Generalization: From Count-based to Neural Language Models

What happens if a sentence never occurs in the text that a count-based language model is trained on? Consider a sentence that is highly unlikely to occur in known text like the following:

```
A lion is chasing a llama.
```

It's rather unlikely for these two animals to interact in real life, since they exist on different continents, and even less likely that there is a piece of writing describing such an event. It is *conceivable* however, that a predator like a lion *could hypothetically* chase a domesticated animal like a llama. But a count-based language model will assign a zero probability if this *exact* sequence of words does not appear in the text it has been exposed to!

The inability to work with possibilities that do not appear in training data is because count-based models are unable to *generalize*. Generalization is the ability of a model to adapt to new and unseen data. In the context of language models generating text, it means that they can produce text that doesn't exist in its training data.

To get a language model to generalize, we would need to **move beyond counting words** and instead, find a way to link words to their meaning and context. This type of model uses **semantics** (i.e., what words mean) rather than counting (which words exist in text). For instance, a semantic language model might map "lions" to the idea of predators and "llamas" to cattle/prey/domesticated animals. This allows it to use the connection that predators chase prey and even assign a non-zero probability for lions chasing llamas. This is often known as a "semantic representation". The mathematical objects that allow language models to generalize this way are known as **word embeddings**.

Word embeddings allow us to turn each word into a series of numbers, also known as a word vector. These vectors exist in an abstract space where words are semantically linked, i.e., they're linked by the meaning and context in which they appear. This makes for a more sophisticated model. One example would be the way it treats homonyms, i.e., two words that have the same spelling but different meanings. For instance, the word "lead" is different when paired with the word "President" than it is when paired with "pipe". A language model using word embeddings would be able to distinguish between homonyms.

The most popular and efficient way to build such embeddings from text is using **neural networks** and language models built using neural networks are referred to as **neural language models**.

Note 1: "Generalizing" and "generative" are similar sounding words used in the context of language models that mean different things:

- "Generative" refers to the *ability to generate* data and this might be text/image/audio/video depending on the kind of model being built.
- "Generalizability" refers to the *ability to generalize* , i.e., the ability to adapt to or produce unseen data.

Note 2:

- The phrase "non-zero probability" means "it is possible that…"
- The phrase "zero probability" means "it is impossible that…"

Instructions

This example, illustration and explanation is inspired by AI researcher (and New York University professor) **Prof. Kyunghun Cho**'s excellent talk on *"A slight-less-magical perspective into autoregressive language modeling: count, compress and prune"*. You can check out more talks and amazing material on language models and text generation on his webpage here!

Now, if we perform the kind of probabilistic calculations that we did in the previous exercise for the sentence "The lion is chasing a llama", it might look something like the image shown to the right. Do the probabilities shown in the image seem reasonable to you?

Counting Words:

P (“a”)	=	P (”a”)	= .4
P (”a lion”)	=	P (”a”)	= .4
		* P(”lion” | “a”)	* .05
P (”a lion is ”)	=	P (”a”)	= .4
		* P (”lion” | “a”)	* .05
		* P (”is” | “a lion”)	* .3
P (”a lion is chasing”)	=	P (”a”)	= .4
		* P (”lion” | “a”)	* .05
		* P (”is” | “a lion”)	* .3
		* P (”chasing” | “a lion is”)	* .6
P (”a lion is chasing a”)	=	P (”a”)	= .4
		* P (”lion” | “a”)	* .05
		* P (”is” | “a lion”)	* .3
		* P (”chasing” | “a lion is”)	* .6
		* P (”a” | “a lion is chasing”)	* .75
P (”a lion is chasing a llama”)	=	P (”a”)	= .4
		* P (”lion” | “a”)	* .05
		* P (”is” | “a lion”)	* .3
		* P (”chasing” | “a lion is”)	* .6
		* P (”a” | “a lion is chasing”)	* .75
		* P(”llama” | “a lion is chasing a”)	* .0
			= 0!

Word Embeddings: