FNLP Tutorial 3

Question 1. A bigram language model

We are given the following corpus, modified from J&M (Chapter 3 in the 3rd edition):

<s> language is awesome </s>
<s> language is awesome </s>
<s> language and students </s>
<s> awesome students like language </s>

1. Using a bigram language model without smoothing, generate 4 sentences, starting from '<s>'. To generate sentences manually, randomly choose a real number between 0 and 1, and use the cumulative probabilities of words in the vocabulary to select a word from the language model. For example, if there are two words in the vocabulary with non-zero conditional probabilities $P(\text{one}|\text{<s>}) = 0.25$ and $P(\text{two}|\text{<s>}) = 0.75$, then choosing a random number between 0 and 0.25 would result in 'one', whereas a random number between 0.25 and 1 would result in 'two'.
2. How does the lack of smoothing impact the novelty of the sentences generated?
3. How does the nature of the corpus impact the lengths of the generated sentences?

Prefix	**(Non-zero) Conditional probabilities**		
<s>	$P(\text{language}\|\text{<s>}) = \frac{3}{4}$	$P(\text{awesome}\|\text{<s>}) = \frac{1}{4}$	
and	$P(\text{students}\|\text{and}) = 1$		
awesome	$P(\text{</s>}\|\text{awesome}) = \frac{2}{3}$	$P(\text{students}\|\text{awesome}) = \frac{1}{3}$	
is	$P(\text{awesome}\|\text{is}) = 1$		
language	$P(\text{is}\|\text{language}) = \frac{1}{2}$	$P(\text{and}\|\text{language}) = \frac{1}{4}$	$P(\text{</s>}\|\text{language}) = \frac{1}{4}$
like	$P(\text{language}\|\text{like}) = 1$		
students	$P(\text{</s>}\|\text{students}) = \frac{1}{2}$	$P(\text{like}\|\text{students}) = \frac{1}{2}$	

Table 1: The non-zero conditional probabilities of the language model discussed in Exercise 2.1.

Solution

1. Table 1 provides the non-zero conditional probabilities that can be used to generate sentences for this question. Some example sentences that can be generated, are:
   - <s> awesome students like language and students like language </s>
   - <s> awesome students </s>
   - <s> language </s>
   - <s> language is awesome students like language </s>
   - <s> language is awesome </s>
   - <s> language </s>

- <s> awesome </s>
- <s> language </s>
- <s> awesome </s>
- <s> language and students like language is awesome students like language and students like language and students like language is awesome students like language </s>

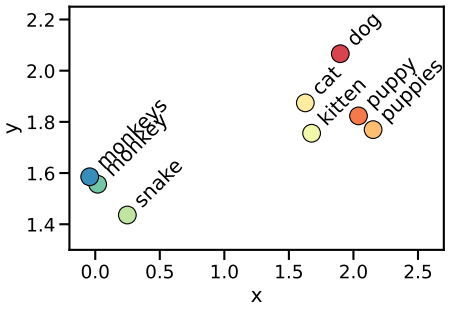
2. The language model can generate sentences that are novel (i.e. that were not in the training corpus). In fact, 9 out of 10 example sentences above were not in the training corpus. However, every bigram does exist in the training corpus, so the language model is severely limited in the sentences it can generate.

3. The average lengths of the training corpus (5.25) and the generated sentences above (5.7) do not vary greatly, but the standard deviations—0.4 and 6.5, respectively—do. This is due to the presence of words that can appear in multiple positions in the sentence: 'awesome', 'language' and 'students'. As a result, a sentence can end after one word, or can restart when one would expect it to end.

1 Question 2. Word vectors

Consider the following two-dimensional word vectors that encode animals. They are projections of 300-dimensional word vectors, that were trained using data from all over the web.

$$\overrightarrow{\text{dog}} = \begin{bmatrix} 1.9 \\ 2.07 \end{bmatrix}, \overrightarrow{\text{puppy}} = \begin{bmatrix} 2.04 \\ 1.82 \end{bmatrix}, \overrightarrow{\text{puppies}} = \begin{bmatrix} 2.15 \\ 1.77 \end{bmatrix}, \overrightarrow{\text{cat}} = \begin{bmatrix} 1.63 \\ 1.87 \end{bmatrix}$$
$$\overrightarrow{\text{kitten}} = \begin{bmatrix} 1.68 \\ 1.76 \end{bmatrix}, \overrightarrow{\text{snake}} = \begin{bmatrix} 0.25 \\ 1.44 \end{bmatrix}, \overrightarrow{\text{monkey}} = \begin{bmatrix} 0.02 \\ 1.56 \end{bmatrix}, \overrightarrow{\text{monkeys}} = \begin{bmatrix} -0.04 \\ 1.59 \end{bmatrix}$$



1. Visually inspect the word vectors. Distances in the vector space capture word similarities (albeit strongly simplified when using only two dimensions). Why would 'snake' (a reptile) be closer to 'monkey' (a mammal) compared to the remaining animals (that are also mammals)? Please refer to the distributional hypothesis and the way word vectors are constructed in your answer.

