# Assignment 2

## 1. Concepts, intuitions, and big picture

1. The n-gram models we've shown make the n-th order Markov assumption, i.e. that the distribution of words depends only on the previous n – 1 words. What properties of language will it not capture? Discuss (briefly) several distinct ways in which this assumption is false for natural language.

   **Answer:** There are several properties of language that n-gram models fail to capture. Some of these include -

   - Long distance dependencies between words: N-gram models view the relationship between words simply through the lense of frequency of co-occurence and limited to the number of words (i.e. n) that they occur within each other. They fail to capture the relationship of words that coexist out of that range of n.
   - Syntactic Structure: N-gram models do not capture grammatical structure of sentences, without understanding for any hierarchy or relationship beyond the n-gram window. As such, with n-grams, there is no concept of grammatically correct or incorrect sentences.
   - Semantic meaning: Similarly,they also do not capture sentence semantics or meaning. As such, they are blind to aspects of language that depend on context, especially out of that n-gram window.

2. Follow-up to previous question: Despite the Markov assumption, n-gram models are remarkably good at predicting the next word. Discuss why this might be. What information is in the previous word(s) that makes these models perform so surprisingly well? In particular, what kinds of grammatical information do they capture?

   **Answer:** Despite not being able to capture long-term dependencies, syntactic structure, and semantic meaning beyond the n-gram window, n-grams tends to perform quite well in at predicting the next word because they are good at capturing local patterns, i.e. word patterns within the n-gram window. While language is complex overall, it also consists of many patterns that are local, statistically regular in their occurences, and therefore predictable e.g. verbs typically follow nouns or pronouns, and adverbs tend to follow verbs. As such, when trained with a diverse and large-enough data corpus, n-grams can be quite effective such that even a limited context window can provide enough information for accurately rpedicting the next word.

3. Explain how are perplexity and cross-entropy loss related?

**Answer:** Cross-entropy loss is the a measure used to evaluate a language model's prediction error, i.e. how well the predicted probability distribution of the next word matches the actual probability distribution of the next word. Mathematically, perplexity is an exponential of cross-entropy loss. It also measures how well a probability-based language model predicts, but it provides a more intuitive interpretation of the model's performance as it can be viewed as the weighted average number of choices that the model has when predicting the next word. In other words, for every prediction that a model has to make, on average, how many effective choices does it have to consider. Just like cross-entrop loss, the lower the perplexity, the better the model. For illustrative purposes, if the perplexity of a model is 1, that can be seen as saying that for every prediction the model has to make, the amount of confusion it faces is on average eequivalent to chosing between 1 effective choice, which is the same as saying it's 100% sure to have the correct answer. If the perplexity is 10, that can be interepreted as saying for each prediction/decision that the model has to make, the amount of confusion it faces is equivalent to having to choose between 10 effective choices.

4. For a vocabulary of $|V|$ words, what would you expect perplexity to be if your model predictions were completely random? Compute the corresponding cross-entropy loss for $|V| = 2000$ and $|V| = 10000$, and keep this in mind as a baseline.

   **Answer:**
   Let $Cross - Entropy$ loss of a single prediction be $H$
   Let the probability assigned to the actual next word be $P(w_{actual}) = \frac{1}{|V|}$
   $H = -\log_2 P(w_{actual}) = \log_2 |V|$

In [1]:
```
import math

V1 = 2000
ce1 = math.log2(V1)
print(f"Cross-entropy loss for |V| = {V1}: {ce1:.4f}")

V2 = 10000
ce2 = math.log2(V2)
print(f"Cross-entropy loss for |V| = {V2}: {ce2:.4f}")
```

```
Cross-entropy loss for |V| = 2000: 10.9658
Cross-entropy loss for |V| = 10000: 13.2877
```

5. Which of these are reasons for the recent wave of neural networks taking off? (check the options that apply.)

   a. We have access to a lot more computational power
   b. Neural Networks are a brand new field.
   c. We have access to a lot more data.
   d. There has been significant improvements in benchmarks in speech recognition and NLP

**Answer**: a, c, d

6. What does a neuron compute?

   a. A neuron computes an activation function followed by a linear function (z = Wx + b)
   b. A neuron computes a linear function (z = Wx + b) followed by an activation function.
   c. A neuron computes a function g that scales the input x linearly (Wx + b).
   d. A neuron computes the mean of all features before applying the output to an activation function.

   **Answer:** b

7. What is the point of applying a Softmax function to the logits output by a sequence classification model

   a. It softens the logits so that they're more reliable.
   b. It applies a lower and upper bound so that they're understandable.
   c. The total sum of the output is then 1, resulting in a possible probabilistic interpretation.

