Note: My original plan for Mini-Project 3 included 4 tasks (0 to 3) and was meant to be due Monday December 6. However, I saw that Mini-Project 2 was mostly done at the last minute, so I have toned down MP3 to include only 3 tasks (0 to 2), and you will be able to submit in until December 13 without penalty.

Due Dates Task 0: November 27, 2021
Tasks 1&2: Monday December 6, 2021 but will be accepted until December 13, without penalty.

Late Submissions 30% per day per late deliverable
Teams You can do the mini-project in teams of at most 3.
Teams must submit only 1 copy of the project via the team leader's account.

Experiments with Word Embeddings

In this project, you will experiment with word embeddings to solve a Synonym test automatically and see if your program does better than humans! You will then compare the performance of different models on different data sets and analyse these results to draw insight.

1 The Dataset

Download the Synonym Test dataset available on Moodle. This dataset contains 80 English words and 4 choices of synonyms for each word. The task is to select the closest synonym out of these 4 choices. For example, the first entry of the data set contains:

`1.`	`enormously`	⟵ the question-word
`a.`	`appropriately`	⟵ guess-word-1
`b.`	`uniquely`	⟵ guess-word-2
`c.`	`tremendously`	⟵ guess-word-3
`d.`	`decidedly`	⟵ guess-word-4
`c`		

The above indicates that for the question-word `enormously`, out of the 4 guess-words `appropriately, uniquely, tremendously, decidedly`, the closest in meaning is choice `c` i.e. `tremendously`.

2 Your Tasks

Write a program to use different word embeddings to answer the Synonym Test automatically and compare the performance of different models. You must use:

1. Python 3.8 and the `Gensim` library. `Gensim` https://radimrehurek.com/gensim/ is a free open-source Python library for representing documents as vectors. The library allows you to load pre-trained word embeddings, train your own Word2Vec embeddings from your own corpus, computes the similarity of word embeddings, ...

2. GitHub (make sure your project is private while developing).

Your work will be divided into ~~4~~ 3 tasks.

2.1 Task 0: Contribution to a Collective Human Gold-Standard

This task is to be done individually. In order to better judge the performance of your code (tasks 1 to 3 below), each of you will do the Synonym Test manually.

Before November 27, 2021 (11:59pm), go on the Moodle Quiz called `Crowdsourced Gold-Standard for MP3` and answer each question to the best of your ability. Task 0 will count for 4% of your MP3 grade. You will be graded on participation only, not on the correctness of your answers. Your grade will be proportional to the number of questions that you answer.

After November 27, 2021, I will compile all the results and publish them on Moodle so that all teams can use them as a human gold-standard to evaluate the results of your code (tasks 1 to 3 below).

2.2 Task 1: Evaluation of the `word2vec-google-news-300` Pre-trained Model

In this first experiment, you will use the pre-trained Word2Vec model called `word2vec-google-news-300` to compute the closest synonym for each word in the dataset. First, use `gensim.downloader.load` to load the `word2vec-google-news-300` pretrained embedding model. Then use the `similarity` method from `Gensim` to compute the cosine similarity between 2 embeddings (2 vectors) and find the closest synonym to the question-word.

The output of this task should be stored in 2 files:

1. In a file called `<model_name>-details.csv`, for each question in the Synonym Test dataset, in a single line:
   (a) the question-word, a comma,
   (b) the correct answer-word, a comma
   (c) your system's guess-word, a comma
   (d) one of 3 possible labels:
   - the label `guess`, if either question-word or all four guess-words (or all 5 words) were not found in the embedding model (so if the question-word was present in the model, and at least 1 guess-word was present also, you should not use this label).
   - the label `correct`, if the question-word and at least 1 guess-word were present in the model, and the guess-word was correct.
   - the label `wrong` if the question-word and at least 1 guess-word were present in the model, and the guess-word was not correct.

