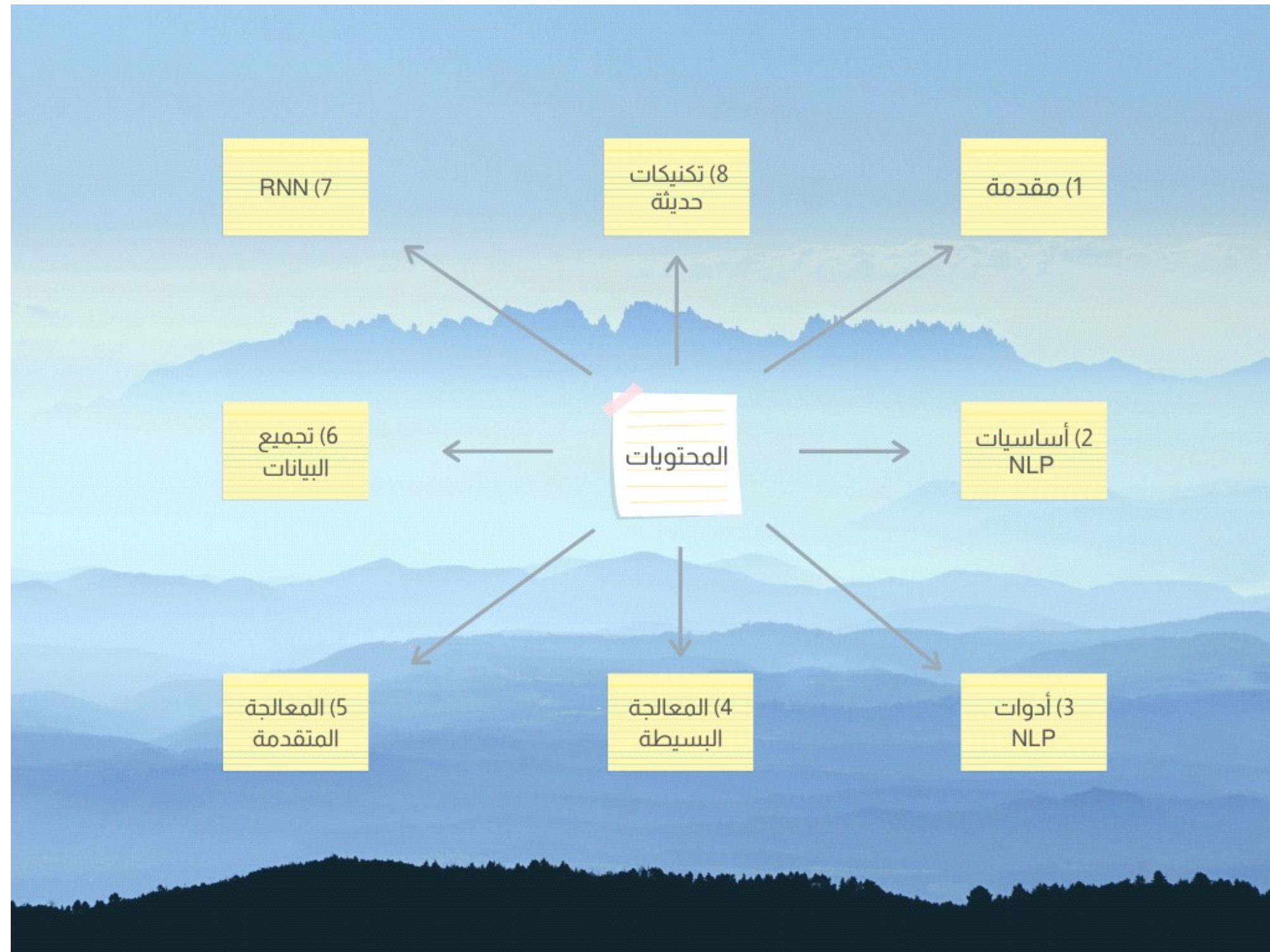


# NATURAL LANGUAGE PROCESSING

# المعالجة اللغوية الطبيعية



# المحتويات

				التطبيقات	العقبات و التحديات	تاريخ NLP	ما هو NLP	المحتويات	1) مقدمة
					البحث في النصوص	ملفات pdf	الملفات النصية	المكتبات	2) أساسيات NLP
T.Visualization	Syntactic Struc.	Matchers	Stopwords	NER	Stem & Lemm	POS	Sent. Segm.	Tokenization	3) أدوات NLP
	Dist. Similarity	Text Similarity	TF-IDF	BOW	Word2Vec	T. Vectors	Word embed	Word Meaning	4) المعالجة البسيطة
T. Generation	NGrams	Lexicons	GloVe	L. Modeling	NMF	LDA	T. Clustering	T. Classification	5) المعالجة المتقدمة
	Summarization & Snippets		Ans. Questions		Auto Correct	Vader	Naïve Bayes	Sent. Analysis	
Search Engine	Relative Extraction		Information Retrieval		Information Extraction		Data Scraping	Tweet Collecting	6) تجميع البيانات
					Rec NN\TNN	GRU	LSTM	Seq to Seq	7) RNN
Chat Bot	Gensim	FastText	Bert	Transformer	Attention Model	T. Forcing	CNN	Word Cloud	8) تكنيكات حديثة

## القسم الخامس : المعالجة المتقدمة للنصوص

## Non-Negative Matrix Factorization : الجزء الرابع

الـ **Non-Negative Matrix Factorization** أو تحليل المصفوفة الموجبة وهو اسلوب مشابه في مهمة تصنيف النصوص مثل LDA لكن مختلف نوعا في الفكرة , وهو اسرع و قد يكون ادق

و يعتمد في فكرته علي عمل مصفوفتين من ارقام عشوائية , واحدة هي العلاقة بين الكلمات و الموضوع , والثانية بين الموضوع و ال-clutser , والتي تكون الأساس الرياضي لتصنيف النصوص كما حدث في LDA

وهو يستخدم في NLP و ايضا في تطبيقات image processing

و الكود مشابه الى حد كبير الكود السابق هكذا . .

\* \* \* \* \*

```
import pandas as pd  
npr = pd.read_csv('npr.csv')  
npr.head()
```

بدلا من عمل خطوة countvectorizer ثم LDA , نقوم باستدعاء Tfidf لاستخدامها

```