   **Answer:** c

## 2. Linear Algebra Recap

### 2.1 Gradients

Consider the following scalar-valued function:

$$f(x, y, z) = x^2y + sin(z + 6y)$$

1. Compute partial derivatives with respect to $x, y, z$.

   $$\frac{\partial f}{\partial x} = 2xy$$

   $$\frac{\partial f}{\partial y} = x^2 + 6\cos(z + 6y)$$

   $$\frac{\partial f}{\partial z} = \cos(z + 6y)$$

   $$\nabla f(x, y, z) = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right]^\top$$

   Substituting the partial derivatives:

   $$\nabla f(x, y, z) = \left[2xy, x^2 + 6\cos(z + 6y), \cos(z + 6y)\right]^\top$$

2. We can consider $f$ to be a function that takes a vector $\theta \in \mathbb{R}^3$ as input, where $\theta = [x, y, z]^\top$. Write the gradient as a vector and evaluate it at $\theta = [3, \pi/2, 0]^\top$.

For $\theta = [3, \pi/2, 0]^\top$, $x = 3$, $y = \pi/2$, and $z = 0$

$$\nabla f(x, y, z) = \left[2xy, x^2 + 6\cos(z + 6y), \cos(z + 6y)\right]^\top$$

Substituting $x, y, z$:

$$\nabla f(3, \pi/2, 0) = \left[2(3)(\pi/2), (3)^2 + 6\cos(0 + 6(\pi/2)), \cos(0 + 6(\pi/2))\right]^\top$$

$$\nabla f(3, \pi/2, 0) = \left[3\pi, 9 + 6\cos(3\pi), \cos(3\pi)\right]^\top$$

Substituting $\cos(3\pi) = -1$

$$\nabla f(3, \pi/2, 0) = \left[3\pi, 9 + 6(-1), -1\right]^\top$$

$$\nabla f(3, \pi/2, 0) = \left[3\pi, 9 - 6, -1\right]^\top$$

$$\nabla f(3, \pi/2, 0) = \left[3\pi, 3, -1\right]^\top$$

### 2.2 Gradients of vectors

Let $\mathbf{x}, \mathbf{c} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$. For the following parts, before taking any derivatives, identify what the derivative looks like (is it a scalar, vector, or matrix?) and how we calculate each term in the derivative. Then carefully solve for an arbitrary entry of the derivative, then stack/arrange all of them to get the final result.

- Show that $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top \mathbf{c}) = \mathbf{c}^\top$

  Let $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ and $\mathbf{c} = [c_1, c_2, \ldots, c_n]^\top$.

  The expression $\mathbf{x}^\top \mathbf{c}$ is a scalar value resulting from the dot product of the two vectors: $\mathbf{x}^\top \mathbf{c} = \sum_{i=1}^n x_i c_i = x_1 c_1 + x_2 c_2 + \cdots + x_n c_n$

  The derivative of a scalar function with respect to a vector $\mathbf{x}$ is a vector where each element is the partial derivative of the scalar function with respect to the corresponding element of $\mathbf{x}$.

  Hence, the $j$-th element of the gradient vector $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top \mathbf{c})$ is the partial derivative of $\mathbf{x}^\top \mathbf{c}$ with respect to $x_j$:

  $$\left[\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top \mathbf{c})\right]_j = \frac{\partial}{\partial x_j}(\mathbf{x}^\top \mathbf{c}) = \frac{\partial}{\partial x_j}\left(\sum_{i=1}^n x_i c_i\right)$$

  When solving for partial derivative with respect to $x_j$, all terms $x_i c_i$ where $i \neq j$ will are treated as constants, and their partial derivative with respect to $x_j$ is 0.

  $$\frac{\partial}{\partial x_j}\left(x_1 c_1 + x_2 c_2 + \cdots + x_j c_j + \cdots + x_n c_n\right) = 0 + 0 + \cdots + \frac{\partial}{\partial x_j}(x_j c_j) + \cdots + 0$$

  $$\frac{\partial}{\partial x_j}(x_j c_j) = c_j$$

  As shown above, the $j$-th element of the gradient vector is $c_j$.

  Hence, $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top \mathbf{c}) = [c_1, c_2, \ldots, c_n]^\top = \mathbf{c}^\top$

- Show that $\frac{\partial}{\partial \mathbf{x}}(\|\mathbf{x}\|_2^2) = 2\mathbf{x}^\top$.

  With $\|\|\mathbf{x}\|\|$ as the euclidean norm of a vector (i.e. $\sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$), the squared euclidean norm of a vector $\|\|\mathbf{x}\|\|_2^2$ is given by:

  $\|\mathbf{x}\|_2^2 = \mathbf{x}^\top\mathbf{x} = \sum_{i=1}^n x_i^2 = x_1^2 + x_2^2 + \cdots + x_n^2$

  Similar to above, the derivative of a scalar function with respect to a vector $\mathbf{x}$ is a vector where each element is the partial derivative of the scalar function with respect to the corresponding element of $\mathbf{x}$.

