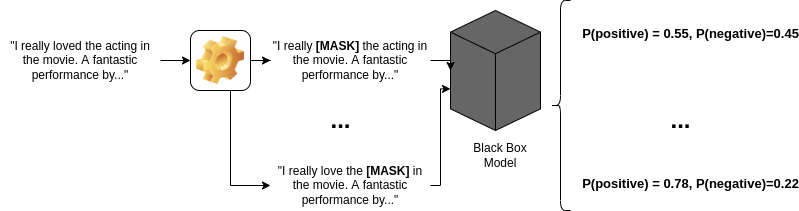
Model-Agnostic

LIME’s model agnosticism is one of its most useful attributes. As long as you know how to encode the input data and your model has the ability to provide probabality distributions over its outputs, you can provide local explanations for any type of model. This is because the explanation comes from the local model and the BoW features therein rather than the black box model.

In the section below I’ve provided some examples of how to use ELI5 with some different types of models.

Some machine learning models like [linear models](https://scikit-learn.org/stable/modules/linear_model.html) and [Decision Trees](https://scikit-learn.org/stable/modules/tree.html) are inherently interpretable through being able to measure parameter coefficients (how big the weight of the feature is when calculating the decision boundary line) in the case of the former and how early on a feature appears in a decision tree (since decision trees use [information gain](https://en.wikipedia.org/wiki/Information_gain_in_decision_trees) to put features that tell us most about the final classification/decision near the top of the tree so that they impact more data points) in the case of the latter.

LIME exploits these explainable models in order to explain the local context around a given input example. We perturb (slightly change) the input example and use the black-box model under analysis to make predictions. As words are added or removed from the input, the output from the black box model changes slightly (in the [contrived again] example below, removing the word ’love’ from the movie review reduces the probability that the review is positive.)



LIME perturbs input examples by changing words around in order to understand the individual contributions of words to an outcome

These perturbed inputs and the outputs from the ‘black box’ model that we’re analysing outputs are then used as a training set to train the local, interpretable model.

For text models, LIME uses [Bag-of-Words](https://en.wikipedia.org/wiki/Bag-of-words_model) (BoW) representations of the perturbed input as the features for the local model.

We can then use the interpretable information (parameter coefficients/feature position in decision tree) for the local model to approximately interpret the effect that the different words have on the bigger model since each word in the local BoW vocabulary will have an associated coefficient.

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form (From [Wikipedia](https://en.wikipedia.org/wiki/Stemming))

For example, if there are two words in the corpus `walks` and `walking`, then stemming will stem the suffix to make them `walk`. But say in another example, we have two words `console` and `consoling`, the stemmer will remove the suffix and make them `consol` which is not a proper english word.

Lemmatization is similar to stemming in reducing inflected words to their word stem but differs in the way that it makes sure the root word (also called as lemma) belongs to the language.

As a result, this one is generally slower than stemming process. So depending on the speed requirement, we can choose to use either stemming or lemmatization.

Wow. It returned `running` as such without converting it to the root form `run`. This is because the lemmatization process depends on the POS tag to come up with the correct lemma. Now let us lemmatize again by providing the POS tag for the word.

Now we are getting the root form `run`. So we also need to provide the POS tag of the word along with the word for lemmatizer in nltk. Depending on the POS, the lemmatizer may return different results.

Let us take the example, `stripes` and check the lemma when it is both verb and noun.