Encoding words as vectors

This workshop was done during the NLP class at the University of Lille, supervised by Pr. Keller.

https://huggingface.co/cmarkea/distilcamembert-base

```
import gensim
import numpy as np
from tqdm import tqdm
from transformers import AutoTokenizer, AutoModel, pipeline

prng = np.random.default_rng(2)
```

1. download skipgram's pre-entrained English word embeddings (without lemmatization).

Embedding is a technique for converting text data into vectors. So we translate each word into a direction in a given space, and if two words have the same meaning, or are often in the same context, then they'll have almost the same direction.

We've downloaded model 222, “English Wikipedia Dump of November 2021”. the embedding is in the `model.txt` file in the `222` folder.

```
222/
├── meta.json
├── model.bin
├── model.txt
└── README
```

This dump is actually a “dump” of all English wikipedia articles. So we'll have an English bias of words.

```
embedding_file = "222/model.txt"
```

```
with open(embedding_file) as file:
    data = file.read().split("\n")
```

```
line_nb, embedding_size = np.array(data[0].split(), dtype=int)
print(f"Number of words : {line_nb}, size of vectors : {embedding_size}")
```

```
Number of words : 199807, size of vectors : 300
```

```
word_list = [x.split()[0] for x in data[1:-1]]
```

```
get_word_by_idx = lambda idx: data[idx].split()[0] if idx<=line_nb and idx>0
else False
get_embd_by_idx = lambda idx: np.array(data[idx].split()[1:], dtype=np.float32)
if idx<=line_nb and idx>0 else False
```

2. Choose a word and calculate its 10 nearest neighbors in terms of the cosine similarity between 2 rows of this matrix. Comment

Avec `gensim`

```
model = gensim.models.KeyedVectors.load_word2vec_format(embedding_file,
binary=False)
```

```
#
https://tedboy.github.io/nlps/generated/generated/gensim.models.Word2Vec.most_s
imilar.html
model.most_similar("Sun", topn=10)
```

```
[('Moon', 0.6126450300216675),
 ('Sky', 0.5235841274261475),
 ('Shines', 0.510596752166748),
 ('Yat', 0.5048438310623169),
 ('sun', 0.49705934524536133),
 ('Tzu', 0.4793660640716553),
 ('Ra', 0.4791143536567688),
 ('Shengnan', 0.4696047902107239),
 ('Jupiter', 0.46762827038764954),
 ('Rises', 0.46191972494125366)]
```

Without `gensim`

```
cosine_similarity = lambda vect1, vect2: np.dot(vect1,
vect2)/(np.linalg.norm(vect1)*np.linalg.norm(vect2))
```

```
choosen_idx = word_list.index("Sun")+1
print(f"Chosen word: {get_word_by_idx(choosen_idx)}")
```

```
Chosen word: Sun
```

```
word_vector = get_embd_by_idx(choosen_idx)
distance = []
for i in tqdm(range(1, line_nb)):
    distance.append(cosine_similarity(word_vector, get_embd_by_idx(i)))
```

```
100%|██████████| 199806/199806 [00:12<00:00, 16106.69it/s]
```

```
for idx in (np.array(distance)).argsort()[::-1][1:11]:
    print(get_word_by_idx(idx+1), distance[idx])
```

```
Moon 0.61264515
Sky 0.5235842
Shines 0.5105968
Yat 0.50484395
sun 0.49705938
Tzu 0.47936606
Ra 0.47911435
Shengnan 0.4696049
Jupiter 0.46762824
Rises 0.46191972
```

To calculate the directional difference between two word vectors, we use *cosine similarity*. It is based on the calculation of the angle between two vectors. It is defined by the following formula:

$$
\textbf{cos-sim}(x, x') = \frac{\langle x, x' \rangle}{\left| x \right| \left| x' \right|}
$$

In Python, we defined it using an anonymous function taken from `cosine_similarity`.

The cosine similarity scores range from -1 to 1 and can be interpreted as follows:

- If it is equal to -1, this represents perfect dissimilarity (the vectors point in completely opposite directions).
- If it is close to 0, it means there is no similarity between the two words.
- If it is close to 1, the two words point in the same direction (the words are similar).
- If it is equal to 1, the words are exactly the same.

We searched for words similar to "Sun," and we found the following:

- "Moon," which makes sense, as these are the two most well-known celestial bodies, the most visible from Earth, often put in opposition (sun versus moon, day versus night, etc.). They participate together in solar eclipses, appear on tarot cards, and so on.
- "Sky," "Shines," and "Rises"—it's in the **sky** that we see the sun **shining** brightly once it has **risen** in the morning.
- "Yat," "Tzu," and "Shengnan," which are all Chinese personalities with this name (Sun Yat, a Chinese statesman also called "the father of modern China," Sun Tzu, the strategist who wrote the famous "The Art of War," and Sun Shengnan, a Chinese tennis player).
- "sun," which is just the same word without the capital letter.
- "Ra," who is the sun god in Egyptian mythology (more precisely, the god of the sun at its zenith, but he is the most famous Egyptian sun god 🤓).
- "Jupiter," which is also a celestial body in the solar system.