  The partial derivate with respect to $x_j$ is:

  $\frac{\partial}{\partial x_j}(x_1^2 + x_2^2 + \cdots + x_j^2 + \cdots + x_n^2) = 0 + 0 + \cdots + \frac{\partial}{\partial x_j}(x_j^2) + \cdots + 0$

  $\frac{\partial}{\partial x_j}(x_j^2) = 2x_j$

  Hence:

  $\frac{\partial}{\partial \mathbf{x}}(\|\mathbf{x}\|_2^2) = [2x_1, 2x_2, \ldots, 2x_n]^\top$

  Factor out 2:

  $\frac{\partial}{\partial \mathbf{x}}(\|\mathbf{x}\|_2^2) = 2[x_1, x_2, \ldots, x_n]^\top$

  For $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$:

  $\frac{\partial}{\partial \mathbf{x}}(\|\mathbf{x}\|_2^2) = 2\mathbf{x}^\top$

- Show that $\frac{\partial}{\partial \mathbf{x}}(\mathbf{A}\mathbf{x}) = \mathbf{A}$.

  Let $\mathbf{y} = \mathbf{A}\mathbf{x}$. The $i$-th component of $\mathbf{y}$ is $y_i = \sum_{j=1}^n a_{ij}x_j$.

  The derivative of the vector $\mathbf{y}$ with respect to the vector $\mathbf{x}$ is a matrix where the element in the $i$-th row and $j$-th column is the partial derivate of $\mathbf{y}_i$ with respect to $\mathbf{x}_j$, i.e. $\frac{\partial y_i}{\partial x_j}$.

  $\frac{\partial y_i}{\partial x_j} = \frac{\partial}{\partial x_j}\left(a_{i1}x_1 + \cdots + a_{ij}x_j + \cdots + a_{in}x_n\right)$

  When differentiating with respect to $x_j$, all terms except $a_{ij}x_j$ are constant, so:

  $\frac{\partial y_i}{\partial x_j} = a_{ij}$

  Since the element in every $i$-th row and $j$-th column of the derivative matrix $\frac{\partial y_i}{\partial x_j}$ ( or $\frac{\partial}{\partial \mathbf{x}}(\mathbf{A}\mathbf{x})$) is $a_{ij}$, the resulting matrix itself is $\mathbf{A}$.

  Hence: $\frac{\partial}{\partial \mathbf{x}}(\mathbf{A}\mathbf{x}) = \mathbf{A}$

- Show that $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top\mathbf{A}\mathbf{x}) = \mathbf{x}^\top(\mathbf{A} + \mathbf{A}^\top)$.

2.3 Jacobian

Consider the following vecor function from $\mathbb{R}^3$ to $\mathbb{R}^3$:

$$\mathbf{f}(\theta = [x_1, x_2, x_3]) = \begin{cases} \sin(x_1 x_2 x_3) \\ \cos(x_2 + x_3) \\ \exp\left(-\frac{1}{2} x_3^2\right) \end{cases}$$

1. What is the Jacobian matrix of $\mathbf{f}(\theta)$?

The Jacobian matrix $J_{\mathbf{f}}(\theta)$ is a matrix where the element in the $i$-th row and $j$-th column is the partial derivative of the $i$-th component function $f_i$ with respect to the $j$-th variable $x_j$.

$$J_{\mathbf{f}}(\theta) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{pmatrix}$$

**Partial derivatives of $f_1(x_1, x_2, x_3) = \sin(x_1 x_2 x_3)$:**

- $\frac{\partial f_1}{\partial x_1} = \cos(x_1 x_2 x_3) \cdot \frac{\partial}{\partial x_1}(x_1 x_2 x_3) = \cos(x_1 x_2 x_3) \cdot (x_2 x_3)$
- $\frac{\partial f_1}{\partial x_2} = \cos(x_1 x_2 x_3) \cdot \frac{\partial}{\partial x_2}(x_1 x_2 x_3) = \cos(x_1 x_2 x_3) \cdot (x_1 x_3)$
- $\frac{\partial f_1}{\partial x_3} = \cos(x_1 x_2 x_3) \cdot \frac{\partial}{\partial x_3}(x_1 x_2 x_3) = \cos(x_1 x_2 x_3) \cdot (x_1 x_2)$