   Solution According to the distributional hypothesis words that occur in similar contexts tend to have similar meanings. This is reflected in the point in the vector space we associate with a word: other points in its neighbourhood will occur in similar contexts. The contexts come from the training corpus. For arbitrary content from the web concerning animals, we expect pets like dogs and cats to appear in similar contexts. The same might hold for wildlife like a monkey and a snake, but it seems less likely that a cat and a snake appear in similar contexts.

The question suggests the vectors could have reflected the mammal vs. reptile difference, but this difference is not as prominent from text as the very different habitats of pets and wildlife. Were the vectors trained on a specialised corpus about biological evolution or the anatomy of animals, they probably *would* reflect that difference more.

2. Word vectors can capture relationships between words, as discussed in J&M Section 6.10, 3rd edition. An example of such a relationship is an analogy, such as Edinburgh is to Scotland as Canberra is to . . . ? When we have word vectors, we can use the parallelogram method to solve analogies: by subtracting $\overrightarrow{\text{edinburgh}}$ from $\overrightarrow{\text{scotland}}$ and adding $\overrightarrow{\text{canberra}}$. You would pick the word for which the word vector has the smallest distance to the resulting vector. J&M represent the procedure as follows for the analogy a:b::a*:b* (a is to b as a* is to b*):

$$\vec{b}* = \text{argmin distance}_{\vec{x}}(\vec{x}, \vec{b} - \vec{a} + \vec{a}*) \tag{1}$$

Using Euclidean distances, compute $\vec{b}*$ for:

- $\vec{a} = \overrightarrow{\text{dog}}$, $\vec{b} = \overrightarrow{\text{puppy}}$ and $\vec{a}* = \overrightarrow{\text{cat}}$
- $\vec{a} = \overrightarrow{\text{puppy}}$, $\vec{b} = \overrightarrow{\text{puppies}}$ and $\vec{a}* = \overrightarrow{\text{monkey}}$

Solution

- $\vec{b} - \vec{a} + \vec{a}* = \begin{bmatrix}2.04\\1.82\end{bmatrix} - \begin{bmatrix}1.9\\2.07\end{bmatrix} + \begin{bmatrix}1.63\\1.87\end{bmatrix} = \begin{bmatrix}1.77\\1.62\end{bmatrix}$

word	**distance**
dog	$\sqrt{(1.9-1.77)^2+(2.07-1.62)^2} \approx 0.47$
puppy	$\sqrt{(2.04-1.77)^2+(1.82-1.62)^2} \approx 0.34$
puppies	$\sqrt{(2.15-1.77)^2+(1.77-1.62)^2} \approx 0.41$
cat	$\sqrt{(1.63-1.77)^2+(1.87-1.62)^2} \approx 0.29$
kitten	$\sqrt{(1.68-1.77)^2+(1.76-1.62)^2} \approx 0.17$
snake	$\sqrt{(0.25-1.77)^2+(1.44-1.62)^2} \approx 1.53$
monkey	$\sqrt{(0.02-1.77)^2+(1.56-1.62)^2} \approx 1.75$
monkeys	$\sqrt{(-0.04-1.77)^2+(1.59-1.62)^2} \approx 1.81$

The word vector for 'kitten' has the minimum distance, hence the vectors suggest dog:puppy::cat:kitten, which makes perfect sense!

