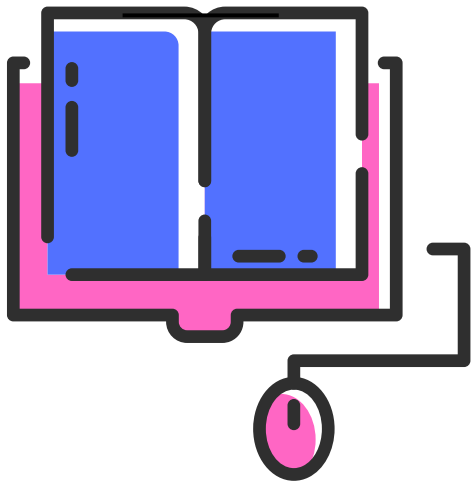
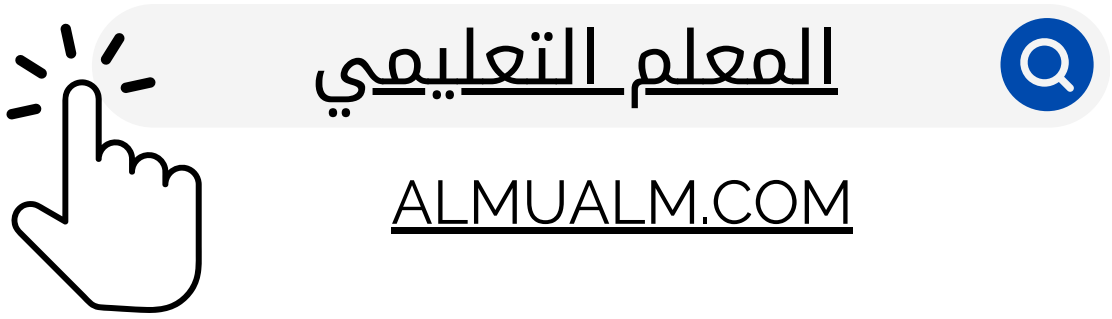
تم تحميل ورفع المادة على منصة



المعلم التعليمي

للعودة الى الموقع اكتب في بحث جوجل



3. معالجة اللغات الطبيعية

سيتعلّم الطالب في هذه الوحدة عملية تدريب شاملة لنموذج التعلُّم الموجَّه والتعلُّم غير الموجَّه لفهم المعنى الكامن في أجزاء النصوص. وكذلك سيتعلّم كيفية استخدام تعلُّم الآلة (Machine Learning - ML) في دعم التطبيقات ذات الصلة بمعالجة اللغات الطبيعية (Natural Language Processing -NLP).

أهداف التعلُّم

بنهاية هذه الوحدة سيكون الطالب قادرًا على أن:

- يُعرِّف التعلُّم الموجَّه.
- يُدرِّب نموذج التعلُّم الموجَّه على فهم النص.
- يُعرِّف التعلُّم غير الموجَّه.
- يُدرِّب نموذج التعلُّم غير الموجَّه على فهم النص.
- يُنشئ روبوت دردشة بسيط.
- يُنتج النصوص باستخدام تقنيات توليد اللغات الطبيعية (Natural Language Generation -NLG).

الأدوات

- مفكرة جوبيتر (Jupyter Notebook)

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الأول

التعلُّم الموجَّه

استخدام التعلُّم الموجَّه لفهم النصوص

Using Supervised Learning to Understand Text

معالجة اللغات الطبيعية (Natural Language Processing – NLP) هي إحدى مجالات **الذكاء الاصطناعي** (Artificial Intelligence – AI) التي تركّز على تمكين أجهزة الحاسب لتصبح قادرة على فهم اللغات البشريّة، وتفسيرها، وإنتاجها. حيث تُعنى معالجة اللغات الطبيعية بعدد من المهام، مثل: تصنيف النصوص، وتحليل المشاعر، والترجمة الآلية، والإجابة على الأسئلة. سيركز هذا الدرس بشكل خاص على كيفية استخدام التعلُّم الموجَّه الذي يُعدُّ أحد الأنواع الرئيسة **لتعلُّم الآلة** (Machine Learning – ML) في تحقيق الفهم والتنبؤ التلقائي لخصائص النصوص.

لقد تعلّمت في الوحدة الأولى أن الذكاء الاصطناعي هو مصطلح يشملُ كلًّا من تعلُّم الآلة والتعلُّم العميق، كما يتضح في الشكل 3.1، فالذكاء الاصطناعي هو ذلك المجال الواسع من علوم الحاسب الذي يُعنى بابتكار آلات ذكية، بينما تعلُّم الآلة هو أحد فروع الذكاء الاصطناعي الذي يركّز على تصميم الخوارزميات وبِناء النماذج التي تُمكِّن الآلة من التعلُّم من البيانات دون الحاجة إلى برمجتها بشكل صريح.

شكل 3.1: فروع الذكاء الاصطناعي

التعلُّم العميق (Deep learning):
التعلُّم العميق هو أحد أنواع تعلُّم الآلة الذي يستخدِم الشبكات العصبية العميقة للتعلُّم تلقائيًا من مجموعات كبيرة من البيانات، فهو يسمح لأجهزة الحاسب بالتعرّف على الأنماط واتخاذ القرارات بطريقة تحاكي الإنسان، عبر تصميم نماذج مُعقدة من البيانات.

تعلُّم الآلة Machine Learning

تعلُّم الآلة هو أحد فروع الذكاء الاصطناعي المعني بتطوير الخوارزميات التي تُمكِّن أجهزة الحاسب من التعلُّم من البيانات المُدخَلة، بدلًا من اتباع التعليمات البرمجية الصريحة، فهو يعمل على تدريب نماذج الحاسب للتعرُّف على الأنماط والقيام بالتنبؤات وفقًا للبيانات المُدخَلة مما يسمح للنموذج بتحسين الدقة مع مرور الوقت، وكذلك يتيح للآلة أداء مهام متعددة، مثل: التصنيف، والانحدار، والتجميع، وتقديم التوصيات دون الحاجة إلى برمجة الآلة بشكل صريح للقيام بكل مُهِمَّة على حدة. يمكن تصنيف تعلُّم الآلة إلى ثلاثة أنواع رئيسة:

التعلُّم الموجَّه (Supervised learning) هو نوع من تعلُّم الآلة تتعلّم فيه الخوارزمية من بيانات تدريب **مُعنوَنة** (Labelled) بهدف القيام بالتنبؤات حول بيانات جديدة غير موجودة في مجموعة التدريب أو الاختبار كما هو موضح في شكل 3.2، ومن الأمثلة عليه:

- **تصنيف الصور** (Image Classification)، مثل: التعرف على الكائنات في الصور.
- **كشف الاحتيال** (Fraud Detection)، مثل: تحديد المُعامَلات المالية المشبوهة.
- **تصفية البريد الإلكتروني العشوائي** (Spam Filtering)، مثل: تحديد رسائل البريد الإلكتروني غير المرغوب فيها.

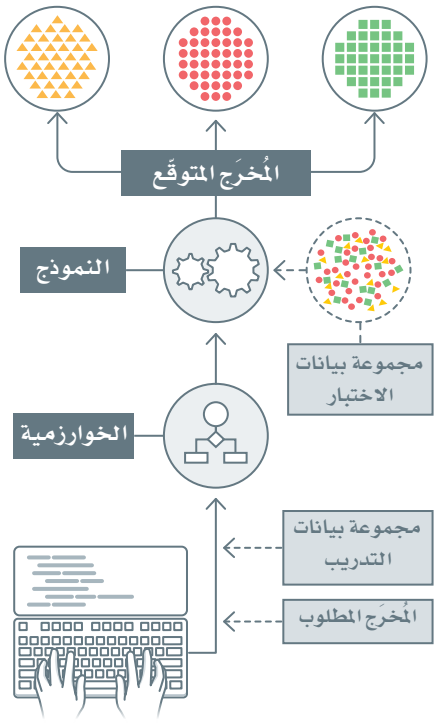
التعلُّم غير الموجَّه (Unsupervised learning) هو نوع من تعلُّم الآلة تعمل فيه الخوارزمية بموجب بيانات **غير مُعنوَنة** (Unlabeled) في محاولة لإيجاد الأنماط والعلاقات بين البيانات، ومن الأمثلة عليه:

- **الكشف عن الاختلاف** (Anomaly Detection)، مثل: تحديد الأنماط غير العادية في البيانات.
- **التجميع** (Clustering)، مثل: تجميع البيانات ذات الخصائص المتشابهة.
- **تقليص الأبعاد** (Dimensionality Reduction)، مثل: اختيار الأبعاد المُستخدَمة للحدِّ من تعقيد البيانات.

