# Natural language descriptions for natural language directions

Welcome to this document. It is intended as a resource to improve intuition and understanding of the embedding space used by transformers. Feel free to make your own copy or just add your analyses to this one!
Public link: https://docs.google.com/spreadsheets/d/1AonbHnUlWxloGbawgHz6EoFCeO-KRCOh9VF9R4LqhjU/edit?usp=sharing

Introduction
For most transformer architectures, the input is tokenised in some way, and then each token is linearly embedded in a high-dimensional vectorspace.
To get back from that vectorspace to some representation of tokens, a vector is unembedded again through a linear function, sometimes simply the transpose of the embedding matrix.

In those cases, it is highly likely that semantic meaning of embedded directions is conserved throughout the model.
If we had some natural language description of those directions, it would be much easier to give natural-language descriptions of what parts of the model are doing

One problem: since everything interacts linearly with the embedding space, adding an arbitrary rotation to it and updating the weights would give the exact same model.
So we might ask what it even means to have special directions that mean something when there is an arbitrary rotation we can perform without changing functionality.
And even if there are any, how would we find them?

One thing you could ask is: if I take a random token, what is the direction along which I have the most uncertainty for where it will land? Let's call that our first direction.
This is the direction that our network dedicates the highest resolution to, in a sense it is the direction it deems the most important.
We can then ask: If we take away that direction, what is the new most important direction in the remaining space? This way we get a series of orthogonal directions in decreasing order of "importance".
This is what the singular value decomposition of a matrix calculates, so let's just do that and see what comes out!

General thoughts, motivations:
When a network embeds a token, it has to be able to unembed it again. Giving the same task to a human would be like saying:
"I am thinking of a word or part of a word. You can now decide on 700 questions that I have to answer to determine that word. Also, spelling and pronunciation don't exist, you have to find it through its meaning."
Then, there are a bunch of questions that it intuitively makes sense to ask, and we'd strongly expect to find a lot of those represented in the embedding.
Also, any computation will cut into this space, so it'll share some space with the semantic meanings and give rise to new meanings like "this word is the third in the sentence".
All of this is more a property of the information theory of a language than a property of some specific transformer, so there'll be overlap between transformers that look at the same language.

At this point you might want to look at some of the other sheets. The code used to generate them can be found here: https://colab.research.google.com/drive/1nMNBvET7NORtG65ldOfbXoMBxs39CVjj?usp=sharing
Most of the analysis I did was on gpt2, feel free to look at any of them and add your analysis

Stuff to do from here:

Currently doing
- get distribution of token along dimensions to differentiate between binary and continuous dimensions, if possible condense this to a single metric of binaryness.

General further stuff
- find more semantic meanings
- do this for other transformers
- singular values that are close together can get rotated into each other (if you have directions x and y with the same singular value, you could also get sqrt(2)x+y and sqrt(2)x-y and singular vectors), but for two or three of those, our understanding of natural language should give us the ability to pry those apart. Simple example: directions 29 and 30 of gpt-2
- write something that tells you the semantics for some token (basically reading out the transpose of the right singular matrix we're looking at here)
- might be nice to refine our svd on not just sampling a token uniformly but from the training token distribution, but that needs training token distribution data

Applications
- go through the first few layers of the transformer, match the linear functions to semantic directions in embedding space, get natural language descriptions of parts of the transformer
- talk to a linguist
- after looking at the first few layers, get updated descriptions for directions that don't just correspond to semantics of this particular token but also larger sentence structure
- currently used tokenisations are a mess. Go through this to find something better