- $\vec{b} - \vec{a} + \vec{a}* = \begin{bmatrix}2.15\\1.77\end{bmatrix} - \begin{bmatrix}2.04\\1.82\end{bmatrix} + \begin{bmatrix}0.02\\1.56\end{bmatrix} = \begin{bmatrix}0.13\\1.51\end{bmatrix}$

word	**distance**
dog	$\sqrt{(1.9-0.13)^2+(2.07-1.51)^2} \approx 1.85$
puppy	$\sqrt{(2.04-0.13)^2+(1.82-1.51)^2} \approx 1.93$
puppies	$\sqrt{(2.15-0.13)^2+(1.77-1.51)^2} \approx 2.04$
cat	$\sqrt{(1.63-0.13)^2+(1.87-1.51)^2} \approx 1.54$
kitten	$\sqrt{(1.68-0.13)^2+(1.76-1.51)^2} \approx 1.57$
snake	$\sqrt{(0.25-0.13)^2+(1.44-1.51)^2} \approx 0.14$
monkey	$\sqrt{(0.02-0.13)^2+(1.56-1.51)^2} \approx 0.12$
monkeys	$\sqrt{(-0.04-0.13)^2+(1.59-1.51)^2} \approx 0.19$

Dependent on whether we exclude a, b and a* as potential answers or not, 'monkey' or 'snake' is closest, which, unfortunately, is not 'monkeys'. The vector $\overrightarrow{\text{monkeys}}$ is close to $\overrightarrow{\text{monkey}}$ in the vector space, but the relationship is not analogical to the one between $\overrightarrow{\text{puppies}}$ and $\overrightarrow{\text{puppy}}$.

3. Retrieve the word that is the most similar to 'monkey' using cosine similarity, euclidean distance and the dot product. Where do the different metrics disagree, and how does this relate to their definition?

Solution We collected the similarities below. For both the Euclidean distance and the cosine similarity, the word most similar to 'monkey' is 'monkeys', but the dot product suggests 'dog' is the most similar to 'monkey'. This is due to the impact of the vectors' norm on the computation of the dot product (Equation 2), since there is no length normalisation.

$$dot(\vec{u}, \vec{v}) = \sum_{i=1}^{d} \vec{u}_i \vec{v}_i \tag{2}$$

word	**Euclidean dist.**	**cosine dist.**	**dot product**
monkey	0.00	0.00	2.43
dog	1.95	0.25	3.27
puppy	2.04	0.32	2.88
puppies	2.14	0.35	2.80
cat	1.64	0.24	2.95
kitten	1.67	0.27	2.78
snake	0.26	0.01	2.25
monkeys	0.07	0.00	2.48

4. Explain how Euclidean distance, the dot product and the cosine similarity are related.

Solution The cosine similarity represents the cosine of the angle between two input vectors, and is the dot product normalised by the norms of the two vectors (Equation 3). The relation between the cosine similarity and the Euclidean distance may not be clear directly. After all, two vectors in the same direction can be very distant as per the Euclidean distance, but still very similar according to the cosine similarity. However, for normalised vectors, they are related, as is illustrated in Equations 4 and 5. If $\vec{u}$ and $\vec{v}$ have length 1, $ED(\vec{u}, \vec{v})^2 = 2 - 2dot(\vec{u}, \vec{v})$.

$$cos\theta_{u,v} = \frac{dot(\vec{u}, \vec{v})}{\|\vec{u}\|\|\vec{v}\|} \tag{3}$$

$$ED(\vec{u}, \vec{v}) = \sqrt{\sum_{i=1}^{d} (\vec{v}_i - \vec{u}_i)^2} = \sqrt{\sum_{i=1}^{d} (\vec{v}_i \vec{v}_i - 2\vec{v}_i \vec{u}_i + \vec{u}_i \vec{u}_i)} = \sqrt{\sum_{i=1}^{d} \vec{v}_i \vec{v}_i + \sum_{i=1}^{d} \vec{u}_i \vec{u}_i - 2\sum_{i=1}^{d} \vec{v}_i \vec{u}_i} \tag{4}$$

$$ED(\vec{u}, \vec{v})^2 = \sum_{i=1}^{d} \vec{v}_i \vec{v}_i + \sum_{i=1}^{d} \vec{u}_i \vec{u}_i - 2\sum_{i=1}^{d} \vec{v}_i \vec{u}_i \tag{5}$$

Question 3. Model design

The next problem looks at how to apply neural models to a classical problem in NLP: part-of-speech (POS) tagging. POS tags are labels assigned to each word in a sentence to indicate their grammatical role in the sentence. For example:

$i =$	1	2	3	4	5	6	7	8	9	10
$x_i =$	Each	day	starts	with	one	or	two	lectures	by	researchers
$y_i =$	DT	NN	VBZ	IN	CD	CC	JJR	NNS	IN	NNS

Common categories are Nouns (N), Pronouns (PRP), Verbs (VB), Adjectives (JJ), Adverbs (RB), Prepositions (IN), Determiners (DT), Conjunctions (CC) and Interjections(UH). There are also more fine-grained categories: word *day* in the sentence is a singular noun (NN). However, the same word can have different PoS tags, depending on the context.

