We propose a version of a sequence-to-sequence encoder-decoder model to create semantically coherent *statement embeddings*. The model architecture is depicted in figure __.

Initial Statement Representation
We treat tokens (tokens are discussed further in *Data Generalization*) as the atomic units of each statement—which we restrict to a maximum character-length. Each token is first encoded into a *token vector* using Fasttext [1]. Fasttext is chosen to create token vectors because of its ability capture morphological similarities between tokens. This feature is particularly useful in dealing with slang, typos, and out-of-vocabulary-words.

Encoding
The token vectors are combined to form a variable-sized token matrix. Following other sequence-to-sequence learning models [2], we then feed the token matrix to an encoding LSTM. The hidden state of the final unit of the encoding LSTM is used as the statement embedding.

Decoding and Training
We apply 2 (maybe more?) layers of LSTMs to decode the statement embedding. The statement embedding is used as the hidden state of the first unit in the decoding LSTM sequence. The output of each unit in the last decoding LSTM is fed through a softmax layer to generate the model's ultimate output—which takes the form of a number of 70-dimensional vectors equal to the maximum character length Each vector represents a probability distribution over the set of possible Unicode characters. The target that the output is compared to is a generalized form of the input statement. This target is represented by a sequence of one-hot, 70-dimensional vectors for each character. To train the model we minimize the sum of the cross-entropy loss between each pair of outputted character-vectors and target character vectors.

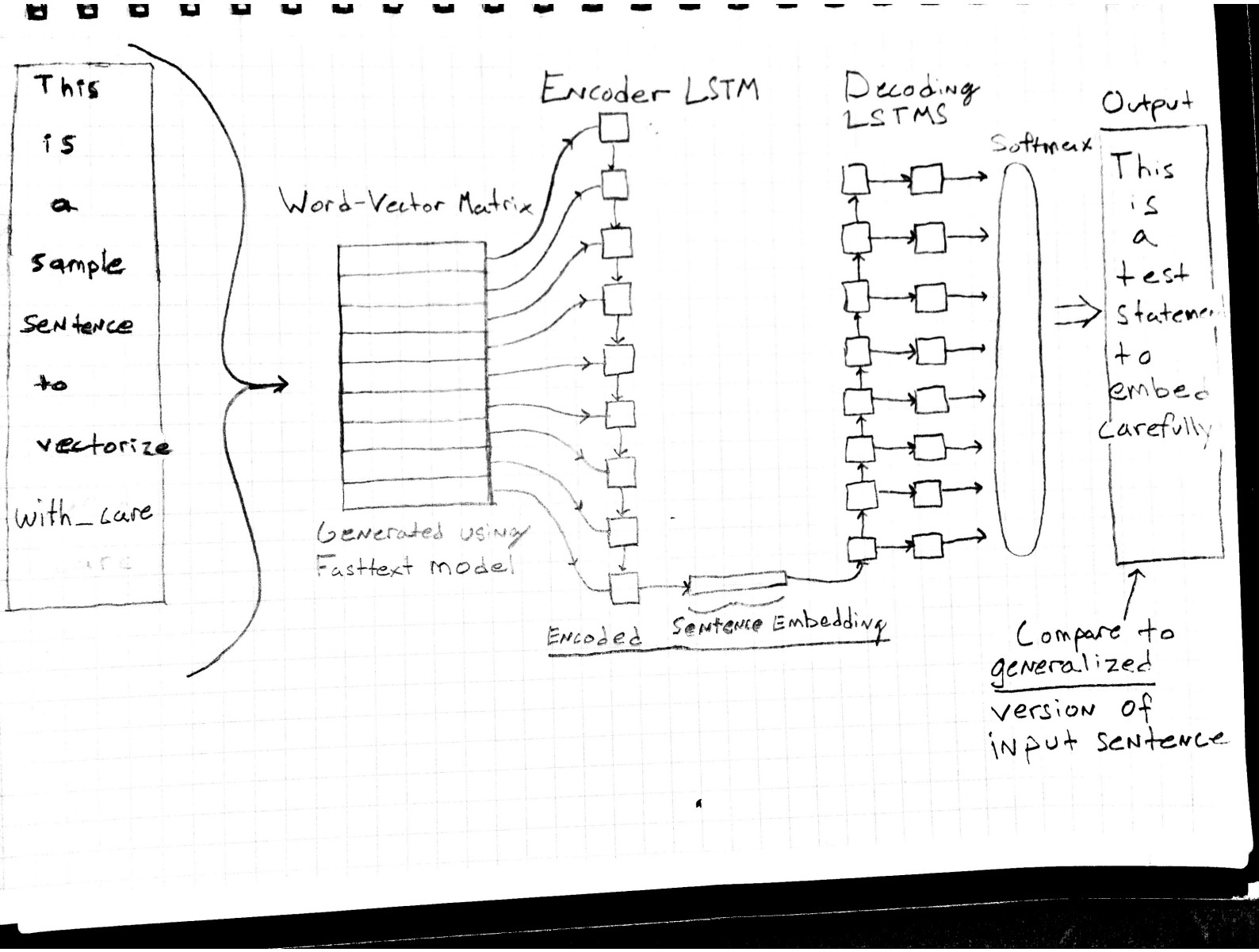
Data Augmentation
A problem with embedding models similar to the one described here is that while they are effective at encoding similarity between similarly worded sentences, they are prone to miss the similarities between semantically similar but structurally dissimilar sentences. To address this problem we probabilistically generalize the target statements in the following way:

As a preprocessing step we apply a phrase-detecting model to the corpus of training data. On the first run, this will identify bigrams that appear together relatively often compared to the total occurrence of each word in the corpus. For instance, the model may find that *plausible deniability* is best represented by *plausible_deniability.* To identify phrases of longer length, we can simply apply the phrase-detection model again to identify phrases such as *new_york_times*. We can repeat this process to create coherent tokens of arbitrary word-length. Once we have collapsed

common phrases of arbitrary length to single tokens, we then embed each token with Fasttext.

This preprocessing step allows us to make bolder generalizations of the target sentences by replacing the now potentially long tokens with synonyms, as defined by the cosign similarity of the tokens' embeddings.

[1] P. Bojanowski*, E. Grave*, A. Joulin, T. Mikolov, *Enriching Word Vectors with Subword Information*
[2] **arXiv:1409.3215**
[3] Soroush Vosoughi, Prashanth Vijayaraghavan and Deb Roy. (2016). Tweet2Vec: Learning Tweet Embeddings using Character-level CNN-LSTM Encoder-Decoder. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016). Pisa, Italy.
[4] https://radimrehurek.com/gensim/models/phrases.html


This is a sample sentence to vectorize with_care
Word-Vector Matrix
Generated using Fasttext model
Encoder LSTM
Encoded
Sentence Embedding
Decoding LSTMS
Softmax
Output
This is a test statement to embed carefully
Compare to generalized version of input sentence