# Text auto encoding

## General plan

Build an auto encoder for text. You can use the Keras guide. You should change the input/output representation, and experiment with results.

* Keras auto encoder guide: https://blog.keras.io/building-autoencoders-in-keras.html
* One-hot encoding guide: https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/

The test and train set are the same. The size of the input layer should be the size of your vocabulary, and also the size of the output layer. Use the NLTK movie review corpus, or any other data you like.

## Experiments

1. Try with one-hot encoding of words: i.e. one dimension per word, and just one word per example.
2. Try with one-hot encoding of words with sentences: i.e. one dimension per word, but represent a whole sentence with a few 1s
3. Try a one-hot encoding with bigrams: the input vector might have length |V|^2 (it's large).
    1. It'll be helpful to have a marker for start / end.
    2. Can you recover a sentence from the output bigrams?
    3. You might like to discard uninformative bigrams to start with (e.g. those with frequency 0 in the data), to reduce the vector size.

## Extensions

Can you recover words based on their characters? Try making the input dimensions equal to the character bigrams seen in the vocabulary, and then representing words based on their character bigrams. E.g. "cheese" becomes _START_c, ch, he, ee, es, se, e_END_. Can this technique work OK with very low dimension counts?

## Recommendations

Use GPU instances.

A big input layer is slow to process, so consider only including frequent items.

Consider an encoding dimension of 32 to start with. You might be able to go very low, to e.g. 10, or you might not get good results until you reach 100.