التعلُّم المعزَّز (Reinforcement learning) هو نوع من تعلُّم الآلة تتفاعل فيه الآلة مع البيئة المحيطة وتتعلّم عبر المحاولة والخطأ أو تلقّي المكافأة والعقاب، ومن الأمثلة عليه:

- لعب الألعاب، مثل: لعبة الشطرنج أو لعبة **قو** (GO) .
- الروبوتية، مثل: تعليم الروبوت كيف يتنقل في البيئة المحيطة به.
- تخصيص الموارد، مثل: تحسين استخدام الموارد في شبكة ما.

جدول 3.1 يلخص مزايا وعيوب أنواع تعلُّم الآلة.



شكل 3.2: تمثيل التعلُّم الموجَّه

جدول 3.1: مزايا أنواع تعلّم الآلة، وعيوبها

المزايا	العيوب
التعلُّم الموجَّه	
• أثبت كفاءة وفعالية كبيرة ويُستخدم على نطاق واسع. • سهل الفهم والتطبيق. • يُمكنه التعامل مع البيانات الخطية وغير الخطية على حد سواء.	• يتطلب بيانات مُعنوَنة، والتي قد تكون مرتفعة التكلفة. • يقتصر استخدامه على المُهِمَّة التي تم تدريبه عليها، وقد لا يمكنه إعطاء التنبؤ الصحيح للبيانات الجديدة. • يصعب تكيفه مع المشكلات الأخرى في حالات النماذج المُعقدة جدًا.
التعلُّم غير الموجَّه	
• لا يتطلب بيانات مُعنوَنة، مما يجعله أكثر مرونة. • يُمكنه اكتشاف الأنماط الخفية في البيانات. • يُمكنه التعامل مع البيانات الضخمة والمُعقدة.	• أصعب من التعلُّم الموجَّه من حيث الفهم والتفسير. • يقتصر على التحليل الاستكشافي، وقد لا يناسب عمليات صنع القرار. • يصعب تكيفه مع المشكلات الأخرى في حالات النماذج المُعقدة جدًا.
التعلُّم المعزَّز	
• يتّسم بالمرونة، ويُمكنه التعامل مع البيئات المُعقدة والمتغيرة باستمرار. • يمكنه التعلُّم من التجارب السابقة وتحسين الكفاءة مع مرور الوقت. • يتناسب مع عمليات صنع القرار مثل لعب الألعاب والروبوتية.	• أكثر تعقيدًا من التعلّم الموجَّه وغير الموجَّه. • صعوبة تصميم نظم مكافآت تُحدد السلوك المطلوب بشكل دقيق. • قد يتطلب مجموعات كبيرة من بيانات التدريب والموارد الحسابية.

التعلُّم الموجَّه Supervised Learning

التعلُّم الموجَّه هو أحد أنواع تعلُّم الآلة الذي يعتمد على استخدام البيانات المُعنوَنة لتدريب الخوارزميات للقيام بالتنبؤات. يتم تدريب الخوارزمية على مجموعة من البيانات المُعنوَنة ثم اختبارها على مجموعة بيانات جديدة لم تكن جزءًا من بيانات التدريب. يُستخدَم التعلُّم الموجَّه عادةً في معالجة اللغات الطبيعية للقيام بمهام مثل: تصنيف النصوص، وتحليل المشاعر، **والتعرّف على الكيانات المسماة** (Named Entity Recognition – NER). في هذه المهام يتم تدريب الخوارزمية على مجموعة من البيانات المُعنوَنة، حيث يتم إدراج كل مثال تحت عنوان التصنيف المناسب أو المشاعر المناسبة. يُطلَق على عملية التعلُّم الموجَّه اسم **الانحدار** (Regression) عندما تكون القيم التي تتنبأ بها الآلة رقميّة، بينما يطلق عليها اسم **التصنيف** (Classification) عندما تكون القيم متقطّعه.

التعلُّم الموجَّه (Supervised Learning):
ستستخدم في التعلُّم الموجَّه مجموعات البيانات المُعنوَنة والمُنظمة بشكل يدوي لتدريب خوارزميات الحاسب على التنبؤ بالقيم الجديدة.

الانحدار

على سبيل المثال، قد يُستخدم الانحدار في التنبؤ بسعر بيع المنزل وفقًا لمساحته، وموقعه، وعدد غرف النوم فيه. كما يمكن استخدامه في التنبؤ بحجم الطلب على أحد المنتجات استنادًا إلى بيانات المبيعات التاريخية وحجم الإنفاق الإعلاني. وفي مجال معالجة اللغات الطبيعية، يَستخدِم الانحدار النصوص المُدخَلة المتوفرة للتنبؤ بتقييم الجمهور للفيلم أو مدى التفاعل مع المنشورات الخاصة به على وسائل التواصل الاجتماعي.

التصنيف

من ناحية أخرى، يُستخدم التصنيف في التطبيقات مثل: تشخيص الحالات الطبية وفقًا للأعراض ونتائج الفحوصات. وعندما يتعلق الأمر بفهم النصوص، يمكن استخدام التعلُّم الموجَّه في تصنيف النصوص المُدخَلة إلى فئات أو عناوين أو التنبؤ بها بناءً على الكلمات أو العبارات الموجودة في المُستنَد. على سبيل المثال، يمكن تدريب نموذج التعلُّم الموجَّه لتصنيف رسائل البريد الإلكتروني إلى رسائل مزعجة أو غير مزعجة وفقًا للكلمات أو العبارات المُستخدَمة في رسالة البريد الإلكتروني. ويُعدّ تصنيف المشاعر أحد التطبيقات الشهيرة كذلك، حيث يمكن التنبؤ بالانطباع العام حول مستند ما سواء كان سلبيًّا أم إيجابيًّا. وسَيُستخدم هذا التطبيق كمثال عملي في هذه الوحدة، لشرح كل خطوات عملية بِناء واستخدام نموذج التعلُّم الموجَّه بشكل شامل من بداية رحلة التعلم حتى نهايتها.

في هذه الوحدة ستَستخدم مجموعة بيانات من مراجعات الأفلام على موقع IMDb.com الشهير. ستجد البيانات مُقسّمه إلى مجموعتين؛ الأولى ستُستخدم لتدريب النموذج، والثانية لاختبار أداء النموذج. في البداية لابد أن تُحَمّل البيانات إلى DataFrame، لِذا عليك استخدام مكتبة **باندا بايثون** (Pandas Python) والتي استخدمتها سابقًا. مكتبة باندا هي إحدى الأدوات الشهيرة التي تُستخدم للتعامل مع جداول البيانات. التعليمات البرمجية التالية ستقوم باستيراد المكتبة إلى البرنامج، ثم تحميل مجموعتي البيانات:

```
%%capture  # capture is used to suppress the installation output.

# install the pandas library, if it is missing.
!pip install pandas
import pandas as pd
```

مكتبة باندا هي مكتبة شهيرة تُستخدم لقراءة ومعالجة البيانات الشبيهة بجداول البيانات.

```
# load the train and testing data.
imdb_train_reviews=pd.read_csv('imdb_data/imdb_train.csv')
imdb_test_reviews=pd.read_csv('imdb_data/imdb_test.csv')

imdb_train_reviews
```

	text	label
0	I grew up (b. 1965) watching and loving the Th...	0
1	When I put this movie in my DVD player, and sa...	0
2	Why do people who do not know what a particula...	0
3	Even though I have great interest in Biblical ...	0
4	Im a die hard Dads Army fan and nothing will e...	1
...	...	...
39995	"Western Union" is something of a forgotten cl...	1
39996	This movie is an incredible piece of work. It ...	1
39997	My wife and I watched this movie because we pl...	0
39998	When I first watched Flatliners, I was amazed....	1
39999	Why would this film be so good, but only gross...	1

40000 rows × 2 columns

شكل 3.3: مجموعة بيانات التدريب المُعنوَنة

وكما يتضح في الشكل 3.3، فإن مجموعة بيانات DataFrame تحتوي على عمودين:

- نصّ التقييم.
- القيم (الصَّنف).