   For example, the file `word2vec-google-news-300-details.csv` could contain:

```
enormously,tremendously,uniquely,wrong
provisions,stipulations,stipulations,correct
...
```

2. In a file called `analysis.csv`, in a single line:
   (a) the model name (clearly indicating the source of the corpus and the vector size), a comma
   (b) the size of the vocabulary (the number of unique words in the corpus[1]), a comma
   (c) the number of `correct` labels (call this C), a comma
   (d) the number of questions that your model answered without guessing (i.e. $80-$ `guess`) (call this V), a comma
   (e) the accuracy of the model (i.e. $\frac{C}{V}$)

   For example, the file `analysis.csv` could contain:

```
word2vec-google-news-300,3000000,44,78,0.5641025641025641
```

[1] also known as the word-types, as opposed to work-tokens

2.3 Task 2: Comparison with Other Pre-trained Models

Now that you have obtained results with the `word2vec-google-news-300` pre-trained model, you will experiment with 4 other English word2vec pretrained models and compare the results. You can use any pre-trained embeddings that you want, but you must have:

1. 2 new models from different corpora (eg. Twitter, English Wikipedia Dump ...) but same embedding size (eg. 25, 100, 300)
2. 2 new models from the same corpus but different embedding sizes

Many pre-trained embeddings are available on line (including in Gensim or at http://vectors.nlpl.eu/repository).

For each model that you use, create a new `<model_name>-details.csv` output file and append the results to the file `analysis.csv` (see Section 2.2). For example, the file `analysis.csv` could now contain:

```
word2vec-google-news-300,3000000,44,78,0.5641025641025641 // from Task 1
C1-E1,...,...,...,...
C2-E2,...,...,...,...
C3-E3,...,...,...,...
C4-E4,...,...,...,...
```

where `C1` to `C4` refer to the corpora and `E1` to `E4` refer to their embedding sizes, and `C1` $\neq$ `C2` and `E1` $=$ `E2` and `C3` $\neq$ `C4` and `E3` $=$ `E4`.

Compare the performance of these models (graphs would be very useful here) and compare them to a random baseline and a human gold-standard. Analyse your data points and speculate on why some model perform better than others.

2.4 Task 3: Train your Own Models

Now, you should be ready to train your own Word2Vec model[2].

Go to the Project Gutenberg (https://www.gutenberg.org/) and download at least 5 online books.

Preprocess these books to generate a list of sentences (i.e. a list of list of words). You may find `nltk`[3] useful for pre-processing your corpus as it provides a variety of functions to tokenize and split natural language texts (eg. `sent_tokenize`).

Use Gensim to create word2vec embeddings from your list of sentences. For this, use `gensim.models.Word2Vec`. Important parameters to pass include:

The list of sentences: use your pre-processed books

Window size: try 2 different values `W1` and `W2`

Embedding Size: try 2 different values `E5` and `E6`

This will therefore lead to 4 new models. By default, `gensim.models.Word2Vec` uses the CBOW method that we saw in class, so keep that default value.

For each model that you use, create a new `<model_name>_details.csv` output file and append the results to the file `analysis.csv` (see Section 2.2). For example, the file `analysis.csv` could now contain:

```
word2vec-google-news-300, 44,78,0.5641025641025641   // from Task 1
C1-E1,...,...,...,...                                // from Task 2
C2-E2,...,...,...,...                                // from Task 2
C3-E3,...,...,...,...                                // from Task 2
C4-E4,...,...,...,...                                // from Task 2
your_own_corpus-E5-W1,...,...,...,...
your_own_corpus-E5-W2,...,...,...,...
your_own_corpus-E6-W1,...,...,...,...
your_own_corpus-E6-W2,...,...,...,...
```

where `E5` $\neq$ `E6` and `W1` $\neq$ `W2`.

Compare the performance of these models (graphs would be very useful here). Analyse your data points and speculate on why some model perform better than others.

[2] Woohoo!

[3] Natural Language Toolkit for Python, see https://www.nltk.org/

3 Deliverables

The submission of the mini-project will consist of 3 deliverables:

1. Your individual contribution to the collective human Gold-standard (see the Moodle Quiz called `Crowdsourced Gold-Standard for MP3)`. Note that the deadline for completing the Quiz is November 27, 2021.
2. The code & output files
3. The demo (8 min presentation & Q/A)

3.1 The Code & Output files

Submit all files necessary to run your code in addition to a `readme.md` which will contain specific and complete instructions on how to run your experiments. You do not need to submit the datasets. If the instructions in your readme file do not work, are incomplete or a file is missing, you will not be given the benefit of the doubt.