**Partial derivatives of $f_2(x_1, x_2, x_3) = \cos(x_2 + x_3)$:**

- $\frac{\partial f_2}{\partial x_1} = \frac{\partial}{\partial x_1}(\cos(x_2 + x_3)) = 0$ (since the expression does not contain $x_1$)
- $\frac{\partial f_2}{\partial x_2} = -\sin(x_2 + x_3) \cdot \frac{\partial}{\partial x_2}(x_2 + x_3) = -\sin(x_2 + x_3) \cdot 1 = -\sin(x_2 + x_3)$
- $\frac{\partial f_2}{\partial x_3} = -\sin(x_2 + x_3) \cdot \frac{\partial}{\partial x_3}(x_2 + x_3) = -\sin(x_2 + x_3) \cdot 1 = -\sin(x_2 + x_3)$

**Partial derivatives of $f_3(x_1, x_2, x_3) = \exp\left(-\frac{1}{2} x_3^2\right)$:**

- $\frac{\partial f_3}{\partial x_1} = \frac{\partial}{\partial x_1}\left(\exp\left(-\frac{1}{2} x_3^2\right)\right) = 0$ (since the expression does not contain $x_1$)
- $\frac{\partial f_3}{\partial x_2} = \frac{\partial}{\partial x_2}\left(\exp\left(-\frac{1}{2} x_3^2\right)\right) = 0$ (since the expression does not contain $x_2$)
- $\frac{\partial f_3}{\partial x_3} = \exp\left(-\frac{1}{2} x_3^2\right) \cdot \frac{\partial}{\partial x_3}\left(-\frac{1}{2} x_3^2\right) = \exp\left(-\frac{1}{2} x_3^2\right) \cdot \left(-\frac{1}{2} \cdot 2 x_3\right) = -x_3 \exp\left(-\frac{1}{2} x_3^2\right)$

Substitute the partial derivates to get the Jacobian of $\mathbf{f}(\theta)$:

$$J_{\mathbf{f}}(\theta) = \begin{pmatrix} x_2 x_3 \cos(x_1 x_2 x_3) & x_1 x_3 \cos(x_1 x_2 x_3) & x_1 x_2 \cos(x_1 x_2 x_3) \\ 0 & -\sin(x_2 + x_3) & -\sin(x_2 + x_3) \\ 0 & 0 & -x_3 \exp\left(-\frac{1}{2} x_3^2\right) \end{pmatrix}$$

2. Evaluate the Jacobian matrix of $\mathbf{f}(\theta)$ at $\theta = [1, \pi, 0]$.

Substituting $x_1 = 1, x_2 = \pi, x_3 = 0$ for the functions in the matrix above, we get:

- $x_2 x_3 \cos(x_1 x_2 x_3) = (\pi)(0)\cos((1)(\pi)(0)) = 0 \cdot \cos(0) = 0 \cdot 1 = 0$
- $x_1 x_3 \cos(x_1 x_2 x_3) = (1)(0)\cos((1)(\pi)(0)) = 0 \cdot \cos(0) = 0 \cdot 1 = 0$
- $x_1 x_2 \cos(x_1 x_2 x_3) = (1)(\pi)\cos((1)(\pi)(0)) = \pi \cdot \cos(0) = \pi \cdot 1 = \pi$
- $-\sin(x_2 + x_3) = -\sin(\pi + 0) = -\sin(\pi) = -0 = 0$
- $-\sin(x_2 + x_3) = -\sin(\pi + 0) = -\sin(\pi) = -0 = 0$
- $-x_3 \exp\left(-\frac{1}{2}x_3^2\right) = -(0)\exp\left(-\frac{1}{2}(0)^2\right) = 0 \cdot \exp(0) = 0 \cdot 1 = 0$

$$J_{\mathbf{f}}([1, \pi, 0]^\top) = \begin{pmatrix} 0 & 0 & \pi \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

## $n$-gram Language Models

### 3.1 A toy $n$-gram language model

Consider the following vocabulary, $V$ = {BOS, EOS, who, framed, roger, rabbit, the} where BOS is the dummy token indicating the beginning of a sentence, and EOS indicates the end of a sentence. Consider the following training data:

```
BOS who EOS
BOS who framed roger EOS
BOS roger rabbit EOS
BOS who framed roger rabbit EOS
BOS roger framed who EOS
BOS who framed the rabbit EOS
```

1. Compute the following probabilities: $P(\text{rabbit})$, $P(\text{rabbit}|\text{roger})$, $P(\text{EOS}|\text{rabbit})$

   **Counts of Unigrams:**

   - BOS: 6
   - who: 5
   - framed: 4
   - roger: 4
   - rabbit: 3
   - EOS: 6
   - the: 1

   Total number of word tokens in the corpus: 29 tokens.

   **Counts of Bigrams ("roger rabbit" and "rabbit EOS"):**

   - "roger rabbit": Appears in sentences 3 and 4. Count = 2.
   - "rabbit EOS": Appears in sentences 3, 4, and 6. Count = 3.