تقييم إيجابي

تقييم سلبي

القيمة 0 تمثل تقييمًا سلبيًّا

بينما القيمة 1 تمثل تقييمًا إيجابيًّا.

الخطوة التالية هي إسناد أعمدة النص والقيم إلى متغيرات مستقلة في أمثلة التدريب والاختبار المُمثّلة كمجموعة بيانات DataFrame كما يلي:

```
# extract the text from the 'text' column for both training and testing.
X_train_text=imdb_train_reviews['text']
X_test_text=imdb_test_reviews['text']

# extract the labels from the 'label' column for both training and testing.
Y_train=imdb_train_reviews['label']
Y_test=imdb_test_reviews['label']
X_train_text  # training data in text format
```

تستخدم الرموز X وY عادةً في التعلُّم الموجَّه فيعبّر X عن البيانات المدخلة للتنبؤ، وY عن القيم المستهدفة.

```
0        I grew up (b. 1965) watching and loving the Th...
1        When I put this movie in my DVD player, and sa...
2        Why do people who do not know what a particula...
3        Even though I have great interest in Biblical ...
4        Im a die hard Dads Army fan and nothing will e...
                               ...
39995    "Western Union" is something of a forgotten cl...
39996    This movie is an incredible piece of work. It ...
39997    My wife and I watched this movie because we pl...
39998    When I first watched Flatliners, I was amazed....
39999    Why would this film be so good, but only gross...
Name: text, Length: 40000, dtype: object
```

شكل 3.4: صورة من أمثلة التدريب (X___train___text) من مجموعة بيانات DataFrame

تجهيز البيانات والمعالجة المُسبقة Data Preparation and Pre-Processing

على الرغم من أن تنسيق النص الأولي كما في شكل 3.4 بديهي للقارئ البشري، إلا أنَّ خوارزميات التعلُّم الموجَّه لا تستطيع التعامل معه بصورته الحالية. فبدلًا من ذلك، تحتاج الخوارزميات إلى تحويل هذه المُستنَدات إلى تنسيق **متَّجَه رقمي** (Numeric Vector). فيما يُعرف بعملية **البرمجة الاتجاهية** (Vectorization). ويمكن تطبيق عملية البرمجة الاتجاهية بعدة طرائق مختلفة، وتتميز بأن لها تأثيرًا إيجابيًّا كبيرًا على أداء النموذج المُدرّب.

مكتبة سكليرن Sklearn Library

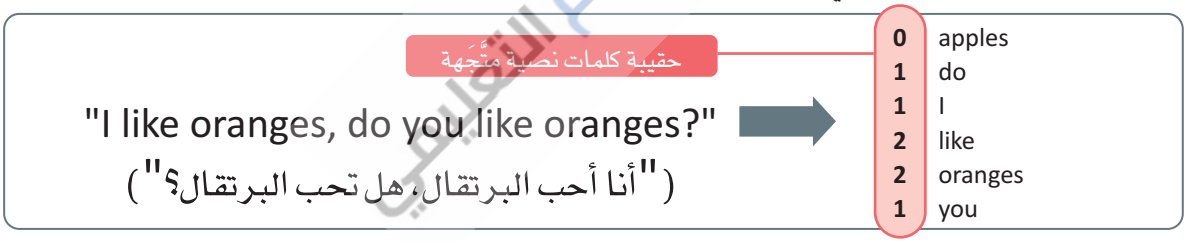
سيتم بناء النموذج الموجَّه باستخدام مكتبة سكليرن وتُعرف كذلك باسم مكتبة **سايكيت ليرن** (Scikit-Learn)، وهي مكتبة شهيرة في بايثون تختص بتعلُّم الآلة. توفر المكتبة مجموعة من الأدوات والخوارزميات لأداء مهام متعددة، مثل: التصنيف، والانحدار، والتجميع، وتقليص الأبعاد. إحدى الأدوات المفيدة في مكتبة سكليرن هي أداة تُسمى CountVectorizer، ويمكن استخدامها في تهيئة عملية المعالجة وتمثيل البيانات النصية بالمتَّجَهات.

أداة CountVectorizer

تُستخدم أداة CountVectorizer في تحويل مجموعة من المُستنَدات النصية إلى مصفوفة من رموز متعددة، حيث يمثّل كلّ صفّ مستندًا وكل عمود يمثل رمزًا خاصًا. قد تكون الرموز كلمات فردية أو عبارات أو بُنيات أكثر تعقيدًا تقوم بالتقاط الأنماط المتعددة من البيانات النصية الأساسية. تُشير المُدخَلات في المصفوفة إلى عدد مرات ظهور الرمز في كل مستند. ويُعرف ذلك أيضًا باسم تمثيل **حقيبة الكلمات** (BoW) "bag-of-words"، حيث يتجاهل ترتيب الكلمات في النص مع المحافظة على تكرارها فيه. على الرغم من أن تمثيل حقيبة الكلمات هو تبسيط شديد للغة البشرية، إلا أنه يحقق نتائج تنافسية للغاية عند التطبيق العملي.

البرمجة الاتجاهية (Vectorization):

البرمجة الاتجاهية هي عملية تحويل السلاسل النصية المكوَّنة من الكلمات أو العبارات (النص) إلى متَّجَه متجانس من الأرقام الحقيقية يستخدم لترميز خصائص النص باستخدام تنسيق تفهمه خوارزميات تعلُّم الآلة.



شكل 3.5: تمثيل حقيبة الكلمات (bag-of-words)

يستخدم المقطع البرمجي التالي أداة CountVectorizer لتمثيل مجموعة بيانات التدريب IMDb بالمتَّجَهات:

```
from sklearn.feature_extraction.text import CountVectorizer

# the min_df parameter is used to ignore terms that appear in less than 10 reviews.
vectorizer_v1 = CountVectorizer(min_df=10)

vectorizer_v1.fit(X_train_text) # fit the vectorizer on the training data.
# use the fitted vectorizer to vectorize the data.
X_train_v1 = vectorizer_v1.transform(X_train_text)

X_train_v1
```

```
<40000x23392 sparse matrix of type '<class 'numpy.int64'>'
        with 5301561 stored elements in Compressed Sparse Row format>
```

```
# expand the sparse data into a sparse matrix format, where each column represents a different word.
X_train_v1_dense=pd.DataFrame(X_train_v1.toarray(),
                              columns=vectorizer_v1.get_feature_names_out())
X_train_v1_dense
```

	00	000	007	01	02	04	05	06	07	08	...	zoo	zoom	zooming	zooms	zorro	zu	zucco	zucker	zulu	über
0	0	0	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	0	0	0
2	0	0	0	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	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
39995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
39996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
39997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
39998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
39999	0	2	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

