work with pen and paper.

You have the following 20 words: `cat`, `dog`, `Paris`, `Berlin`, `king`, `queen`, `happy`, `joyful`, `car`, `bicycle`, `eat`, `drink`, `fast`, `slow`, `summer`, `winter`, `man`, `woman`, `red`, `blue`.

1. Arrange the 20 words in a 2D grid on paper, or on a flip chart, or on your desk in class. Place words that feel *similar in meaning* close together and words that feel *different* far apart. There is no right answer, use your own intuition. Sketch the result or take a photo.

2. On what basis did you group them? By meaning, by topic, by grammar, or something else? Write down a short explanation for at least three of your groupings.

3. Compare your arrangement with a classmate. Where do you agree? Where do you disagree? Pick one difference and discuss: whose arrangement is more "correct", and what would it even mean for a word arrangement to be correct?

Step 2: Defined Dimensions

In Step 1 the axes on your paper had no labels. You placed words purely by intuition. Now assign explicit meaning to the axes and see how far that gets you.

4. Start with these four words: `man`, `woman`, `boy`, `girl`.

Draw a 2D coordinate system on paper with these two axes:

- x-axis: **age** (0 = very young, 10 = very old)
- y-axis: **gender** (0 = male, 10 = female)

Place the four words in the grid. Which pairs share the same age value? Which pairs share the same gender value? Draw an arrow from `man -> woman` and a second arrow from `boy -> girl`. What do you notice about the two arrows, their direction, length, and orientation relative to each other?

5. Add these four words to the same grid: `grandfather`, `adult`, `child`, `infant`. The words `adult` and `child` describe a person without specifying gender. Where do you place them on the gender axis? Is there a meaningful value, or is the axis simply not applicable to them? Write down your reasoning.

6. Add `grandmother`, `grandparent`, `teenager`, `octogenarian`, a person in their eighties. Place all four in the grid. The words `grandparent`, `teenager`, and `octogenarian` are gender-neutral, where did you place them? Now draw arrows for `grandfather -> grandmother` and `man -> woman`. Are the arrows parallel? What would it mean geometrically if they are?

7. Your 2D grid now contains 12 words. Write down the `(age, gender)` coordinates you chose for each word and fill in the following table.

Word	Age (0-10)	Gender (0-10)
man		
woman		
boy		
girl		
grandfather		
adult		
child		
infant		

Word	Age (0-10)	Gender (0-10)
grandmother		
grandparent		
teenager		
octogenarian		

8. Representing words as vectors allows you to perform arithmetic operations on them. Try three vector calculations by hand using the 2D coordinates from the table above. For each operation, look up the values you wrote down, subtract and add the components, and identify the closest word in the table:

- `grandfather - man + woman = ?`
- `boy - man + woman = ?`
- `grandmother - woman + man = ?`

Show your working for at least one of them. What component does the subtraction remove in each case? What does the addition replace it with?

9. Two dimensions are not enough for every word. In your 2D grid, where would you place `king`? It seems similar to `man` in age and gender, but there is at least one important additional difference.

Add a third axis to your vocabulary:

- z-axis: **royalty** (0 = not royal, 10 = royal)

This third axis cannot be drawn in 2D, but you can add it as a third number. Now add `king`, `queen`, `prince`, `princess` to your vocabulary and extend the table with the royalty column. Fill in all 16 words in the table below.

Word	Age (0-10)	Gender (0-10)	Royalty (0-10)
man			
woman			
boy			
girl			
grandfather			
adult			
child			
infant			
grandmother			
grandparent			

Word	Age (0-10)	Gender (0-10)	Royalty (0-10)
teenager			
octogenarian			
king			
queen			
prince			
princess			

10. Look at the relationships between `man -> king`, `woman -> queen`, `boy -> prince`, and `girl -> princess`. In terms of the three dimensions, what do they have in common?

11. Think of each word as the 3D vector that you wrote down before. For each operation below, predict the result before doing any calculation. Think about which component the subtraction removes and what the addition replaces it with. Write down a prediction and your reasoning for each one:

- `king - man + woman = ?`
- `prince - boy + girl = ?`
- `king - man + boy = ?`

Step 3: Similarity and Arithmetic

Now that every word is a vector of three numbers, you can measure similarity mathematically. To use your computer for calculations, you first need to put the vectors into a digital format.