Generate one output file for each model and a single analysis file for all models. This means 9 files named `<model_name>-details.csv` and 1 `analysis.csv`.

3.2 The Demos

You will have to demo your mini-project for $\approx$ 12 minutes. Regardless of the demo time, you will demo the program that was uploaded as the official submission. The schedule of the demos will be posted on Moodle. The demos will consist of 2 parts: a presentation $\approx$ 8 minutes and a Q/A part ($\approx$ 4 minutes). Note that the demos will be recorded.

3.2.1 The Presentation

Prepare an 8-minute presentation to analyse and compare the performance of your models, and to discuss possible improvements over your best model. The intended audience of your presentation is your TAs. Hence there is no need to explain the theory behind the models. Your presentation should focus on **your** work and the comparison of the performance of the different models.

Your presentation should contain at least the following:

- ☐ An analysis of the results of all models. In particular, compare and contrast the performance of each model with one another, and against a random baseline and a human gold-standard.
- ☐ One or two concrete ideas on how to improve your best model further.
- ☐ In the case of team work, a description of the responsibilities and contributions of each team member.

Any material used for the presentation (slides, ...) must be uploaded on EAS before the due date.

3.2.2 Q/A

After your presentation, your TA will proceed with a $\approx$ 4 minute question period. Each student will be asked questions on the code/mini-project, and he/she will be required to answer the TA satisfactorily. In particular, each member should know what each parameter that you experimented with represent and their effect on the performance. Hence every member of team is expected to attend the demo.

In addition, your TA may give ask you to change your code and ask you to run it again. The output files generated at demo time will have to be uploaded on EAS during your demo.

4 Evaluation Scheme

Students in teams can be assigned different grades based on their individual contribution to project.

Individual grades will count for 20% and will be based on:

(5%) your individual contribution to the collective human Gold-standard (see the Moodle Quiz called `Crowdsourced Gold-Standard for MP3`). Note that the deadline for completing the quiz is November 27, 2021.

(10%) a peer-evaluation done after the submission & the contribution of each student as indicated on GitHub.

(5%) the Q/A of each student during the demo.

The team grade will count for 80% and will be based on:

(45%) Code: functionality, design, programming style, ...

(15%) Demo: Presentation & depth of the analysis, clarity and conciseness, presentation, time-management, ...

(10%) Output with initial datasets: correctness and format

(10%) Output with demo-dataset: correctness and format

5 Submission

If you work in a team, identify one member as the team leader. The only additional responsibility of the team leader is to upload all required files (including the files at the demo) from her/his account and book the demo on the Moodle scheduler. If you work individually, by definition, you are the team leader of your one-person team.

5.1 Submission Schedule

Each deliverable is due on the date indicated below.

Deliverable	**Due Date**	**Submit as**
Task 0	November 27, 2021, 11:59pm	Moodle Quiz
Submit your code, output files, presentation material	December 13, 2021, 11:59pm	EAS as ssignment 3
Submit the output files generated at demo time	during your demo	EAS as Assignment 30

5.2 Submission Checklist

In your GitHub project, include a `README.md` file that contains:

1. on its first line: the URL of your GitHub repository,
2. specific and complete instructions on how to run your program.

Code & Output files

- ☐ Create one zip file containing all your code, the output files for the initial test set on Moodle and the `README.md` file.
- ☐ Name your zip file: `472_Assignment3_ID1_ID2_ID3.zip` where `ID1` is the ID of the team leader.
- ☐ Have the team leader upload the zip file at: https://fis.encs.concordia.ca/eas/ as `Assignment3`.

Demo & Outut files During your actual demo with the TA:

- ☐ Prior to your demo, make your GitHub repository public.
- ☐ Generate the output files for the test set that the TA will give you.
- ☐ Create a zip file called: `472_Demo1_ID1_ID2_ID3` where `ID1` is the ID of the team leader.
- ☐ Have the team leader upload the zip file at: https://fis.encs.concordia.ca/eas/ as `Assignment3`.

Have fun!