   $P(\text{rabbit}) = \frac{\text{Count(rabbit)}}{\text{Total tokens}} = \frac{3}{29}$

   $P(\text{rabbit}|\text{roger}) = \frac{\text{Count(roger rabbit)}}{\text{Count(roger)}} = \frac{2}{4} = \frac{1}{2}$

$$P(\text{EOS}|\text{rabbit}) = \frac{\text{Count(rabbit EOS)}}{\text{Count(rabbit)}} = \frac{3}{3} = 1$$

2. Briefly explain what the sparsity problem is in $n$-gram language models?

   The sparsity problem with $n$-gram language models arise as the $n$ value increases, resulting in an exponentially large mount of possible word combinations or sequences that can result from a training corpus. For example, for a corpus of 5 unique token, the number of possible sequences for a unigram is 5, for a bigram is $5^2$, for a trigram is $5^3$, and so on. For corpuses with large numbers of tokens, this results in many unique sequences will either not appear, or will apear infrequently, hence the sparsity. During prediction, the $n$-gram will assign 0 probability to sequences that it did not encounter in the training data corpus. As a result, this leads to poor generalization of the model and inacurate language modeling that is skewed towards the training data as opposed to general language patterns.

## Challenges of linear classifiers of sentences

A simple way to compute a representation for a phrase $s$ is to add up the representations of the words in that phrase: $\text{repr}(s) = \sum_{w \in s} \mathbf{v}_w$, where $w \in s$ are the words in $s$ and $\mathbf{v}_w \in \mathbb{R}^d$ is the embedding for word $w$.

1. Now, consider sentiment analysis on a phrase $s$ in which the predicted sentiments are

   $$f(s; \theta) = \theta \cdot \text{repr}(s)$$

   for some choice of parameters $\theta$. Note that here $\theta \in \mathbb{R}^d$ and "$\cdot$" is the "dot-product". Prove that in such a model, the following inequalities cannot hold for any choice of $\theta$ and word embeddings:

   $$f(\text{good}; \theta) > f(\text{not good}; \theta)$$

   $$f(\text{bad}; \theta) < f(\text{not bad}; \theta)$$

   Thereby, showing the inadequcy of this model in capturing negations.

**Answer:**

For $f(\text{good}; \theta) > f(\text{not good}; \theta)$:

$$\text{repr}(\text{good}) = \mathbf{v}_{\text{good}}$$

$$\text{repr}(\text{not good}) = \mathbf{v}_{\text{not}} + \mathbf{v}_{\text{good}}$$

$$\theta \cdot \mathbf{v}_{\text{good}} > \theta \cdot (\mathbf{v}_{\text{not}} + \mathbf{v}_{\text{good}})$$

$$\theta \cdot \mathbf{v}_{\text{good}} > \theta \cdot \mathbf{v}_{\text{not}} + \theta \cdot \mathbf{v}_{\text{good}}$$

$$0 > \theta \cdot \mathbf{v}_{\text{not}}$$

For $f(\text{bad}; \theta) < f(\text{not bad}; \theta)$:

$$\text{repr}(\text{bad}) = \mathbf{v}_{\text{bad}}$$

$$\text{repr}(\text{not bad}) = \mathbf{v}_{\text{not}} + \mathbf{v}_{\text{bad}}$$

$$\theta \cdot \mathbf{v}_{\text{bad}} < \theta \cdot (\mathbf{v}_{\text{not}} + \mathbf{v}_{\text{bad}})$$

$$\theta \cdot \mathbf{v}_{\text{bad}} < \theta \cdot \mathbf{v}_{\text{not}} + \theta \cdot \mathbf{v}_{\text{bad}}$$

$$0 < \theta \cdot \mathbf{v}_{\text{not}}$$

This sytem cannot hold because the two inequalities $0 > \theta \cdot \mathbf{v}_{\text{not}}$ and $0 < \theta \cdot \mathbf{v}_{\text{not}}$ are contradictory. They both cannot be simultanously true.

2. **Extra Credit:** Construct another example of a pair of inequalities similar to the ones above that cannot both hold

   **Answer:**

   $f(\text{win}; \theta) > f(\text{not win}; \theta)$

   $f(\text{lose}; \theta) < f(\text{not lose}; \theta)$

3. **Extra Credit:** Consider a slight modification to the previous predictive model:

   $$f(s; \theta) = \theta \cdot \text{ReLU}(\text{repr}(s))$$

   where ReLU (rectified linear function) is defined as:

   $$\text{ReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

   Given this choice of predictive function, show that it is possible to satisfy the above inequalities for some choice of $\theta$. Hint: Show there exist parameters $\theta$ and word embeddings $\mathbf{v}_{\text{good}}$, $\mathbf{v}_{\text{bad}}$, and $\mathbf{v}_{\text{not}}$ such that the inequalities are satisfied.