12. Create a text file called `my_embeddings.txt` in a Python project. Enter your ratings from Step 2, one word per line: the word first, then its three numbers for each dimension, all separated by spaces:

```
man 7 0 0
woman 7 10 0
king 7 0 10
...
```

Fill in all 16 words using your own ratings from the table. This is the same file format used by real embedding models such as GloVe. You will load a file structured this way again later, and you can reuse your code.

Now create a Python file called `embeddings.py`, implement a function to load the embeddings. Test your function by loading your ratings from that file. Print `words["man"]` and

`words["king"]` to verify the file was read correctly.

13. Before continuing with calculations, visualize how your three-dimensional embedding looks when projected onto a 2D plane. Using a method called PCA, Principal Component Analysis, you can compress the three dimensions down to two while preserving as much of the original structure as possible. Run the corresponding code in the provided Jupyter notebook to create a PCA plot of your manual embedding.

Compare this plot to your hand-drawn arrangement from Step 1. Do the royalty words, `king`, `queen`, `prince`, `princess`, form a cluster? Where do the gender-neutral words, `adult`, `child`, `infant`, `grandparent`, end up?

Euclidean Distance

The simplest measure of similarity is **Euclidean distance**, the straight-line distance between two points in space.

14. Implement a function called `euclidean_distance` to compute the distance between two vectors:

$$d(\mathbf{v_1}, \mathbf{v_2}) = \sqrt{\sum_{i=1}^{n} (v_{1i} - v_{2i})^2}$$

Use `zip()` to iterate through the two vectors at the same time.

15. Use the function to compute the Euclidean distance for selected word pairs and organize the results in a table.

Compare the distances for `man / woman`, `man / king`, and `king / queen`. Are any of the results surprising?

Cosine Similarity

Euclidean distance answers "how far apart are the two points?" Cosine similarity answers a different question: "what angle do the two vectors make?", regardless of how long they are.

To see why this distinction matters, consider `man = (7, 1, 1)` and `boy = (2, 1, 1)` using typical ratings from Step 2. Their Euclidean distance is 5. But they point in exactly the same direction in space: both have gender = 1 and royalty = 1, only age differs. If you were to scale

one vector up or down, the direction would stay the same. Cosine similarity captures this: it returns 1.0 for any two vectors pointing in the same direction, regardless of their length.

Now compare `man = (7, 1, 1)` and `king = (8, 1, 8)`. Their Euclidean distance is approximately 7, similar to `man / woman`. But their directions differ, because the royalty dimension pulls `king` away from the non-royal direction. Cosine similarity is noticeably less than 1.0 here, reflecting this directional difference.

16. Research cosine similarity using any source you like, for example Wikipedia, ChatGPT, a video, or a textbook. Write down, in your own words, in 3 to 5 sentences, what it measures. Your explanation should answer:

- What is being compared: lengths, directions, or something else?
- Why is the angle between two vectors a useful measure of similarity?

17. Cosine similarity returns a value between -1 and 1. Fill in a small prediction table *before* running any code.

18. Calculate the cosine similarity between $\vec{a} = (1, 0)$ and $\vec{b} = (0, 1)$ by hand using the formula:

$$\text{cosine similarity}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

Now try $\vec{a} = (1, 1)$ and $\vec{b} = (2, 2)$. What do you notice? What does cosine similarity ignore?

19. Using your ratings from Step 2, compute both Euclidean distance and cosine similarity by hand for the pairs `man / boy` and `man / king`.

Discuss your results with a classmate. Which measure do you think captures the similarity between these pairs better? Why?

20. Implement a function called `cosine_similarity`. Re-run the word pairs from Step 3 using cosine similarity. Do the rankings change compared to Euclidean distance? For which pairs does cosine give a noticeably different picture?

Vector Arithmetic

One of the most surprising discoveries about word embeddings is that you can do arithmetic on them and get meaningful results.