from sklearn.feature_extraction.text import TfidfVectorizer  
tfidf = TfidfVectorizer(max_df=0.95, min_df=2, stop_words='english')  
dtm = tfidf.fit_transform(npr['Article'])
```

ثم نستخدم NMF و هي اختصار Nonnegative Matrix Factorization

```
from sklearn.decomposition import NMF  
nmf_model = NMF(n_components=7, random_state=42)  
nmf_model.fit(dtm)
```

```
len(tfidf.get_feature_names())
```

```
import random  
for i in range(10):  
    random_word_id = random.randint(0,54776)  
    print(tfidf.get_feature_names()[random_word_id])
```

```
len(nmf_model.components_)
```

```
nmf_model.components_
```

```
len(nmf_model.components_[0])
```

```
single_topic = nmf_model.components_[0]
```



```
single_topic.argsort()
```

```
single_topic[18302]
```

```
single_topic[42993]
```

```
single_topic.argsort()[-10:]
```

```
top_word_indices = single_topic.argsort()[-10:]
```

```
for index in top_word_indices:  
    print(tfidf.get_feature_names()[index])
```

```
for index,topic in enumerate(nmf_model.components_):  
    print(f'THE TOP 15 WORDS FOR TOPIC #{index}')  
    print([tfidf.get_feature_names()[i] for i in topic.argsort()[-15:]])  
    print('\n')
```

```
dtm.shape
```

```
len(npr)
```

```
topic_results = nmf_model.transform(dtm)
```

```
topic_results.shape
```

```
topic_results[0].round(2)
```

```
topic_results[0].argmax()
```

```
npr.head()
```

```
topic_results.argmax(axis=1)
```

```
npr['Topic'] = topic_results.argmax(axis=1)
```

```
topicdict = {0:'health',1:'election',2:'legis',3:'policy',4:'candidates',5:'music',6:'educaion'}
```

```
npr['Topic Label'] = npr['Topic'].map(topicdict)
```

```
npr.head(10)
```

```
*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*
```