40000 rows × 23392 columns

شكل 3.6: تمثيل مجموعة بيانات التدريب بالمتَّجهات

يُعبِّر هذا التنسيق **الكثيف** (Dense) للمصفوفة عن 40,000 تقييم ومراجعة فلم في بيانات التدريب. تحتوي المصفوفة على عمود لكل كلمة تظهر في 10 مراجعات على الأقل (مُنفذة بواسطة المتغير min_df). كما يتضح بالأعلى، ينتج عن ذلك 23,392 عمودًا، مرتبة في ترتيب أبجدي رقمي. يُعبِّر مُدخَل المصفوفة في الموضع [i،j] عن عدد المرات التي تظهر فيها كلمة j في المراجعة i. وعلى الرغم من إمكانيّة استخدام هذه المصفوفة مباشرةً من قِبَل خوارزمية التعلُّم الموجَّه، إلا أنها غير فعّالة من حيث استخدام الذاكرة. والسبب في ذلك أن الغالبية العظمى من المُدخَلات في هذه المصفوفة تساوي 0. وهذا يحدث لأن نسبة ضئيلة جدًا فقط من بين 23,392 كلمة محتملة ستظهر فعليًا في كل مراجعة. ولمعالجة هذا القصور، تُخزِّن أداة CountVectorizer البيانات الممثلة بالمتَّجهات في مصفوفة متباعدة، حيث تحتفظ فقط بالمُدخَلات غير الصفرية في كل عمود. يستخدم المقطع البرمجي بالأسفل الدالة getsizeof() التي تحدد حجم الكائنات في لغة **البايثون** (Python) **بالبايت** (Bytes) لتوضيح مدى التوفير في الذاكرة عند استخدام المصفوفة المتباعدة لبيانات IMDb:

```
from sys import getsizeof
print('\nMegaBytes of RAM memory used by the raw text format:',
       getsizeof(X_train_text)/1000000)
print('\nMegaBytes of RAM memory used by the dense matrix format:',
       getsizeof(X_train_v1_dense)/1000000)
print('\nMegaBytes of RAM memory used by the sparse format:',
       getsizeof(X_train_v1)/1000000)
```

```
MegaBytes of RAM memory used by the raw text format: 54.864133

MegaBytes of RAM memory used by the dense matrix format: 7485.440144

MegaBytes of RAM memory used by the sparse format: 4.8e-05
```

وبحسب المتوقّع تحتاج المصفوفة المتباعدة إلى ذاكرة أقل بكثير وتحديدًا 0.000048 ميجابايت. بينما تشغل المصفوفة الكثيفة 7 جيجابايت، كما أنّ هذه المصفوفة لن تُستخدَم مرة أخرى وبالتالي يمكن حذفها لتوفير هذا الحجم الكبير من الذاكرة:

```
# delete the dense matrix.
del X_train_v1_dense
```

بِناء خط أنابيب التنبؤ

Build a Prediction Pipeline

الآن بعد أن تمكَّنت من تمثيل بيانات التدريب بالمتَّجهات فإن الخطوة التالية هي بِناء خط أنابيب التنبؤ الأول. أحد الأمثلة على المُصنِّفات المُستخدَمة للتنبؤ بالنَّص هو **المُصنِّف بايز الساذج** (Naive Bayes Classifier). يَستخدِم هذا المصنِّف احتمالات الكلمات أو العبارات المحددة الواردة في النَّص للتنبؤ باحتمال انتمائه إلى تصنيف محدد. جاءت كلمة الساذج (Naive) في اسم المُصنِّف من افتراض أن وجود كلمة بعينها في النَّص مستقل عن وجود أي كلمة أخرى. وهذا افتراض قوي، ولكنه يسمح بتدريب الخوارزمية بسرعة وبفعالية كبيرة.

المُصنِّف (Classifier):
المُصنِّف في تعلُّم الآلة هو نموذج يُستخدم لتمييز نقاط البيانات في فئات أو تصنيفات مختلفة. الهدف من المُصنِّف هو التعلُّم من بيانات التدريب المُعنوَنة، ومن ثم القيام بالتنبؤات حول قيم التصنيف لبيانات جديدة.

يستخدم المقطع البرمجي التالي تطبيق **مصنِّف بايز الساذج** (Multinomial NB) من **مكتبة سكليرن** (Sklearn Library) لتدريب نموذج التعلُّم الموجَّه على بيانات التدريب IMDb بالمتَّجهات:

```
from sklearn.naive_bayes import MultinomialNB

model_v1=MultinomialNB() # a Naive Bayes Classifier

model_v1.fit(X_train_v1, Y_train) # fit the classifier on the vectorized training data.

from sklearn.pipeline import make_pipeline

# create a prediction pipeline: first vectorize using vectorizer_v1, then use model_v1 to predict.
prediction_pipeline_v1 = make_pipeline(vectorizer_v1, model_v1)
```

على سبيل المثال، سيُنتج هذا المقطع البرمجي مصفوفة نتائج يرمز فيها الرقم 1 للتقييم الإيجابي و0 للتقييم السلبي:

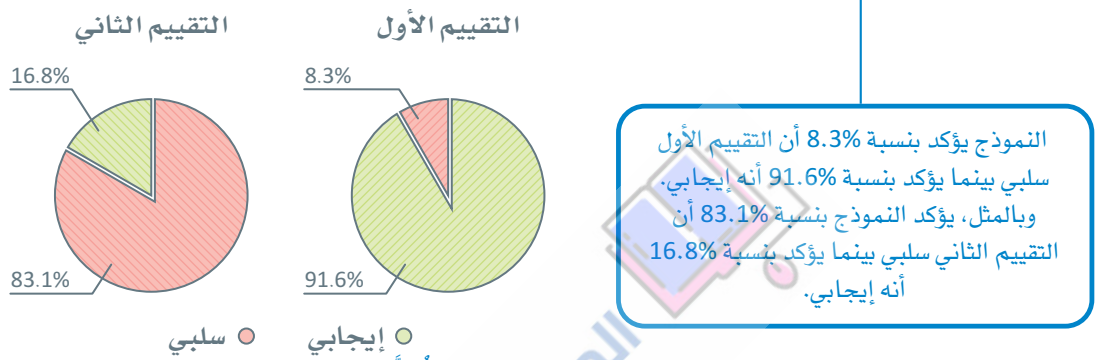
```
prediction_pipeline_v1.predict(['One of the best movies of the year. Excellent
                                 cast and very interesting plot.',
                                'I was very disappointed with his film. I
                                 lost all interest after 30 minutes' ])
```

```
array([1, 0], dtype=int64)
```

يتنبأ خط الأنابيب بشكل صحيح بالقيمة الإيجابية والسلبية للتقييمين الأول والثاني على التوالي. يُمكن استخدام الدالة المُضمّنة predict_proba( ) لتحديد جميع الاحتمالات التي يقوم خط الأنابيب بتخصيصها لكل واحدة من القيمتين المحتملتين. العنصر الأول هو احتمال تعيين 0 والعنصر الثاني هو احتمال تعيين 1:

```
prediction_pipeline_v1.predict_proba(['One of the best movies of the year. Ex
                                      cellent cast and very interesting plot.',
                                      'I was very disappointed with his film.
                                       I lost all interest after 30 minutes' ])
```

```
array([[0.08310769, 0.91689231],
       [0.83173475, 0.16826525]])
```



شكل 3.7: مُخطَّطان دائريان يوضحان النسب المئوية للتقييمين

الخطوة التالية هي اختبار دقة خط الأنابيب الجديد في تصنيف التقييمات في مجموعة بيانات اختبار IMDb. المُخرَج هو مصفوفة تشمل جميع قيم نتائج تصنيف التقييمات الواردة في بيانات الاختبار:

```
# use the pipeline to predict the labels of the testing data.
predictions_v1 = prediction_pipeline_v1.predict(X_test_text) # vectorize the text
data, then predict.

predictions_v1
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

توفر لغة البايثون العديد من الأدوات لتحليل وتصوير نتائج خطوط أنابيب التصنيف. تشمل الأمثلة دالة accuracy_score( ) من مكتبة سكليرن وتمثيل **مصفوفة الدقة** (Confusion Matrix) من مكتبة **سايكيت بلوت** (Scikit-Plot)، وهناك مقاييس تقييم أخرى مثل: الدقة، والاستدعاء، والنوعية، والحساسية، ومقياس درجة F1، وفقًا لحالة الاستخدام التي يمكن حسابها من مصفوفة الدقة. المُخرَج التالي هو تقريب دقيق لدرجة التنبؤ:

```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, predictions_v1) # get the achieved accuracy.
```

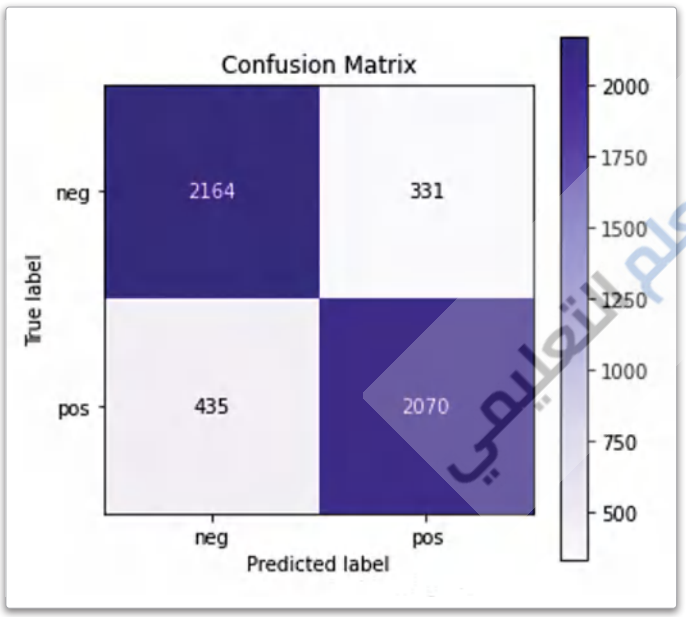
```
0.8468
```

```
%%capture
!pip install scikit-plot; # install the scikit-plot library, if it is missing.
import scikitplot; # import the library

class_names=['neg','pos'] # pick intuitive names for the 0 and 1 labels.