So we see that the words are close to "Sun" in a textual sense, rather than geographically.

Test the biases of CamemBERT

```
tokenizer = AutoTokenizer.from_pretrained("cmarkea/distilcamembert-base",
use_fast=False)
model = AutoModel.from_pretrained("cmarkea/distilcamembert-base")
model.eval()

---


model_fill_mask = pipeline("fill-mask", model="cmarkea/distilcamembert-base",
tokenizer="cmarkea/distilcamembert-base", device='cuda')
model_fill_mask("The sun is <mask>")
```

```
[{'score': 0.13973505795001984,
  'token': 22553,
  'token_str': 'beautiful',
  'sequence': 'The sun is beautiful'},
 {'score': 0.06724607199430466,
  'token': 20918,
  'token_str': 'perfect',
  'sequence': 'The sun is perfect'},
```

```
 {'score': 0.036818213760852814,
  'token': 13972,
  'token_str': 'good',
  'sequence': 'The sun is good'},
 {'score': 0.028121499344706535,
  'token': 24041,
  'token_str': 'great',
  'sequence': 'The sun is great'},
 {'score': 0.026972847059369087,
  'token': 10697,
  'token_str': 'more',
  'sequence': 'The sun is more'}]
```

```
model_fill_mask("The star that shines the most is <mask>")
```

```
[{'score': 0.3696179687976837,
  'token': 22553,
  'token_str': 'beautiful',
  'sequence': 'The star that shines the most is beautiful'},
 {'score': 0.08687812089920044,
  'token': 26134,
  'token_str': 'amazing',
  'sequence': 'The star that shines the most is amazing'},
 {'score': 0.039710622280836105,
  'token': 24041,
  'token_str': 'great',
  'sequence': 'The star that shines the most is great'},
 {'score': 0.0306797306984663,
  'token': 24634,
  'token_str': 'awesome',
  'sequence': 'The star that shines the most is awesome'},
 {'score': 0.024502063170075417,
  'token': 29912,
  'token_str': 'interesting',
  'sequence': 'The star that shines the most is interesting'}]
```

We're going to test the cosine similarities with the word “Soleil” and the close words found previously. As this is a French embedding, we'll translate everything from the previous answers into French.

```
vec_soleil = model(**tokenizer("Soleil", return_tensors="pt"))
for mot in ["Lune", "Ciel", "briller", "Yat", "soleil", "Tzu", "Ra",
"Shengnan", "Jupiter", "lever"]:
    vec_mot = model(**tokenizer(mot, return_tensors="pt"))
    result =
```

```
cosine_similarity(vec_mot.pooler_output.detach().numpy().squeeze(),
vec_soleil.pooler_output.detach().numpy().squeeze())

    print(f"{mot}: {result:.7}")
```

```
Lune: 0.9979714
Ciel: 0.9973636
briller: 0.9377761
Yat: 0.9352101
soleil: 0.9983114
Tzu: 0.9214113
Ra: 0.9956771
Shengnan: 0.9876901
Jupiter: 0.9978147
lever: 0.9968085
```

Note that the cosine similarity between the words is very close to the word “Soleil”, the furthest apart being “Yat” and “Tzu”, probably because they come from another language.

```
cosine_similarity(vec_soleil.pooler_output.detach().numpy().squeeze(),
model(**tokenizer("software",
return_tensors="pt")).pooler_output.detach().numpy().squeeze())
```

```
0.8950914
```

Here we see that if we use a word from another language, the distance is greater.

```
cosine_similarity(vec_soleil.pooler_output.detach().numpy().squeeze(),
model(**tokenizer("Sun",
return_tensors="pt")).pooler_output.detach().numpy().squeeze())
```

```
0.99696505
```

```
cosine_similarity(vec_soleil.pooler_output.detach().numpy().squeeze(),
model(**tokenizer("Sonne",
return_tensors="pt")).pooler_output.detach().numpy().squeeze())
```

```
0.9918537
```

But if we put in the translation, we find a high similarity (tested with the word “Soleil” in English and German).

Contributing 🤝

Contributions are highly encouraged! If you have suggestions, improvements, or feature requests, feel free to reach out to me !

License 📝

This project is licensed under the MIT License - see the LICENSE file for details.

Developed by Pierre LAGUE and François MULLER at the University of Lille, France. 🚀📊