21. You may have noticed that all cosine similarities above are between 0 and 1. This happens because all your ratings are >= 0, every vector points into the *positive* region of space. Two vectors that both have only non-negative components can never point in opposite directions, so their cosine similarity can never be negative.

Real embedding vectors are not constrained this way. To see what negative cosine similarity looks like, **center** your embedding: subtract the mean vector from every word vector. The mean vector is the component-wise average across all 16 words.

After centering, words with above-average age get a positive age component and words with below-average age get a negative one, so `infant` and `octogenarian` will now point in roughly *opposite* directions.

Run the centering code in the Jupyter notebook and re-run the selected word pairs using cosine similarity on the centered vectors. Which pairs now have negative similarity? Does the sign make semantic sense?

22. Implement the following four functions in your script:

- `add_vectors(vec_a, vec_b)`, returns a new list where each element is the sum of the corresponding elements from the two input lists
- `subtract_vectors(vec_a, vec_b)`, same idea, but subtract
- `find_nearest(target, word_dict)`, searches through all words in `word_dict`, computes the Euclidean distance from each word's vector to `target`, and returns the word and distance with the smallest distance
- `analogy(a, b, c, word_dict)`, computes `a - b + c` using the vectors in `word_dict` and returns the nearest word, excluding `a`, `b`, and `c`, which is the model's best guess for the answer to the analogy "`a` is to `b` as `c` is to ?"

Test your `analogy` function with a simple case you can verify by hand. Does the result match what you calculated in Step 2?

23. Using your `analogy` function, run the classic analogy `king - man + woman = ?`. Did you get `queen`? If not, look at your ratings: what values would `king` and `queen` need for the result to land exactly on `queen`?

24. Try these two analogies and report what you get:

- `prince - boy + girl = ?`
- `grandfather - man + woman = ?`

Do the results match your prediction from Step 2? What does it tell you that you can navigate the embedding space by adding and subtracting dimensions?

25. Try one more analogy of your own using words from your list. Report what you tried and what result you got.

Step 4: How Embeddings Learn

Your three hand-crafted dimensions work beautifully for 16 carefully chosen words. But a real vocabulary contains 100,000 or more words, emotions, verbs, abstract concepts, place names, and more. No team of humans could design the right dimensions for all of them. How do real embedding models solve this?

26. Try to assign age, gender, and royalty values to each of the following words: `happy`, `fast`, `democracy`, `river`. For each word, write down the values you chose, or `"does not apply"`, and explain why. What type of meaning do all three dimensions fail to capture entirely?

Learning from Context

The solution real models use is the **distributional hypothesis**: *words that appear in similar contexts tend to have similar meanings.* A model does not need a human to define dimensions, it can discover structure by observing which words appear near each other across billions of sentences.

Consider this small training corpus:

```
the king ruled wisely
the queen ruled wisely
the prince became leader
the princess became leader
the man worked hard
the woman worked hard
```

27. For window size `C = 2`, the context of a word is every other word within two positions to its left or right in the same sentence.

Extract all content-word context words, ignore `the`, for `king` and `queen`. Work through every occurrence of each word in the corpus and list every context word you find.

Which context words do they share? Are there any context words unique to one but not the other?

28. Fill in the co-occurrence matrix below. Enter a `1` if the row-word and the column-word appear within a window of 2 in any sentence, and `0` otherwise. Ignore function words such as `the`.

29. Compare the rows for `king` and `queen`. Are they similar, or completely identical? Now compare `man` and `woman`. Finally, compare the `king / queen` pair to the `man / woman` pair. What does the three-group structure of the matrix predict about which words a model would learn to be similar, and which it would learn to be far apart?

Fill-in-the-Blank Game

Real word embedding models such as Word2Vec are trained by repeatedly solving a fill-in-the-blank task: *given the surrounding words, predict the missing center word.* A small neural network learns to do this, and its weights become the word embeddings.