# plot the confusion matrix.
scikitplot.metrics.plot_confusion_matrix(
                                [class_names[i] for i in Y_test],
                                [class_names[i] for i in predictions_v1],
                                 title="Confusion Matrix", # title to use
                                 cmap="Purples", # color palette to use
                                 figsize=(5,5) # figure size
                                 );
```

القيم الحقيقية.

القيم المُتوقَّعة.

تحتوي مصفوفة الدقة على عدد التصنيفات الحقيقية مقابل المُتوقَّعة. في مُهِمَّة التصنيف الثنائية (مثل: مسألة احتواء قيمتين، الموجودة في مُهِمَّة IMDb)، ستحتوي مصفوفة الدقة على أربع خلايا:



- **التنبؤات السالبة الصحيحة (أعلى اليسار):** عدد المرات التي تنبأ فيها المُصنِّف بالحالات السالبة بشكل صحيح.
- **التنبؤات السالبة الخاطئة (أعلى اليمين):** عدد المرات التي تنبأ فيها المُصنِّف بالحالات السالبة بشكل خاطئ.
- **التنبؤات الموجبة الخاطئة (أسفل اليسار):** عدد المرات التي تنبأ فيها المُصنِّف بالحالات الموجبة بشكل خاطئ.
- **التنبؤات الموجبة الصحيحة (أسفل اليمين):** عدد المرات التي تنبأ فيها المُصنِّف بالحالات الموجبة بشكل صحيح.

شكل 3.8: نتائج مصفوفة الدقة بتطبيق مصنِّف بايز الساذج على بيانات الاختبار باستخدام مجموعة بيانات IMDb.

تُظهر النتائج أنه على الرغم من أن خط الأنابيب الأول يحقق دقة تنافسية تصل إلى 84.68%، إلا أنه لا يزال يُخطئ في تصنيف مئات التقييمات. فهناك 331 تنبّؤًا غير صحيح في الربع الأيمن العلوي و435 تنبّؤًا غير صحيح في الربع الأيسر السفلي. بإجمالي 766 تنبّؤًا غير صحيح. الخطوة الأولى نحو تحسين الأداء هي دراسة سلوك خط أنابيب التنبؤ، لمعرفة كيف يقوم بمعالجة النصّ وفهمه.

الدقة (Accuracy):

الدقة هي نسبة التنبؤات الصحيحة إلى إجمالي عدد التنبؤات.

$$\text{الدقة} = \frac{(\text{التنبؤات الموجبة الصحيحة} + \text{التنبؤات السالبة الصحيحة})}{(\text{التنبؤات الموجبة الصحيحة} + \text{التنبؤات السالبة الصحيحة} + \text{التنبؤات الموجبة الخاطئة} + \text{التنبؤات السالبة الخاطئة})}$$

شرح مُتنبِّئات الصندوق الأسود Explaining Black-Box Predictors

يستخدم مصنَّف بايز الساذج الصيغ الرياضية البسيطة لتجميع احتمالات آلاف الكلمات وتقديم تنبؤاتها. وبالرغم من بساطة النموذج، إلا أنه لا يزال غير قادر على تقديم شرح بسيط ومباشر لكيفية قيام النموذج بتوقُّع القيمة الموجبة أو السالبة لجزء محدد من النص. قارِن ذلك مع مُصنِّفات شجرة القرار الأكثر وضوحًا، حيث يتم تمثيل القواعد التي تعلمها النموذج في الهيكل الشجري، مما يُسهِّل على الأشخاص فهم كيف يقوم المُصنِّف بالتنبؤات. يتيح هيكل الشجرة كذلك الحصول على تصور مرئي للقرارات المُتخذَّة في كل فرع، ممّا يكون مفيدًا في فهم العلاقات بين الخصائص المُدخَلة والمتغير المستهدف.

الافتقار إلى قدرة التفسير تمثل تحديًا كبيرًا في الخوارزميات الأكثر تعقيدًا، كتلك المُستنِدة إلى التجميعات مثل: توليفات من الخوارزميات المتعددة أو الشبكات العصبية. فبدون القدرة على التفسير، تتقلص خوارزميات التعلُّم الموجَّه إلى متنبئات الصندوق الأسود: على الرغم من أنها تفهم النص بشكل كافٍ للتنبؤ بالقيم، إلا أنها لا تزال غير قادرة على تفسير كيف تقوم باتخاذ القرار. أجريت العديد من الأبحاث للتغلب على هذه التحديات بتصميم وسائل قادرة على التفسير تستطيع فهم نماذج الصندوق الأسود. واحدة من الوسائل الأكثر شهرة هي **النموذج المحايد المحلي القابل للتفسير والشرح** (Local Interpretable Model-Agnostic Explanations – LIME).

النموذج المحايد المحلي القابل للتفسير والشرح
Local Interpretable Model-Agnostic Explanations - LIME

النموذج المحايد المحلي القابل للتفسير والشرح (LIME) هو طريقة لتفسير التنبؤات التي قامت بها نماذج الصندوق الأسود. وذلك من خلال النظر في نقطة بيانات واحدة في وقت محدد، وإجراء تغييرات بسيطة عليها لمعرفة كيف يؤثر ذلك على قدرة تنبؤ النموذج، ثم تُستخدم هذه المعلومات لتدريب نموذج مفهوم وبسيط مثل الانحدار الخطي على تفسير هذه التنبؤات. بالنسبة للبيانات النصية، يقوم النموذج المحايد المحلي القابل للتفسير والشرح بالتعرّف على الكلمات أو العبارات التي لها الأثر الأكبر على القيام بالتنبؤات.

وفيما يلي، تطبيق بلغة البايثون يوضّح ذلك:

```
%%capture

!pip install lime # install the lime library, if it is missing
from lime.lime_text import LimeTextExplainer

# create a local explainer for explaining individual predictions
explainer_v1 = LimeTextExplainer(class_names=class_names)

# an example of an obviously negative review
easy_example='This movie was horrible. The actors were terrible and the plot
was very boring.'

# use the prediction pipeline to get the prediction probabilities for this example
print(prediction_pipeline_v1.predict_proba([easy_example]))
```

```
[[0.99874831 0.00125169]]
```

كما هو مُتوقَّع، يقدم نموذج التنبؤ تنبؤًا سلبيًا مؤكدًا بدرجة كبيرة في هذا المثال البسيط.

