ОБРОБКА ПРИРОДНОЇ МОВИ (NLP) У PYTHON. ВКЛАДАННЯ СЛІВ ТА СЕМАНТИЧНА ПОДІБНІСТЬ

ЛЕКЦІЯ 6

- При обробці природної мови вкладання слів використовується для подання слів для аналізу тексту у формі вектора, який виконує кодування значення слова таким чином, щоб слова, які знаходяться ближче у цьому векторному просторі, мали аналогічні у сенсі.

- Вкладання слів — це вивчене уявлення тексту, в якому слова, що мають однакове значення, мають аналогічне подання. Іншими словами, він представляє слова в системі координат, де пов’язані слова, засновані на сукупності відносин, розташовані ближче одне до одного. Саме такий підхід до подання слів і документів можна вважати одним з ключових досягнень глибокого навчання в вирішенні складних проблем обробки природної мови.

- Мета полягає в тому, щоб закодувати (нормовані) слова у вектор, який існує на певній позиції в «просторі слів». По суті, ми представляємо словниковий запас у векторному просторі. В математичних принципах косинусної та евклідової відстані починається магія.

- Word2Vec — один із найпопулярніших методів вивчення вкладання слів з використанням неглибокої нейронної мережі. Його розробив Томаш Міколов у 2013 році в Google. (https://radimrehurek.com/gensim/models/word2vec.html)

- Щоб використовувати вбудовування слів, у вас є два основних варіанти:

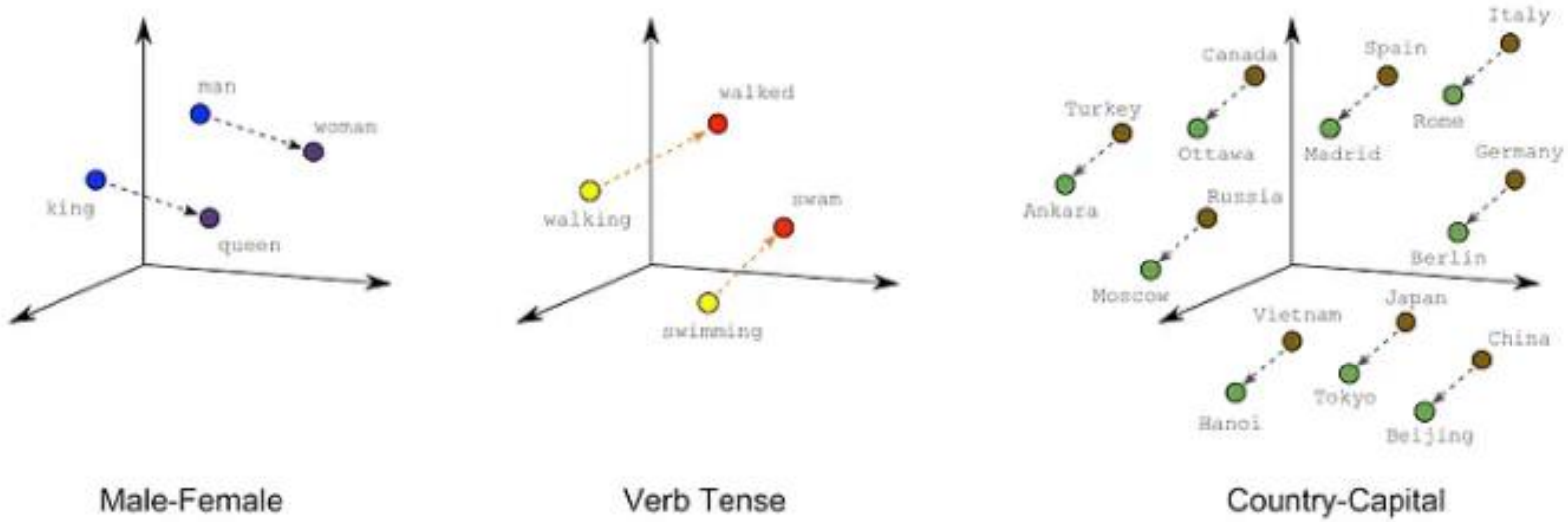
- Використовуйте попередньо навчені моделі, які ви можете завантажити онлайн (найпростіший варіант)
- Навчайте призначені для користувача моделі, використовуючи свої власні дані і алгоритм Word2Vec (або інший).

- Gensim — це бібліотека моделювання тем для Python, яка забезпечує доступ до Word2Vec і інших алгоритмів вбудовування слів для навчання, а також дозволяє завантажувати попередньо навчені вбудовування слів, які ви можете завантажити з Інтернету. Наприклад проекти групи lang-uk Word embeddings (Word2Vec, GloVe, LexVec) (https://lang.org.ua/uk/models/#anchor4).

