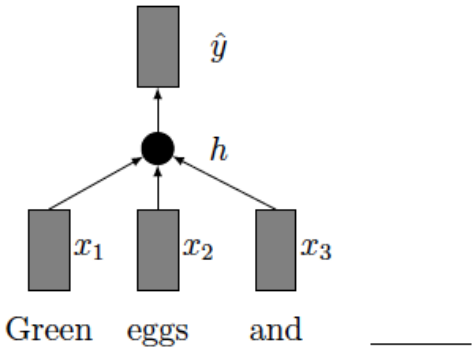
Advanced Human Language Technologies
Exercises on Word Embeddings

Word Embeddings

Exercise 1.

A feed-forward neural language model (NLM) is an alternative architecture for training word vectors. This architecture focuses on predicting a word given the N previous words. This is done by:

- Concatenating the word vectors $x_1 \dots x_N$ of the N previous words in a vector x of size $N \cdot D$ (being D the dimension of the input vectors x_i).
- Input x to a single hidden layer of size H with a non-linearity (e.g. tanh).
- Using a softmax layer predict the next word y, as a one hot vector of size V (number of words in the vocabulary).



$$x = [x1, x2, \dots, x_N]$$
$$h = \tanh(Wx + b)$$
$$\hat{y} = softmax(Uh + d)$$

$x \in \mathbb{R}^{N \cdot D}$
$W \in \mathbb{R}^{H \times N \cdot D}; \quad b \in \mathbb{R}^H$
$h \in \mathbb{R}^H$
$U \in \mathbb{R}^{V \times H}; \quad d \in \mathbb{R}^V$
$\hat{y} \in \mathbb{R}^V$

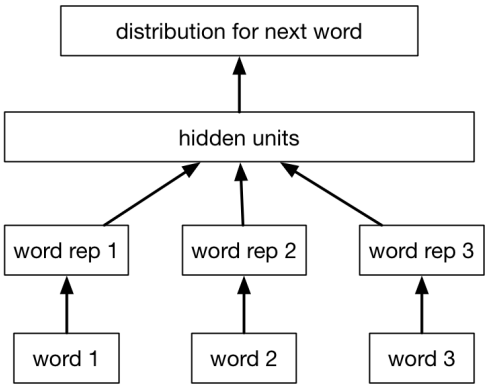
1. Mention 2 important differences between this feed-forward neural network LM and the CBOW model. Explain how these differences might affect the obtained word vectors.
2. Compute the number of required multiplications for the forward propagation in a feed-forward LM for a single training example.

Exercise 2.

1. Regarding the comparison between dense embedding vectors (such as those computed with word2vec or GloVe) and sparse vectors (such as coocurrence or TF·IDF), which of the following are true, and why?
   (a) Dense vectors generalize better to rare words.
   (b) Sparse vectors encode better similarity.
   (c) Dense vectors can infer representations for unseen words.
   (d) Dense vectors can represent polysemy and sparse vectors can't.

Exercise 3.

Explain which exact NLP task is the following neural architecture designed to learn



Exercise 4.

We have two datasets A and B containing examples of the same vocabulary V, and we train Word2Vec to obtain word embeddings with each of the datasets, obtaining two embedding models M_A and M_B.

Once trained, we observe that the dot product of the vectors for any two given words is the same in both models, i.e. $sim_A(u, w) = sim_B(u, w) \quad \forall u, v \in V$

1. Does this observation imply that datasets A and B contained the same exact examples, and thus we used the same training vectors in both cases? Why?

Exercise 5.

Consider the following simplified word embeddings in a 2D space:

Word	2-D embedding
king	$k = (0.7, 0.9)$
queen	$q = (0.8, 0.85)$
man	$m = (0.6, 0.7)$
woman	$w = (0.75, 0.8)$

1. Compute the **Euclidean distance** between *king* and *queen* and between *man* and *woman*. Which pair is closer?
2. Compute the **cosine similarity** between *king* and *queen*. What does this tell us about their relationship?
3. If we perform the analogy *man*→*king*, *woman*→*??*, which will be the result be in this 2D space? Which is the closest word ? Justify your answer.

Exercise 6.

Word embeddings capture semantic relationships. Using the famous *word analogy formula*:

$$\text{vec}(king) - \text{vec}(man) + \text{vec}(woman) \approx \text{vec}(queen)$$

1. Explain the rationale behind this formula.
2. Given the word pairs below, which ones might exhibit a similar analogy pattern in word embeddings?
   - (*Paris*, *France*) → (*Berlin*, ?)

- (*Teacher*, *School*) $\rightarrow$ (*Doctor*, ?)
- (*Sun*, *Day*) $\rightarrow$ (*Moon*, ?)

3. What types of word relationships *cannot* be effectively captured by word embeddings?

Exercise 7.

Consider a 3D word embedding space where words are positioned as follows:

Word	3D embedding
dog	$d = (0.8, 0.2, 0.6)$
cat	$c = (0.75, 0.25, 0.65)$
wolf	$w = (0.85, 0.1, 0.7)$
truck	$t = (0.1, 0.9, 0.2)$

1. Compute the **cosine similarity** between *dog* and *cat*.
2. Find the **nearest neighbor** to *wolf* using cosine similarity.
3. Why is cosine similarity often preferred over Euclidean distance for word embeddings?

Exercise 8.

Compare the following vector-based word representations and discuss their advantages and limitations:

1. **One-hot encoding** vs. **Word2Vec**: How do they differ in capturing semantic similarity?
2. **Word2Vec** vs. **GloVe**: How are their training methodologies different?
3. **Static word embeddings (e.g. Word2Vec, GloVe)** vs. **Contextual embeddings (e.g. BERT, ELMo)**: Why do newer models outperform traditional word embeddings in capturing word meaning?

Exercise 9.

Consider the word *bank*, which can mean a **financial institution** or a **riverbank**.

1. How would a traditional Word2Vec embedding represent *bank* ?
2. Why might this be a problem for NLP tasks?
3. How do **contextual embeddings** (e.g., from BERT) solve this issue?