Let:

- $\theta = [1, 1]^\top$
- $\mathbf{v}_{\text{good}} = [3, 0]^\top$
- $\mathbf{v}_{\text{bad}} = [-3, 0]^\top$
- $\mathbf{v}_{\text{not}} = [-4, 1]^\top$

**For $f(\text{good}; \theta) > f(\text{not good}; \theta)$:**

$f(\text{good}; \theta) = \theta \cdot \mathbf{v}_{\text{good}} = \theta \cdot \text{ReLU}(\text{repr}(\text{good})) = [1, 1]^\top \cdot [3, 0]^\top = (1)(3) + (1)(0) =$

$f(\text{not good}; \theta) = \theta \cdot (\mathbf{v}_{\text{not}} + \mathbf{v}_{\text{good}}) = \theta \cdot \text{ReLU}(\text{repr}(\text{not good})) = [1, 1]^\top \cdot [\text{ReLU}(-$

Here, $f(\text{good}; \theta) > f(\text{not good}; \theta)$, i.e. $3 > 1$ (True).

**For $f(\text{bad}; \theta) < f(\text{not bad}; \theta)$**

$f(\text{bad}; \theta) = \theta \cdot \mathbf{v}_{\text{bad}} = \theta \cdot \text{ReLU}(\text{repr}(\text{bad})) = [1, 1]^\top \cdot [0, 0]^\top = (1)(0) + (1)(0) = 0$

$f(\text{not bad}; \theta) = \theta \cdot (\mathbf{v}_{\text{not}} + \mathbf{v}_{\text{bad}}) = \theta \cdot \text{ReLU}(\text{repr}(\text{not bad})) = [1, 1]^\top \cdot [\text{ReLU}(-7),$

Here, $f(\text{bad}; \theta) < f(\text{not bad}; \theta)$, i.e. $0 < 1$ (True).

Since both inequalities $f(\text{good}; \theta) > f(\text{not good}; \theta)$ and $f(\text{bad}; \theta) < f(\text{not bad}; \theta)$ hold true with this specific choice of $\theta$ and embeddings, this example shows that it is possible to satisfy them using the ReLU model.

4. Given the above result, explain (in 1-2 sentence) why the use of neural networks (which have more complexity than linear models)

   Unlike linear models, neural networks incorporate non-linear activation functions (such as ReLU, sigmoid, or tanh) within their neurons that introduce non-linearity into the model's computations. As shown above, this non-linearity enables the network to learn and model complex non-linear relationships between the input and the output, which are vastly present in natural language. The examples illustrates how it's possible to capture negations in language through a model that uses the Rectified Linear Unit (ReLU) function to introduce non-linearity, compared to the simpler linear model.

## 5 Softmax function

Remember the Softmax functions from the class.

Softmax: $\sigma(\mathbf{z}) \triangleq [\sigma(\mathbf{z})_1, \ldots, \sigma(\mathbf{z})_K]$ s.t. $\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ for $i = 1, \ldots, K$ and $\mathbf{z} = (z_1, \ldots, z_K)$

1. Prove that Softmax is invariant to constant offsets in the input, i.e., for any input vector $\mathbf{z}$ and any constant $c$,

   $$\sigma(\mathbf{z}) = \sigma(\mathbf{z} + c)$$

   **Pro tip:** We make use of this property in practice to increase the numerical stability of our models. Specifically, using $c = -\max_{i \in \{1 \ldots K\}} z_i$, i.e., subtracting its maximum element from all elements of $\mathbf{z}$ would prevent numerical instability due to large values.

**Answer:**

The $i$-th component of $\sigma(\mathbf{z})$ is given by:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

The $i$-th component of $\sigma(\mathbf{z} + c)$, where the input vector for $\mathbf{z} + c = [z_1 + c, \ldots, z_K + c]$ is given by:

$$\sigma(\mathbf{z} + c)_i = \frac{e^{z_i + c}}{\sum_{j=1}^{K} e^{z_j + c}}$$

Using the property of exponents $e^{a+b} = e^a e^b$:

$$\sigma(\mathbf{z} + c)_i = \frac{e^{z_i} e^c}{\sum_{j=1}^{K} (e^{z_j} e^c)}$$

Factoring out $e^c$ from the denominator since it's a common term in the sum:

$$\sigma(\mathbf{z} + c)_i = \frac{e^{z_i} e^c}{e^c \sum_{j=1}^{K} e^{z_j}}$$

Cancel out $e^c$ from both numerator and denominator:

$$\sigma(\mathbf{z} + c)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} = \sigma(\mathbf{z})_i$$

Since this holds for every component $i = 1, \ldots, K$:

$$\sigma(\mathbf{z} + c) = \sigma(\mathbf{z})$$

2. Softmax maintains the relative order of the elements in $\mathbf{z}$. In particular, show that the largest index is intact after applying Softmax:

$$\underset{i \in \{1 \ldots K\}}{\arg\max}\, z_i = \underset{i \in \{1 \ldots K\}}{\arg\max}\, \sigma(\mathbf{z})_i$$

**Answer:**

For any 2 elements $z_i$ and $z_j$, if $z_i > z_j$, then $\sigma(\mathbf{z})_i > \sigma(\mathbf{z})_j$.