- Семантична подібність визначається шляхом порівняння векторів слів для «вбудовування слів», багатовимірних представлень значення слова.

- Положення (відстань та напрямок) у векторному просторі може кодувати семантику в хорошому вбудовуванні (https://spacy.io/usage/linguistic-features#vectors-similarity).

- Наприклад, такі візуалізації реальних вкладень показують геометричні відносини, які фіксують семантичні відносини, такі як відносини між країною та її столицею:



https://developers.google.com/machine-learning/crash-course/embeddings/embedding-space

- Такий осмислений простір дає вашій системі машинного навчання можливість виявляти закономірності, які можуть допомогти у вирішенні завдання навчання.

- Давайте встановимо модель української мови Spacy (https://spacy.io/models/uk). Для роботи з векторами потрібно використовувати велику модель.

```
!python -m spacy download uk_core_news_lg
import spacy
nlp = spacy.load("uk_core_news_lg")

doc1 = nlp("Українські військові відбили атаки окупантів у районі
восьми населених пунктів - Генштаб")

for token in doc1:
  print(token.text, token.has_vector)
Українські True
військові True
відбили True
атаки True
окупантів True
у True
районі True
восьми True
населених True
пунктів True
- True
Генштаб True
```

- У прикладі, усі лексеми речення мають вектор, то можемо обчислити семантичну подібність речень:

```
doc2 = nlp("Війська РФ завдали ракетного удару по Дніпропетровщині,
на місці прильоту працюють усі служби. Фото")

doc3 = nlp("Як Європа допомагає Україні через фонд миру: пояснення
посла ЄС")

print(doc1, "<>", doc2, doc1.similarity(doc2))
print(doc1, "<>", doc3, doc1.similarity(doc3))
```

```
Українські військові відбили атаки окупантів у районі восьми
населених пунктів - Генштаб <> Війська РФ завдали ракетного удару по
Дніпропетровщині, на місці прильоту працюють усі служби. Фото
0.7152659483500141
Українські військові відбили атаки окупантів у районі восьми
населених пунктів - Генштаб <> Як Європа допомагає Україні через фонд
миру: пояснення посла ЄС 0.3512228591622778
```

Тут ми використовували вже готову модель з 200 тис. унікальними векторами.

Навчання ваших власних вбудованих слів не повинно бути складним завданням і, для конкретних проблемних областей, призведе до підвищення продуктивності в порівнянні з попередньо навченими моделями.

- Розглянемо приклад з розпізнавання належності назви посади до категорії менеджмент та обслуговування.

```
import pandas as pd

corpus = pd.read_csv('https://drive.google.com/uc?
export=download&id=1JSUVmAkz8ECIMcJZGt8AnscBs0jmSLee', header = 0,
sep = ';')
corpus
```

	titles	is_service
0	Авербандник	1
1	Авіаційний механік з планера та двигунів	1
2	Авіаційний механік з приладів та електроустатк...	1
3	Авіаційний механік з радіоустаткування	1
4	Авіаційний технік (механік) з парашутних та ав...	1
...	...	...
9097	Юрист	1
9098	Юрист-міжнародник	1
9099	Юстирувальник деталей та приладів	1
9100	Юстирувальник оптичних приладів	1
9101	Юстирувальник	1

9102 rows × 2 columns

- Отже, це наш набір даних з колонкою is_service (0 — менеджмент, 1 — обслуговування), яку потрібно вбудувати в модель.

- Виконайте попередню обробку даних: потрібно привести к малому регістру та видалити пунктуацію.

```
import string

corpus['titles'] = corpus['titles'].apply(lambda x :
str(x).lower().strip())

corpus['titles'] = corpus['titles'].apply(lambda x:''.join([i for i
in x if i not in string.punctuation]))
corpus['titles']
```

```
0                                              авербандник
1                  авіаційний механік з планера та двигунів
2         авіаційний механік з приладів та електроустатк...
3                    авіаційний механік з радіоустаткування
4         авіаційний технік механік з парашутних та авар...
                                ...
9097                                                 юрист
9098                                          юристміжнародник
9099                        юстирувальник деталей та приладів
9100                          юстирувальник оптичних приладів
9101                                              юстирувальник
Name: titles, Length: 9102, dtype: object
```

Створимо вектори до ‘titles’, для простоти напишемо функцію:

```
def get_vector(x):
  doc = nlp(x)
  vector = doc.vector
return vector

corpus['vectors'] = corpus['titles'].apply(lambda x: get_vector(x))
corpus.head()
```

	titles	is_service	vectors
0	авербандник	1	[0.4767424, 0.030759603, -2.3752034, 0.0990520...
1	авіаційний механік з планера та двигунів	1	[0.50640464, 0.88554525, 0.45917043, 0.9238674...
2	авіаційний механік з приладів та електроустатк...	1	[0.80926394, 0.974323, 0.465778, 1.3394418, -4...
3	авіаційний механік з радіоустаткування	1	[0.8235947, 1.5725664, 0.21599993, 1.059803, -...
4	авіаційний технік механік з парашутних та авар...	1	[1.3383067, 0.47077453, -0.004781276, 0.634015...

