Vektorisasi Part 2

```
pip install PyPDF2
```

```
Requirement already satisfied: PyPDF2 in /usr/local/lib/python3.10/dist-packages (3.0.1)
```

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
import PyPDF2
import pandas as pd
from gensim.models import Word2Vec, FastText
from nltk.tokenize import sent_tokenize, word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
```

1.

+ Kode + Teks

```
# Membaca teks dari file PDF
def extract_text_from_pdf(pdf_file):
    text = ""
    with open(pdf_file, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        for page_number in range(len(reader.pages)):
            text += reader.pages[page_number].extract_text()
    return text

# Mendapatkan teks dari file PDF
pdf_file = 'MALIN_KUNDANG.pdf'
data = extract_text_from_pdf(pdf_file)
```

```
data
```

```
'MALIN KUNDANG \nPada suatu waktu, hiduplah sebuah keluarga nelayan di \npesisir pantai wilayah Sumatra. Keluarga ters
ebut terdiri \ndari ayah, ibu dan seorang anak laki-laki yang diberi nama \nMalin Kundang. Karena kondisi keuangan kel
uarga yang \nmemprihatinkan, sang ayah memutuskan untuk mencari \nnafkah di negeri seberang dengan mengarungi lautan y
ang \nluas.\nMaka tinggallah si Malin dan ibunya di gubug mereka. Semingg u, dua minggu, sebulan, dua \nbulan bahkan s
udah 1 tahun lebih lamanya, ayah Malin tidak  juga kembali ke kampung \nhalamannya. Sehingga ibunya harus menggantikan
posisi ayah Malin untuk mencari nafkah. \nMalin termasuk anak yang cerdas tetapi sedikit nakal. Ia  sering mengejar ay
am dan \nmemukulnya dengan sapu. Suatu hari ketika Malin sedang me ngejar ayam, ia tersandung \nbatu dan lengan kanann
```

```
token = sent_tokenize(data)
teks = ''.join(token)

corpus = word_tokenize(teks)
corpus
```

```
['MALIN',
 'KUNDANG',
 'Pada',
 'suatu',
 'waktu',
 ',',
 'hiduplah',
 'sebuah',
 'keluarga',
 'nelayan',
 'di',
 'pesisir',
 'pantai',
 'wilayah',
 'Sumatra.Keluarga',
 'tersebut',
 'terdiri',
 'dari',
 'ayah',
 ',',
 'ibu',
 'dan',
 'seorang',
 'anak',
 'laki-laki',
 'yang',
 'diberi',
 'nama',
 'Malin',
 'Kundang.Karena',
 'kondisi',
 'keuangan',
 'keluarga',
 'yang',
 'memprihatinkan',
 ',',
 'sang',
 'ayah',
 'memutuskan',
 'untuk',
 'mencari',
 'nafkah',
 'di',
 'negeri',
 'seberang',
 'dengan',
 'mengarungi',
 'lautan',
 'yang',
 'luas.Maka',
 'tinggallah',
 'si',
 'Malin',
 'dan',
 'ibunya',
 'di',
 'gubug',
 'mereka.Semingg',
```

```
hasil_encoding = pd.get_dummies(corpus)
hasil_encoding
```

	!	'	''	,	-buahan	-manggil	-masing	-mutar	.	..	...	yata	yatim	yelamatkan	yet.Mereka	yik	yikan	yir	y
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
39398	0	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
39399	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
39400	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
39401	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
39402	0	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	

2.

```
from sklearn.feature_extraction.text import HashingVectorizer

vectorizer = HashingVectorizer(n_features=10)
hashed_representation = vectorizer.fit_transform([data])

hash = pd.DataFrame(hashed_representation.toarray())
hash
```

	0	1	2	3	4	5	6	7	8	9
0	0.097088	-0.525854	-0.282916	0.027677	0.230199	-0.601855	0.059746	-0.075561	0.385275	-0.245574

3.