30. For each sentence below, the center word is hidden. Write down which words from the vocabulary, `king`, `queen`, `prince`, `princess`, `man`, `woman`, could plausibly fill the blank, and which could never appear there:

```
the _____ ruled wisely
the _____ became leader
the _____ worked hard
```

31. Imagine a small neural network designed to solve the fill-in-the-blank task above. It would have three layers:

- **Input layer**: a vector with one slot per vocabulary word. A 1 is placed at the position of each *context* word, and 0 everywhere else. With a vocabulary of 6 words, this layer has 6 values, but several of them will be 1.
- **Hidden layer**: a small layer of 2 neurons that combines the context signals into a

single compact representation.

- **Output layer**: a probability distribution over the vocabulary, which word is most likely to be the missing center word?

Sketch this three-layer architecture on paper and label the size of each layer. Which layer contains the word embeddings? Can you control how many dimensions the embeddings have, or does the model decide that for itself?

32. Suppose this network were trained until it could reliably predict the missing center word. Reasoning from your co-occurrence matrix in Step 4: which pairs of words always appear with exactly the same set of context words? What does that mean for how similar their weight rows, and therefore their embeddings, would become?

Look at `king` and `queen`, then `prince` and `princess`, then `man` and `woman`. Could a model trained only on this toy corpus tell any of these pairs apart? What do all three pairs have in common, and what does this reveal about the corpus? How does this explain why real embedding models are trained on billions of sentences rather than dozens?

Step 5: GloVe Embeddings

So far you have rated words on three scales you designed yourself. Now you will load *real* word embeddings trained on billions of words, without connecting to any external API. The Jupyter notebook for this experiment provides the functions you need.

GloVe (Global Vectors for Word Representation) is a set of pre-trained word vectors published by Stanford University. Like the neural network in Step 4, GloVe discovers its dimensions automatically by reading billions of words from Wikipedia and news archives and factorizing the global co-occurrence matrix across the entire corpus. The version you will use gives each word a vector of 50 numbers.

Setup: Download `glove.6B.zip` from the Stanford NLP website, extract it, and place `glove.6B.50d.txt` in the same folder as your notebook. The file is about 160 MB.

33. Open the notebook and run the cell that loads GloVe. How many words does the model know? Print the vectors for `man` and `king`. Can you interpret any of the 50 numbers by looking at them directly, or do they only make sense through comparison with other vectors?

34. Compute cosine similarity for the same word pairs you used earlier, this time using GloVe vectors. Transfer your earlier results into a comparison table and add the GloVe values.

Where does GloVe agree with your manual ratings? Where does it diverge most, and which result feels more correct to you?

35. Test the analogy `king - man + woman = ?` with GloVe using the `find_nearest_glove` function in the notebook. Does it find `queen`? How does this compare to what your manual three-dimensional version returned earlier?

36. Try these two analogies with GloVe:

- `prince - boy + girl`
- `grandfather - man + woman`

Do the results improve on your manual ones? Explain briefly.

37. Use `find_nearest_glove` to find the five most similar words in the full 400,000-word vocabulary for `man`, `king`, and `queen`. Do the neighbours match your intuition? What does it mean that the model can answer this question for 400,000 words when your manual embedding covered only 16?

Step 6: Embeddings at Scale

GloVe gives each word a fixed vector of 50 numbers, learned from a large but static corpus. Modern models like the API-based embeddings from OpenAI go further: they can embed not just individual words but arbitrary text, they update over time, and they use 1536 or more dimensions. The provided Jupyter notebook provides the functions for this step.

38. Use the `get_embedding` function from the notebook to get embeddings for all 16 words from Step 2. How many numbers does each embedding contain, compared to your three-dimensional version and GloVe's 50?

39. Compute cosine similarity for the same word pairs using the API embeddings. Transfer the values from your earlier steps into a comparison table and add the API results.

Look across all three columns. Which pairs show the most agreement? Which show the greatest disagreement, and which model's result feels most correct for those pairs?

40. Repeat the analogy `king - man + woman` using the API embeddings and the `find_nearest_np` function from the notebook. Does it still find `queen`? How do the top

results compare across all three models, manual, GloVe, and API?

41. Try the analogy `Paris - France + Germany`. Does it work? Try one more analogy of your own choice.

Step 7: Visualizing the Space

The API embeddings have 1536 dimensions, far too many to draw. The notebook uses **PCA**, Principal Component Analysis, to project them down to 2 dimensions. Think of it like casting a shadow: the shadow is not the full object, but it still reveals something about its shape. Run the visualization cells in the notebook.