Підготуємо до навчання для нашої моделі. По-перше реорганізуємо масив векторів в один стовпець.

```
X = corpus['vectors'].to_numpy()
X = X.reshape(-1, 1)
X.shape
 (9102, 1)
```

Тепер потрібно перетворити ці дані, щоб зробити розмірність 300.

```
import numpy as np
X = np.concatenate(np.concatenate(X, axis=0), axis = 0).reshape(-1,
300)
X.shape
  (9102, 300)
```

Після цього розділемо на два набори: тренування та тестування.

```
from sklearn.model_selection import train_test_split
y = corpus['is_service']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=0 )
X_train.shape, X_test.shape
  ((6371, 300), (2731, 300))
```

Питання дослідження визначатиме модель машинного навчання для виконання, і це сильно залежить від ваших тренувальних та тестових змінних.

- Але для вирішення цієї проблеми існує пакет Python LazyPredict (https://pypi.org/project/lazypredict/) — чудовий інструмент, який автоматично запускає дані через різні типи моделей і повертає інформацію про продуктивність кожної моделі.

```
!pip install lazypredict

from lazypredict.Supervised import LazyClassifier

clf = LazyClassifier(verbose=0, ignore_warnings=True,
custom_metric=None, classifiers="all", random_state=0)

models, predictions = clf.fit(X_train, X_test, y_train, y_test)

models
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
PassiveAggressiveClassifier	0.99	0.97	0.97	0.99	0.27
LogisticRegression	0.99	0.96	0.96	0.99	0.27
LinearSVC	0.98	0.96	0.96	0.98	0.42
Perceptron	0.98	0.95	0.95	0.98	0.14
SVC	0.98	0.94	0.94	0.98	1.39
CalibratedClassifierCV	0.98	0.93	0.93	0.98	1.56
LinearDiscriminantAnalysis	0.97	0.91	0.91	0.97	0.41
SGDClassifier	0.97	0.91	0.91	0.97	0.68
GaussianNB	0.91	0.89	0.89	0.91	0.07
KNeighborsClassifier	0.96	0.89	0.89	0.96	0.71
AdaBoostClassifier	0.96	0.89	0.89	0.96	15.42
LGBMClassifier	0.97	0.89	0.89	0.97	6.56
BernoulliNB	0.89	0.88	0.88	0.90	0.11
NearestCentroid	0.88	0.88	0.88	0.90	0.09
RidgeClassifier	0.97	0.88	0.88	0.97	0.09
RidgeClassifierCV	0.97	0.88	0.88	0.97	0.31
XGBClassifier	0.96	0.88	0.88	0.96	9.40
BaggingClassifier	0.95	0.85	0.85	0.95	17.31
ExtraTreesClassifier	0.95	0.82	0.82	0.95	1.20
RandomForestClassifier	0.95	0.82	0.82	0.95	7.88
DecisionTreeClassifier	0.92	0.79	0.79	0.92	3.12
QuadraticDiscriminantAnalysis	0.92	0.79	0.79	0.91	0.31
ExtraTreeClassifier	0.90	0.76	0.76	0.90	0.08
LabelSpreading	0.14	0.51	0.51	0.05	1.95
LabelPropagation	0.14	0.51	0.51	0.05	1.46
DummyClassifier	0.87	0.50	0.50	0.82	0.06

- Отже для наших змінних найкращою моделлю машиного навчання буде PassiveAggressiveClassifier. (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.PassiveAggressiveClassifier.html)

```
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.metrics import classification_report

clf = PassiveAggressiveClassifier(C = 0.5, random_state = 5)
clf.fit(X_train, y_train)

y_predict = clf.predict(X_test)
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.95      0.91      0.93       344
           1       0.99      0.99      0.99      2387

    accuracy                           0.98      2731
   macro avg       0.97      0.95      0.96      2731
weighted avg       0.98      0.98      0.98      2731
```

- Наостанок збережемо модель для можливості переносу та виконання нових прогнозів. Ви можете використовувати операцію pickle (https://docs.python.org/3/library/pickle.html), щоб серіалізувати свої моделі машинного навчання у файл.

```
import pickle
pickle.dump(clf, open('model_service.pkl', 'wb'))

model = pickle.load(open('model_service.pkl', 'rb'))

model.predict(np.array(get_vector('Керівник відділу
продажу')).reshape(1, -1))
  array([0])
```