$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$ and $\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$.

The denominator $\sum_{k=1}^{K} e^{z_k}$ is the same and positive for all components since the exponential function is always positive for any real input.

The exponential function $f(x) = e^x$ is also strictly monotonically increasing, i.e. if $z_i > z_j$, then $e^{z_i} > e^{z_j}$.

Hence, while $\sigma(\mathbf{z})_i$ and $\sigma(\mathbf{z})_j$ always have the sam epositive denominator, their numerators will also be positive and increasing with the $\mathbf{z}$ term. As such, the inequality $e^{z_i} > e^{z_j}$ directly implies that $\sigma(\mathbf{z})_i > \sigma(\mathbf{z})_j$

This confirms that Softmax preserves the relative order of the input elements. If $z_i$ is the largest element in $\mathbf{z}$, then $\sigma(\mathbf{z})_i$ will also be the largest element in $\sigma(\mathbf{z})$, and the index of the maximum element remains unchanged.

3. Define the Sigmoid function as follows:

$$\text{Sigmoid} : S(x) \triangleq \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} = 1 - S(-x)$$

Prove that Softmax is equivalent to the Sigmoid functions when the number of possible labels is two: $K = 2$. Specifically:

$$\sigma_1([z_1, z_2]) = S(z_1 - z_2)$$

**Answer:**

Definitions for softmax($\sigma_1$) and Sigmoid :

$$\sigma_1([z_1, z_2]) = \frac{e^{z_1}}{e^{z_1} + e^{z_2}}$$

$$S(x) = \frac{1}{1 + e^{-x}}$$

$S(z_1 - z_2)$:

$$\frac{1}{1 + e^{-(z_1 - z_2)}}$$

Using exponent properties:

$$\frac{1}{1 + e^{-z_1} e^{z_2}}$$

Multiplying the numerator and denominator by $e^{z_1}$:

$$\frac{1 \cdot e^{z_1}}{(1 + e^{-z_1} e^{z_2}) \cdot e^{z_1}}$$

$$\frac{e^{z_1}}{e^{z_1} + e^{-z_1} e^{z_2} e^{z_1}}$$

Since $e^{-z_1} e^{z_1} = e^{(-z_1 + z_1)} = e^0 = 1$:

$$S(z_1 - z_2) = \frac{e^{z_1}}{e^{z_1} + e^{z_2}} = \sigma_1([z_1, z_2])$$

4. **Extra Credit:** Next, let's extend (1) to prove the following inequality:

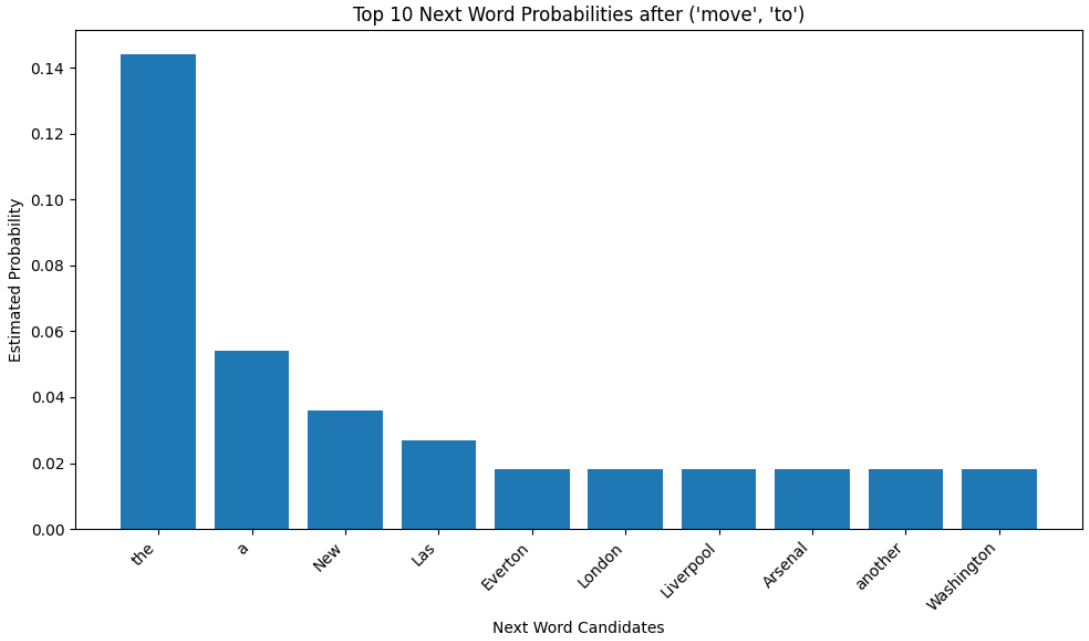
$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \geq \prod_{\substack{j=1 \\ j \neq i}}^{K} \frac{1}{1 + e^{-(z_i - z_j)}} = \prod_{\substack{j=1 \\ j \neq i}}^{K} S(z_i - z_j)$$