Given an input sentence $x = x_1 \dots x_{|x|}$, we want to predict the corresponding tag sequence $y = y_1 \dots y_{|x|}$. Let x_i denote the ith word of x, y_i denote the ith word of y, and $|x|$ denote the length of x. Note that $|y| = |x|$. We have access to many training examples like this, and our goal is to model the conditional probability of the tag sequence given the sentence, that is: $P(y \mid x)$. There are many possible choices here. To simplify the problem, let's *assume* that each element of y is conditionally independent of each other. That is, we want to model:

$$P(y \mid x) = \prod_{i=1}^{|y|} P(y_i \mid x)$$

1. Design a feedforward neural network to model $P(y_i \mid x)$: clearly define the probability being computed and how it is estimated, and specify the input and output vocabulary, as well as the objective function used for training. Identify any independence assumptions you make. Draw a diagram that illustrates how the model computes probabilities for the tag of the word "with": What is the input, and how is the output distribution computed from the input? Write out the basic equations of the model, and explain your choices.

   Solution An effective design for the feedforward network is to model $P(y_i \mid x_{i-k}, \dots, x_{i+k})$ for some fixed window size $2k + 1$. You might, for example, use something like this:

   $$\begin{aligned} P(y_i \mid x_{i-k}, \dots, x_{i+k}) &= \text{softmax}(\mathbf{W}\mathbf{h} + \mathbf{b}_2) \\ \mathbf{h} &= \tanh(\mathbf{V}\mathbf{x} + \mathbf{b}_1) \\ \mathbf{x} &= \text{onehot}(x_{i-k}); \dots; \text{onehot}(x_{i+k}) \end{aligned}$$

   Here, the semicolon (;) denotes concatenation. The choice of non-linearity is not important for this question, but since it asks for a feedforward network, you should have a hidden layer. This is about the simplest possible model.

   Note that your solution *should not* depend on previous tags, since the question explicitly assumes that the tags are conditionally independent.

2. Design an RNN to model $P(y_i \mid x)$: clearly define the probability being computed and how it is estimated, and specify the input and output vocabulary, as well as the objective function used for training. Identify any independence assumptions you make. Draw a diagram that illustrates how the model computes probabilities for the tag of the word "with": What is the input, and how is the output distribution computed from the input? Write out the basic equations of the model, and explain your choices.

   Solution One design for the RNN is to model $P(y_i \mid x_1, \dots, x_i)$. That is, the RNN reads x_1 through x_i one step at a time, and at the ith step produces a distribution for possible

tags y_i. For simplicity, let's use RNN to denote a unit that receives an input and a previous hidden state, and produces a new hidden state; it can easily be replaced with an LSTM or other recurrent unit of your choice:

$$\begin{aligned} P(y_i \mid x_1 \dots x_i) &= \text{softmax}(\mathbf{W}\mathbf{h}_i + \mathbf{b}) \\ \mathbf{h}_i &= \text{RNN}(\text{onehot}(x_i), \mathbf{h}_{i-1}) \end{aligned}$$

One thing you might notice here is that, while this model conditions on all words the left of x_i, it does not use *any* words to its right! Because of this, the feedforward network might have an advantage. Can you think of a reason why you should not use k words of right context in this model?

3. Can you model $P(y_i \mid x)$ without independence assumptions, using multiple RNNs?

   Solution A *bidirectional* RNN can model $P(y_i \mid x_1, \dots, x_{|x|})$. It consists of two RNNs, each with its own set of parameters: one that encodes from left to right (RNN), and one that encodes from right to left (RNN$'$). Again using the semicolon to denote concatenation:

$$\begin{aligned} P(y_i \mid x_1 \dots x_{|x|}) &= \text{softmax}(\mathbf{W}(\mathbf{h}_i; \mathbf{h}'_i) + \mathbf{b}) \\ \mathbf{h}_i &= \text{RNN}(\text{onehot}(x_i), \mathbf{h}_{i-1}) \\ \mathbf{h}'_i &= \text{RNN}'(\text{onehot}(x_i), \mathbf{h}'_{i+1}) \end{aligned}$$

For each question, the goal is to design a *simple* model for the distribution. You solution should only use architectures that we discussed in the first four weeks of the course. If you are aware of other architectures, you should not use them here.