42. Look at the PCA plot of the 16 words. Which words ended up close together? Describe at least two clusters. Compare the plot to:

- your hand-drawn arrangement from Step 1
- the 2D grid you drew in Step 2, age versus gender
- the PCA plot from your manual embedding in Step 3

Where does the API model's arrangement agree with yours, and where does it differ? Do the royalty words cluster together?

43. Run the cell for this step and look at the plot with arrows for the pairs `(man, woman)`, `(king, queen)`, `(boy, girl)`, `(prince, princess)`. Are the arrows roughly parallel? What would it mean geometrically if they are, and how does that connect to the analogy `king - man + woman = queen`?

44. The PCA projection loses information: 1536 dimensions were compressed to 2. Do you think the clusters you see accurately represent the full semantic space, or are they a distortion? How would you test whether the projection is trustworthy?

Step 8: Semantic Search

So far you have worked with individual words. Embeddings work just as well for full sentences, and that makes Large Language Models possible. Only if a large context can be considered can LLMs truly predict the best next token.

Another application enabled by embedding whole sentences is **semantic search**. Instead of matching keywords, you match *meaning*. Run the semantic search cells in the notebook.

The notebook uses the following 15 sentences as the search database:

1. "The patient was taken to the emergency room immediately."
2. "She enjoys long walks in the forest on autumn mornings."
3. "The government announced new tax regulations last Thursday."
4. "A small dog was found wandering near the train station."
5. "Scientists discovered a new species of frog in the rainforest."
6. "The concert was sold out within minutes of tickets going on sale."
7. "He forgot to take his medication and felt unwell by the afternoon."
8. "The school organized a trip to the natural history museum."
9. "Renewable energy sources are becoming cheaper every year."
10. "A stray cat was hiding under the parked car."
11. "The doctor recommended rest and plenty of fluids."
12. "Children in the neighborhood built a treehouse last summer."
13. "Electric vehicles now outsell petrol cars in several countries."
14. "The injured player was carried off the pitch on a stretcher."
15. "Students spent the afternoon planting trees in the schoolyard."

45. Run a semantic search with the query `"A person receiving medical treatment"`. Which three sentences rank highest? Did the search find relevant sentences even though none of them contain the exact words from the query?

46. Now try the same topic using a keyword approach: go through the 15 sentences manually and mark every sentence that contains the word `"medical"` or `"treatment"`. How many matches does keyword search find compared to semantic search? Which approach gives more useful results?

47. Try the query `"Environmental sustainability and clean energy"`. Which sentences does semantic search return? Would a keyword search for `"energy"` return the same results?

48. Design a query that *fools* the semantic search, one where the top result is clearly not what you were looking for. What does this reveal about the limitations of semantic search?

Step 9: Reflection and Discussion

Reflect on the following questions and write down your thoughts. Discuss with your peers.

49. You went from free intuition, Step 1, to 3 named dimensions, Step 2, to 50 automatically learned dimensions, Step 5, to 1536 learned dimensions, Step 6. At what point did the representation start feeling less transparent to you, and why?

50. In Step 4, words like `happy`, `fast`, and `democracy` resisted your three-dimensional scheme. How do you think a model trained on billions of sentences handles these words? What does it use instead of hand-chosen dimensions?

51. In Step 4, you found that a model trained only on the toy corpus cannot distinguish `man` from `woman` because their context words are identical. What does this tell you about what an embedding actually encodes, meaning, or statistical patterns in language use? Are those the same thing?

52. The analogy `king - man + woman = queen` treats gender as a *direction in space*. Can you think of other concepts that might correspond to directions in a high-dimensional embedding? What might go wrong when a model encodes human concepts this way?

53. Semantic search found relevant sentences without any matching keywords. Your phone's search, email client, and many apps now work this way. Does that change how you think about what "search" means?

54. A student says: "The model clearly understands what `king` means, otherwise the analogy would not work." Based on everything you did in this experiment, how would you respond?