```
# explain the prediction for this example.
exp = explainer_v1.explain_instance(easy_example.lower(),
                                    prediction_pipeline_v1.predict_proba,
                                    num_features=10)
# print the words with the strongest influence on the prediction.
exp.as_list()
```

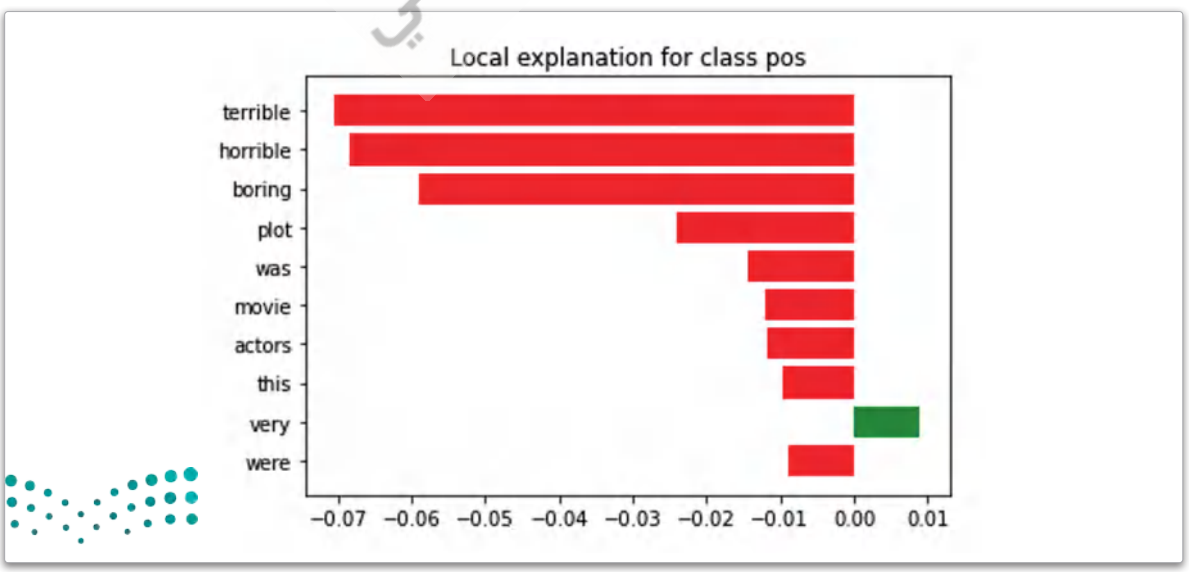
```
[('terrible', -0.07046118794796816),
 ('horrible', -0.06841672591649835),
 ('boring', -0.05909016205135171),
 ('plot', -0.024063095577996376),
 ('was', -0.014436071624747861),
 ('movie', -0.011956911011210977),
 ('actors', -0.011682594571408675),
 ('this', -0.009712387273986628),
 ('very', 0.008956707731803237),
 ('were', -0.008897098392433257)]
```

الخصائص العشرة الأكثر تأثيرًا.

الدرجة المقابلة لكل كلمة تمثل معاملًا في نموذج الانحدار الخطي البسيط المُستخدَم لتقديم التفسير.

يمكن الحصول على تصور مرئي أكثر دقةً على النحو التالي:

```
# visualize the impact of the most influential words.
fig = exp.as_pyplot_figure()
```

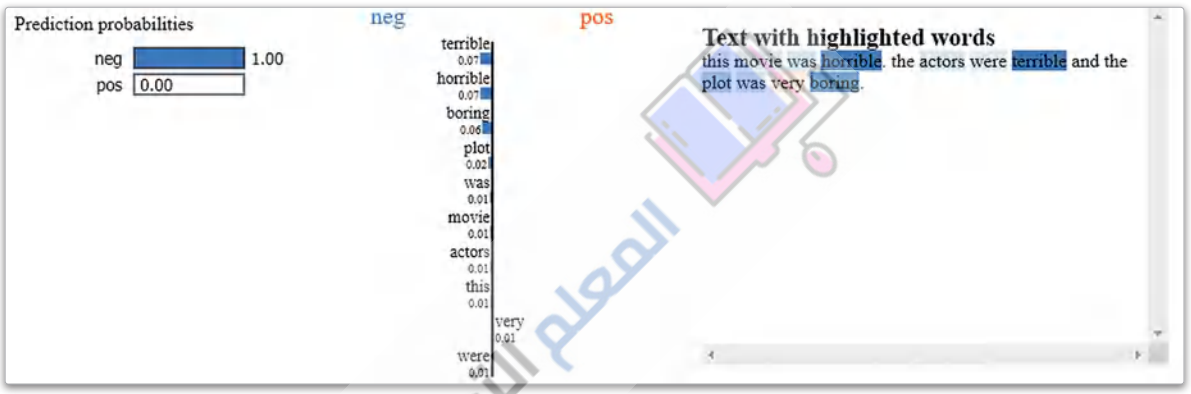


شكل 3.9: الكلمات الأعلى تأثيرًا في القيام بالتنبؤات

يَزيد المُعامِل السالب من احتمالية التصنيف السالب، بينما يُقلل المُعامِل الموجب منه. على سبيل المثال، الكلمات: horrible (فظيع)، و terrible (مريع)، و boring (ممل) لها التأثير الأقوى على قرار النموذج بالتنبؤ بالقيمة السالبة. الكلمة very (جدًا) دفعت النموذج قليلًا في اتجاه آخر إيجابي، ولكنها لم تكن كافية لتغيير القرار. بالنسبة للمراقب البشري، قد يبدو غريبًا أن الكلمات الخالية من المشاعر مثل: plot (الحبكة الدرامية) أو was (كان) لها مُعامِلات مرتفعة نسبيًا. ومع ذلك، من الضروري أن تتذكر أن تعلُّم الآلة لا يتبع دومًا الوعي البشري السليم.

وقد تكشف هذه المُعامِلات المرتفعة بالفعل عن قصور في منطق الخوارزمية وقد تكون مسؤولة عن بعض أخطاء نموذج التنبُّؤ. وعلى نحوٍ بديل، يُعدُّ نموذج التنبُّؤ بمثابة مؤشر على الأنماط التنبؤية الكامنة والغنيّة في الوقت نفسه بالمعلومات. على سبيل المثال، قد يبدو الواقع وكأن المُقيّمين البشريين أكثر استخدامًا لكلمة plot (الحبكة الدرامية) أو صيغة الماضي was (كان) عند الحديث في سياق سلبي. ويمكن لمكتبة **النموذج المحايد المحلي القابل للتفسير والشرح** (LIME) في لغة البايثون تصوير الشروحات بطرائق أخرى. على سبيل المثال:

```
exp.show_in_notebook()
```



شكل 3.10: التمثيلات المرئية الأخرى

التقييم المُستخدم في المثال السابق كان سلبيًا بشكل واضح ويَسهُل التنبؤ به. خُذْ بعين الاعتبار التقييم التالي الأكثر صعوبةً والذي يمكن أن يتسبب في تذبذب دقة الخوارزمية، وهو مأخوذ من مجموعة بيانات اختبار IMDb:

```
# an example of a positive review that is mis-classified as negative by prediction_pipeline_v1
mistake_example= X_test_text[4600]
mistake_example
```

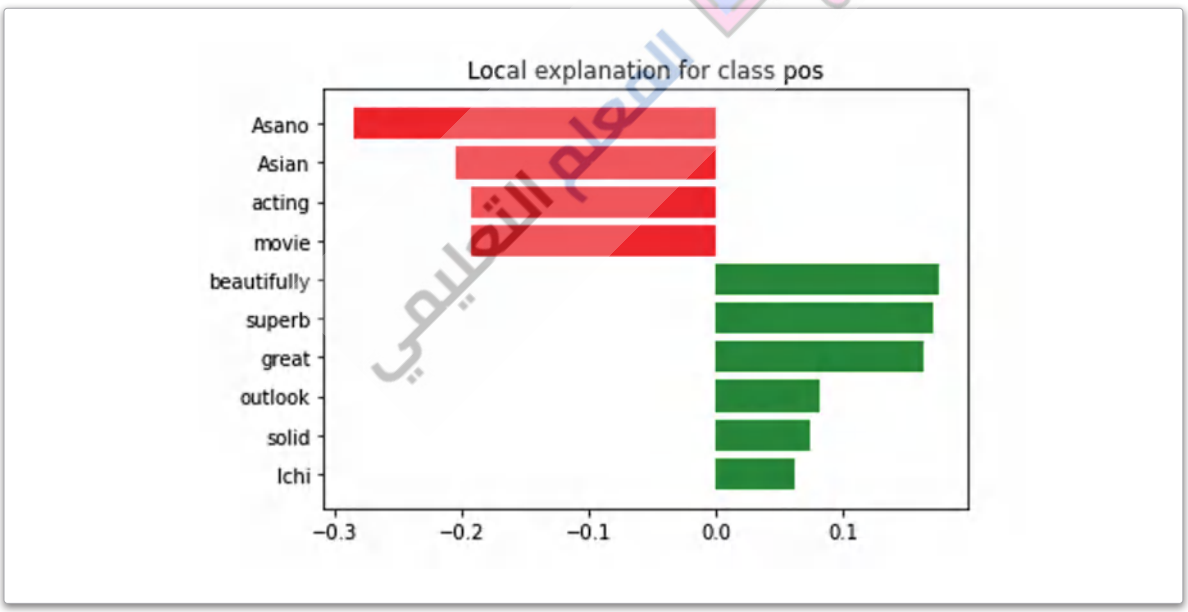
```
"I personally thought the movie was pretty good, very good acting by
Tadanobu Asano of Ichi the Killer fame. I really can't say much about the
story, but there were parts that confused me a little too much, and overall
I thought the movie was just too lengthy. Other than that however, the
movie contained superb acting great fighting and a lot of the locations
were beautifully shot, great effects, and a lot of sword play. Another
solid effort by Tadanobu Asano in my opinion. Well I really can't say
anymore about the movie, but if you're only outlook on Asian cinema is
Crouching Tiger Hidden Dragon or House of Flying Daggers, I would suggest
you trying to rent it, but if you're a die-hard Asian cinema fan I would
say this has to be in your collection very good Japanese film."
```