```
vectorizer = CountVectorizer(tokenizer=lambda x: x.split(), ngram_range=(1, 1))
data_occur = vectorizer.fit_transform([data])
hasil = (data_occur.T * data_occur)
hasil.setdiag(0)

# Menyimpan hasil co-occurrence matrix ke dalam DataFrame
occurence = pd.DataFrame(hasil.toarray(), index=vectorizer.get_feature_names_out(), columns=vectorizer.get_feature_names_out())

# Menampilkan DataFrame
occurence
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_patte
  warnings.warn(
```

	!	!"	!",	!".	!,"	!.	"	",	"a...	"aaa...	...	yet.	yik	yikan	yir	yosaku	yosaku,	yosaku.	yuk!"
!	0	33	44	11	22	11	264	22	11	11	...	11	11	11	22	110	44	55	11
!"	33	0	12	3	6	3	72	6	3	3	...	3	3	3	6	30	12	15	3
!",	44	12	0	4	8	4	96	8	4	4	...	4	4	4	8	40	16	20	4
!".	11	3	4	0	2	1	24	2	1	1	...	1	1	1	2	10	4	5	1
!,"	22	6	8	2	0	2	48	4	2	2	...	2	2	2	4	20	8	10	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
yosaku,	44	12	16	4	8	4	96	8	4	4	...	4	4	4	8	40	0	20	4
yosaku.	55	15	20	5	10	5	120	10	5	5	...	5	5	5	10	50	20	0	5
yuk!"	11	3	4	1	2	1	24	2	1	1	...	1	1	1	2	10	4	5	0
yut	11	3	4	1	2	1	24	2	1	1	...	1	1	1	2	10	4	5	1
zam!	11	3	4	1	2	1	24	2	1	1	...	1	1	1	2	10	4	5	1

7647 rows × 7647 columns

4.

```
# Tokenisasi teks
sentences = sent_tokenize(data)
tokens = [word_tokenize(sentence.lower()) for sentence in sentences]

# Membangun model Word2Vec
model_word = Word2Vec(tokens, vector_size=10, window=3, min_count=1, workers=4)

# Mendapatkan representasi vektor kata
word_vectors = pd.DataFrame(model_word.wv.vectors, index=model_word.wv.index_to_key)

# Menampilkan top 5 kata dengan vektor tertinggi
top_words = word_vectors.sum(axis=1).nlargest(5).index
top_vectors = word_vectors.loc[top_words]

print("Top 5 kata dengan vektor tertinggi:")
top_vectors
```

Top 5 kata dengan vektor tertinggi:

5.

dan -0.254486 1.289224 4.115761 0.777421 0.759582 1.672974 2.282815 1.822091 2.110780 1.606782

```
# Membangun model fasttext
model_fast = FastText(tokens, vector_size=10, window=5, min_count=1, workers=4)

# Mendapatkan representasi vektor kata
vektor_kata = pd.DataFrame(model_fast.wv.vectors, index=model_fast.wv.index_to_key)

# Menampilkan top 5 kata dengan vektor tertinggi
top_fasttext = vektor_kata.sum(axis=1).nlargest(10000).index
top_textfast = vektor_kata.loc[top_fasttext]

print("Top 5 kata dengan vektor tertinggi:")
top_textfast
```

Top 5 kata dengan vektor tertinggi:

	0	1	2	3	4	5	6	7	8	9
kan	4.833005	1.350613	0.666528	7.190577	1.443925	5.032516	3.874599	4.983437	1.454753	8.231963
an	4.406422	1.265550	0.574429	6.542119	1.310891	4.653541	3.545540	4.567470	1.341016	7.490070
meng	4.279175	1.214565	0.548248	6.408331	1.315226	4.524707	3.475570	4.411793	1.342272	7.277483
men	4.127113	1.211929	0.545780	6.219424	1.234795	4.388599	3.357790	4.243525	1.300012	7.126489
tan	3.762504	1.085176	0.509131	5.607734	1.108521	3.969334	3.023685	3.972352	1.131540	6.424398
...	...	...	...	...	...	...	...	...	...	...
999	-0.040855	-0.018895	-0.028650	0.039473	0.007115	0.015630	-0.012646	0.011534	0.020434	0.008612
'	-0.019271	-0.016191	0.006290	0.016182	-0.029571	-0.082454	0.008268	0.041074	-0.017839	0.072930
20	0.045958	0.032440	-0.045486	-0.017957	-0.021666	-0.004383	0.009055	0.018967	-0.024046	-0.022677
o	-0.041124	-0.018746	0.002452	0.045818	-0.031070	-0.005146	0.026030	-0.023567	-0.078748	0.023145
12	0.003201	0.012481	-0.046323	-0.046409	-0.016086	-0.078659	0.043150	-0.011472	0.014817	-0.028632

5421 rows × 10 columns