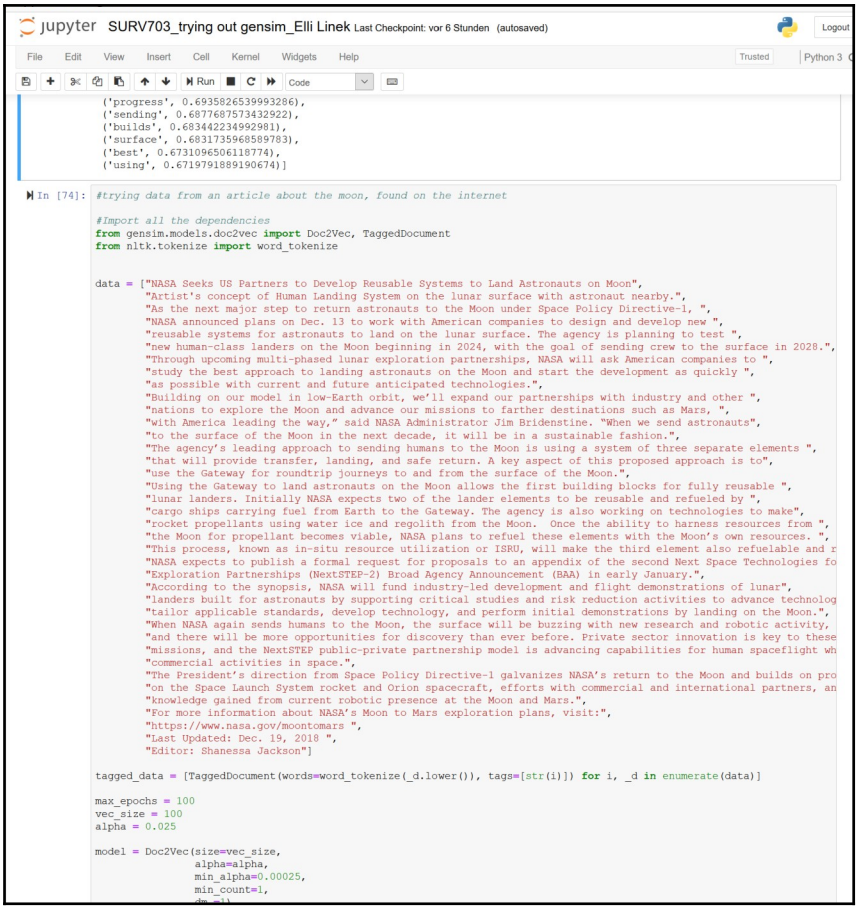
SURV703 – Content Analysis
Installing GENSIM, implementing word2vec embedding model,
briefly explaining the results
Elisabeth Linek

Task: Using gensim (Links to an external site.), train an LDA topic model (Links to an external site.) or a word2vec embedding model (Links to an external site.) on a corpus of your choice. Interpret the results in a short text of variable length (no more than 500 words).

Starting off with the installation of gensim. The first problem I encountered was, the impossibility to define pdf-files as corpus. So I setted up a short .txt-file, which I was able to load into the program, but encountered later on problems with the set up of the word2vector mode. So is started a completely new trial.

Aiming to test the word embedding possibilities of gensim I changed plans and started with the import of a short text about the moon, that I found on the internet – a short text, not at all complex in order to gain overview and having the possibility to assess the results.

Here is a screenshot out of my jupyter notebook, showing parts of the text, I entered in order to train and test the program:



I run 100 iterations, in order to train the model. Based on the then established word to vector model I tested different word-similarities. In a first attempt I asked for similarities regarding the word or term "moon", and received the following matrix of similar terms:

```
    model.train(documents, total_examples=len(documents), epochs=10)
    train(...)
```

```
In [12]: word = "moon"
         model.wv.most_similar (positive = word)
```

```
C:\Users\elli\Anaconda3\lib\site-packages\gensim\matutils.py:737: FutureWarning: Conversion of the second argument of issubdtype from `int` to `np.signedinteger` is deprecated. In future, it will be treated as `np.int32 == np.dtype(int).type`.
  if np.issubdtype(vec.dtype, np.int):
```

```
Out[12]: [('the', 0.847488284111023),
          ('approach', 0.7347332239151001),
          ('humans', 0.7176517248153687),
          ('to', 0.7012702226638794),
          ('progress', 0.6983550786972046),
          ('builds', 0.694743275642395),
          ('sending', 0.6897841095924377),
          ('best', 0.6805385947227478),
          ('using', 0.6788151264190674),
          ('surface', 0.6768609285354614)]
```

In order to have a comparison and for better understanding what gensim does or sets into a relation here I tested another word and selected the term "element" to do so, leading to the following results:

```
In [14]: word2 = "element"
         model.wv.most_similar (positive = word2)
```

```
C:\Users\elli\Anaconda3\lib\site-packages\gensim\matutils.py:737: FutureWarning: Conversion of the second argument of issubdtype from `int` to `np.signedinteger` is deprecated. In future, it will be treated as `np.int32 == np.dtype(int).type`.
  if np.issubdtype(vec.dtype, np.int):
```

```
Out[14]: [('third', 0.987095832824707),
          ('make', 0.9315939545631409),
          ('refuelable', 0.9237496852874756),
          ('isru', 0.8931676149368286),
          ('or', 0.8778015375137329),
          ('also', 0.8760696053504944),
          ('utilization', 0.8195888996124268),
          ('resource', 0.7816526293754578),
          ('in-situ', 0.7010257840156555),
          ('working', 0.6892534494400024)]
```

As both results show, I have stop words within the matrix of similar word, that could have been avoided by further pre-processing steps, which I left aside in the run up of the modelling. These stop words are not interpretable. But I received some terms with a content related sense, such as "refueable" - "element" or "third" - "element". These combinations are very likely to happen, as the accompanied score shows.

Comparable results for moon: Some stop words are included, and most often the prefix "the"-"moon" are accompanied with each other. Not surprisingly, but still, to me as a first contact experience it shows gensim brings up reasonable results.

Concluding, I would say that genism is an astonishing tool, with a lot of possibilities, that I just scratched on but did not at al were digging deeper into. Regarding the results I would have reached more detailed similarities if I would have had done further pre-processing, such as lowercasing, stemming or the removal of small words or stop words. And I assume, that the statistical values would have reached another level if the model would have been trained and tested based on a larger text, which is from a mathematical perspective quite logical.