```
# get the correct labels of this example.
print('Correct Label:', class_names[Y_test[4600]])

# get the prediction probabilities for this example.
print('Prediction Probabilities for neg, pos:',
      prediction_pipeline_v1.predict_proba([mistake_example]))
```

```
Correct Label: pos
Prediction Probabilities for neg, pos: [[0.8367931 0.1632069]]
```

على الرغم من أن هذا التقييم إيجابي بشكل واضح، إلا أنّ نموذج التنبُّؤ قدّم تنبؤًا سلبيًا مؤكدًا للغاية باحتمالية وصلت إلى 83%. يمكن الآن استخدام المُفسِّر لتوضيح السبب وراء اتخاذ نموذج التنبُّؤ مثل هذا القرار الخاطئ:

```
# explain the prediction for this example.
exp = explainer_v1.explain_instance(mistake_example, prediction_pipeline_
v1.predict_proba, num_features=10)

# visualize the explanation.
fig = exp.as_pyplot_figure()
```



شكل 3.11: الكلمات التي أثرت على القرار الخاطئ

على الرغم من أن نموذج التنبُّؤ يستنبط التأثير الإيجابي لبعض الكلمات على نحوٍ صحيح مثل:

beautifully (بشكل جميل)، وgreat (رائع)، وsuperb (مدهش)، إلا أنّه يتّخذ في النهاية قرارًا سلبيًا استنادًا إلى العديد من الكلمات التي يبدو أنها لا تعبر بشكل واضح عن المشاعر السلبية مثل: Asano (أسانو)، وAsian (آسيوي)، وmovie (فيلم)، وacting (تمثيل).

وهذا يوضِّح العيوب الكبيرة في المنطِق الذي يستخدمه نموذج التنبُّؤ لتصنيف المفردات الواردة في نصوص التقييمات المُقدمة. القسم التالي يوضّح كيف أن تحسين هذا المنطق يمكن أن يطور من أداء نموذج التنبُّؤ إلى حدٍ كبير.

تحسين البرمجة الاتجاهية للنصوص

Improving Text Vectorization

استخدم الإصدار الأول لخط أنابيب التنبؤ أداة CountVectorizer لحساب عدد المرات التي تظهر فيها كل كلمة في كل تقييم. تتجاهل هذه المنهجية حقيقتين أساسيتين حول اللغات البشرية:

- قد يتغير معنى الكلمة وأهميّتها حسب الكلمات المُستخدَمة معها.
- تكرار الكلمة في المُستنَد لا يُعدُّ دومًا تمثيلًا دقيقًا لأهميّتها. على سبيل المثال، على الرغم من أن تكرار كلمة great (رائع) مرتين قد يمثل مؤشرًا إيجابيًا في مستند يحتوي على 100 كلمة، إلا أنه يمثل مؤشرًا أقل أهمية بكثير في مستند يحتوي على 1000 كلمة.

التعبير النمطي (Regular Expression):
التعبير النمطي هو نمط نص يُستخدَم لمطابقة ولمعالجة سلاسل النصوص وتقديم طريقة موجزة ومرنة لتحديد أنماط النصوص، كما تُستخدَم على نطاق واسع في معالجة النصوص وتحليل البيانات.

سيشرح هذا الجزء كيفية تحسين البرمجة الاتجاهية للنصوص لأخذ هاتين الحقيقتين في عين الاعتبار. يستدعي المقطع البرمجي التالي ثلاثة مكتبات مختلفة بلغة البايثون، ستُستخدم لتحقيق ذلك:

- **nltk** و **جينسم** (Gensim): تُستَخدم هاتان المكتبتان الشّهيرتان في مهام معالجة اللغات الطبيعية المُتنوّعة.
- **re**: تُستَخدم هذه المكتبة في البحث عن النّصوص، ومعالجتها باستخدام التعبيرات النمطية.

```
%%capture

!pip install nltk # install nltk
!pip install gensim # install gensim

import nltk # import nltk
nltk.download('punkt') # install nltk's tokenization tool, used to split a text into sentences.

import re  # import re

from gensim.models.phrases import Phrases, ENGLISH_CONNECTOR_WORDS # import tools
from the gensim library.
```

تحديد العبارات Detecting Phrases

التقسيم (Tokenization):
يقصد به: عملية تقسيم البيانات النصية إلى أجزاء مثل كلمات، وجُمل، ورموز، وعناصر أخرى يطلق عليها الرموز.

يمكن استخدام الدالة الآتية لتقسيم مستند محدد إلى قائمة من الجُمل المُقسَّمة، حيث يمكن تمثيل كل جملة مُقسَّمة بقائمة من الكلمات:

```
# convert a given doc to a list of tokenized sentences.
def tokenize_doc(doc:str):
    return [re.findall(r'\b\w+\b',
            sent.lower()) for sent in nltk.sent_tokenize(doc)]
```

دالة ( )sent_tokenize تُقسِّم المُستنَد إلى قائمة من الجُمل.

دالة ( )sent_tokenize من مكتبة nltk تُقسِّم المُستنَد إلى قائمة من الجُمل.

بعد ذلك، يتم كتابة كل جملة بأحرف صغيرة وتغذيتها إلى دالة ( )findall من مكتبة re لتقوم بتحديد تكرارات التعبيرات النمطية '\b\w+\b'. ستختبرها على السلسلة النصية الموجودة في متغير raw_text . في هذا السياق

- \w تتطابق مع كل الرموز الأبجدية الرقمية (0-9، A-Z، a-z) والشَرطة السفلية.
- \w+ تُستَخدم للبحث عن واحد أو أكثر من رموز \w. لذلك، في السلسلة النصية hello123_world (مرحبًا 123_ العالم)، النمط \w+ سيتطابق مع الكلمات hello (مرحبًا) و 123 وworld (العالم).
- \b تمثل الفاصل (boundry) بين رمز \w ورمز ليس \w ،وكذلك في بداية أو نهاية السلسلة النصية المُعطاة. على سبيل المثال: سوف يتطابق النمط \bcat\b مع الكلمة cat (القطة) في السلسلة النصية The cat is cute (القطة لطيفة)، ولكنه لن يتطابق مع الكلمة cat (القطة) في السلسلة النصية The category is pets (فئة الحيوانات الأليفة).

أدناه مثالًا على التقسيم باستخدام الدالة ()tokenize_doc.

