**Galytix DE Assessment**

**Problem Statement**

Imagine the Data Science team submitted word embeddings algorithm to encode words as real number vectors. You are now integrating these embeddings into a data processing pipeline and applying them to calculate semantic distance between phrases in English.

We will use pretrained Word2Vec vectors as a proxy for the output from our Data Science team. (Word2Vec algorithm is a well-known standard method and a stepping stone towards many modern methods in natural language processing).

**Tasks**

1. **Init pipeline**

Download the pretrained set of Word2Vec vectors from <https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM>

Load the word embeddings for first million vectors from their binary form using gensim library and then store them as flat file and then continue working with the flat file. See the snippet below:

import gensim; from gensim.models import KeyedVectors; wv = KeyedVectors.load\_word2vec\_format(location, binary=True, limit=1000000) ; wv.save\_word2vec\_format('vectors.csv')

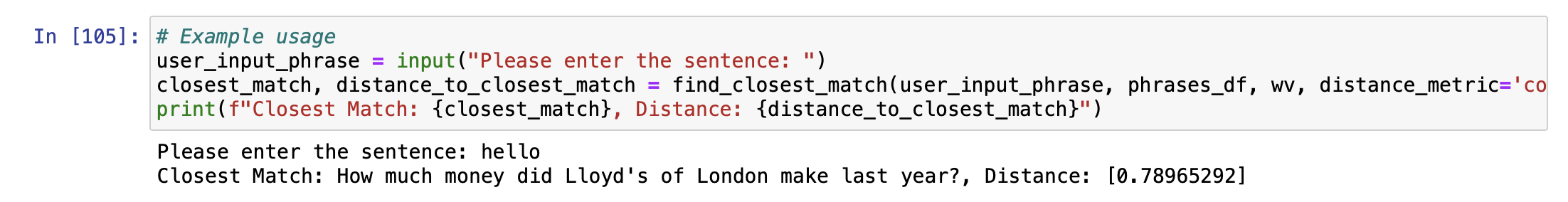
1. **Process data**

Calculate similarity of phrases in phrases.csv with each other:

1. Assign each word in each phrase a Word2Vec embedding.
2. Batch execution: Calculate L2 distance (Euclidean distance) or Cosine distance of each phrase to all other phrases and store results. Try to achieve this in a manner that is not compute or memory wasteful. Note that the whole phrase vector can be approximated by normalized sum of all the individual word tokens embeddings.
3. On the fly execution: Create a function that takes any string, e.g. user-input phrase, and finds and return the closest match from phrases in phrases.csv and the distance
4. **Turn it into an app**

Structure your code into modules. Use OOP programming principles. Prepare setup.py or project.toml, prepare pip or conda environment. Initialize logging. Add some error handling and argument validations.

**Output**

****