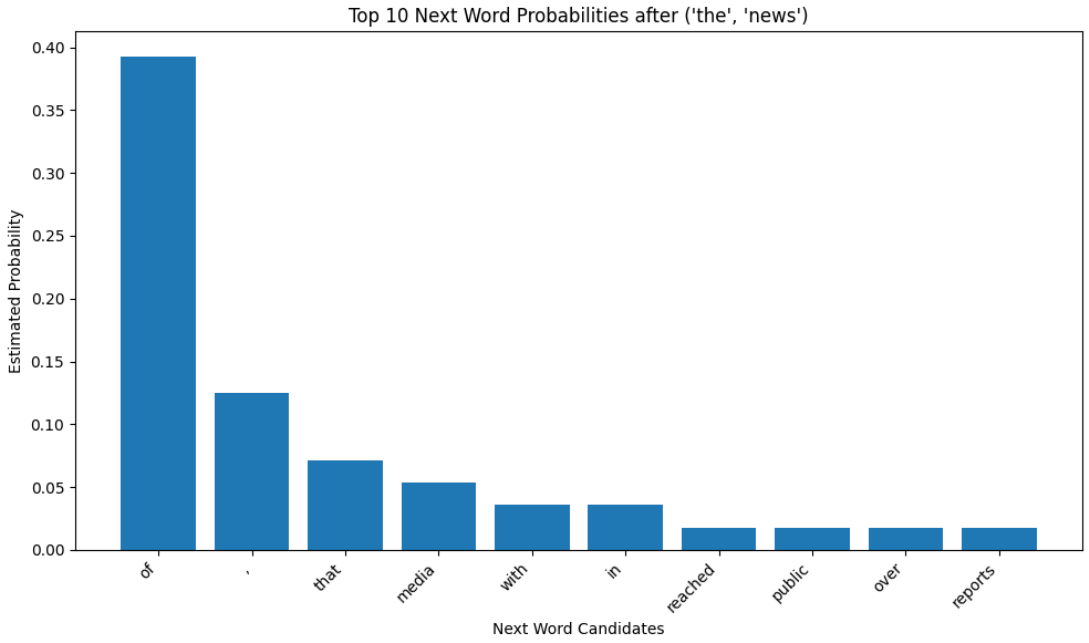
**Hint:** Use the following inequality $(1 + \sum_i \alpha_i) \leq \prod_i (1 + \alpha_i)$ where each $\alpha_i \geq 0$.

## 6 Programming

### 6.1.4 Visualize the N-gram distribution

Complete the plot_next_word_prob function in ngram_lm.py, and run run_ngram in main.py to plot the top-10 most probable next token given an input context. Paste the two plots for the two given contexts (provided in run_ngram), and describe in 2-3 sentences your findings.




The observations from the charts above are similar in that for their respective sequences, the most probable next words of the 10 likeliest options is significantly more likely to occur than the remaining 9. These 2 words, i.e. "the" in the case of "move to" and "of" in the case of "the news", are not surprising as they are common parts of speech. "the" is a definite article that often preceeds nouns, while "of" is preposition often used to introduce objects. Both articles and prepositions play important roles in connecting and highlighting the relationships between different words in a sentence.

### 6.1.5 Generation: Sample from LM

---------- generated text 1 ----------
According to the report, the first time in the United States, and the other hand, the first time in the United States, and the other hand, the first time
---------- generated text 2 ----------
The president of the association ' s " The One I Love You ' re not here to Connacht side

of the game ' s " The One I Love You ' re not

As expected of $n$-grams, especially with greedy decoding as the sampling procedure, the model isn't effective at generating meaningful/coherent sentences. While each $n$th word appears to sounds beside the $n-1$th word, the entire sentence doesn't necessarily make sense. This is a common limitation of $n$-grams.

### 6.2.4 Gradient Descent