```
raw_text='The movie was too long. I fell asleep after the first 2 hours.'
tokenized_sentences=tokenize_doc(raw_text)
tokenized_sentences
```

```
[['the', 'movie', 'was', 'too', 'long'],
 ['i', 'fell', 'asleep', 'after', 'the', 'first', '2', 'hours']]
```

يمكن الآن تجميع الدالة ()tokenize_doc مع أداة العبارات من مكتبة **جينسم** (Gensim) لإنشاء نموذج العبارة، وهو نموذج يمكنه التعرّف على العبارات المكونة من عدة كلمات في جملة معطاة. يستخدم المقطع البرمجي التالي بيانات التدريب IMDB الخاصة بـ (X_train_text) لبناء مثل هذا النموذج:

```
sentences=[] # list of all the tokenized sentences across all the docs in this dataset

for doc in X_train_text: # for each doc in this dataset
    sentences+=tokenize_doc(doc) # get the list of tokenized sentences in this doc

# build  a phrase model on the given data
imdb_phrase_model = Phrases(sentences,1
                            connector_words=ENGLISH_CONNECTOR_WORDS,2
                            scoring='npmi',3
                            threshold=0.25).freeze()4
```

كما هو موضح بالأعلى، تستقبل الدالة ()Phrases أربعة متغيرات:

1. قائمة الجُمل المُقسَّمة من مجموعة النصوص المُعطاة.
2. قائمة بالكلمات الإنجليزية الشائعة التي تظهر بصورة متكررة في العبارات (مثل: the، و of)، وليس لها أي قيمة موجبة أو سالبة، ولكن يمكنها إضفاء المشاعر حسب السياق، ولذلك يتم التعامل معها بصورة مختلفة.
3. تُستَخدم دالة تسجيل النقاط لتحديد ما إذا كان تضمين مجموعة من الكلمات في العبارة نفسها واجبًا. المقطع البرمجي بالأعلى يَستخدم مقياس **المعلومات النقطية المشتركة المُعايَر** (Normalized Pointwise Mutual Information – NPMI) لهذا الغرض. يستند هذا المقياس على تكرار توارد الكلمات في العبارة المُرشحة وتكون قيمته بين 1- و يرمز إلى **الاستقلالية الكاملة** (Complete Independence)، و1+ ويرمز إلى **التوارد الكامل** (Complete Co-occurrence).
4. في حدود دالة تسجيل النقاط يتم تجاهل العبارات ذات النقاط الأقل. ومن الناحية العملية، يمكن ضبط هذه الحدود لتحديد القيمة التي تُعطي أفضل النتائج في التطبيقات النهائية مثل: النمذجة التنبؤية.

تُحوِّل دالة ()freeze نموذج العبارة إلى تنسيق غير قابل للتغيير أي مُجمّد (Frozen) لكنّه أكثر سرعة.

عند تطبيقها على الجملتين المُقسَّمتين بالمثال المُوضح بالأعلى، سيُحقق نموذج العبارة النتائج التالية:

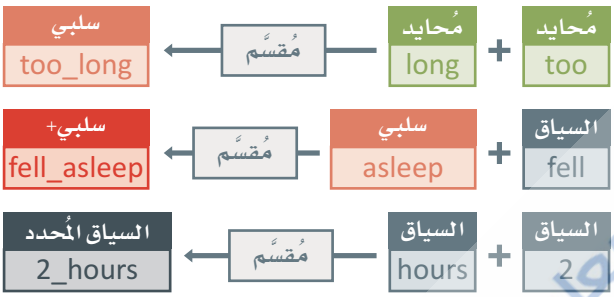
```
imdb_phrase_model[tokenized_sentences[0]]
```

```
['the', 'movie', 'was', 'too_long']
```

```
imdb_phrase_model[tokenized_sentences[1]]
```

```
['i', 'fell_asleep', 'after', 'the', 'first', '2_hours']
```

يحدِّد نموذج العبارة ثلاثة عبارات على النحو التالي: fell_asleep (سقط نائمًا) وtoo_long (طويل جدًا)، و2_hours (2-ساعة) وجميعها تحمل معلومات أكثر من كلماتها المفردة.



شكل 3.12: المشاعر الإيجابية والسلبية قبل التقسيم وبعده

على سبيل المثال، تحمل عبارة too_long (طويل جدًا) مشاعر سلبية واضحة، على الرغم من أن كلمتي too (جدًا) وlong (طويل) لا تعبران عن ذلك منفردتين، وبالمثل، فعلى الرغم من أن كلمة asleep (نائم) في مراجعة الفيلم تمثل دلالة سلبية، فالعبارة fell_asleep (سقط نائمًا) توصل رسالة أكثر وضوحًا. وأخيرًا، تستنبط من 2_hours (2-ساعة) سياقًا أكثر تحديدًا من الكلمتين 2 وhours كلٍّ على حدة.

تستخدم الدالة التالية إمكانية تحديد العبارات بهذا الشكل لتفسير العبارات في وثيقة مُعطاه:

```
def annotate_phrases(doc:str, phrase_model):

    sentences=tokenize_doc(doc)# split the document into tokenized sentences.

    tokens=[] # list of all the words and phrases found in the doc
    for sentence in sentences: # for each sentence
        # use the phrase model to get tokens and append them to the list.
        tokens+=phrase_model[sentence]
    return ' '.join(tokens) # join all the tokens together to create a new annotated document.
```

يستخدم المقطع البرمجي التالي دالة annotate_phrases( ) لتفسير كلٍ من تقييمات التدريب والاختبار من مجموعة بيانات IMDb.

```
# annotate all the test and train reviews.
X_train_text_annotated=[annotate_phrases(doc,imdb_phrase_model) for doc in X_
train_text]
X_test_text_annotated=[annotate_phrases(text,imdb_phrase_model)for text in X_
test_text]
```

```
# an example of an annotated document from the imdb training data
X_train_text_annotated[0]
```

```
'i_grew up b 1965 watching and loving the thunderbirds all my_mates at school watched
we played thunderbirds before school during lunch and after school we all wanted to
be virgil or scott no_one wanted to be alan counting down from 5 became an art_form
i took my children to see the movie hoping they would get_a_glimpse of what i_loved
as a child how bitterly disappointing the only high_point was the snappy theme_tune
not that it could compare with the original score of the thunderbirds thankfully
early saturday_mornings one television_channel still plays reruns of the series
gerry_anderson and his_wife created jonatha frakes should hand in his directors chair
his version was completely hopeless a waste of film utter_rubbish a cgi remake may_be
acceptable but replacing marionettes with homo_sapiens subsp sapiens was a huge error
of judgment'
```

استخدام مقياس تكرار المصطلح-تكرار المُستنَد العكسي في البرمجة الاتجاهية للنصوص
Using TF-IDF for Text Vectorization

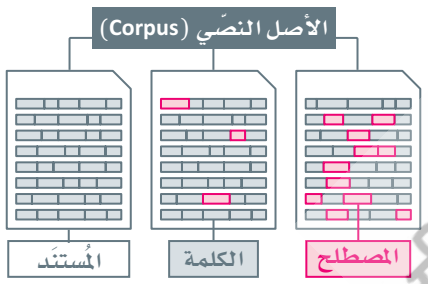
تكرار الكلمة في المُستنَد لا يُعدُّ دومًا تمثيلًا دقيقًا لأهميتها. الطريقة المُثلى لتمثيل التكرار هي المقياس الشهير **لتكرار المصطلح - تكرار المُستنَد العكسي** (TF-IDF). يستخدم هذا المقياس صيغة رياضية بسيطة لتحديد أهمية الرموز مثل: الكلمات أو العبارات في المُستنَد بناءً على عاملين:

- تكرار الرمز في المُستنَد، بقياس عدد مرات ظهوره في المُستنَد مقسومًا على إجمالي عدد الرموز في جميع المُستنَدات.
- تكرار المُستنَد العكسي للرمز، المحسوب بقسمة إجمالي عدد المُستنَدات في مجموعة البيانات على عدد المُستنَدات التي تحتوي على الرمز.

العامل الأول يتجنب المبالغة في تقدير أهمية المصطلحات التي تظهر في الوثائق الأطول، أمّا العامل الثاني فيستبعد المصطلحات التي تظهر في كثيرٍ من المُستنَدات، مما يساعد على إثبات حقيقة أن بعض الكلمات هي أكثر شيوعًا من غيرها.

تكرار المصطلح - تكرار المستند العكسي
Term Frequency Inverse Document Frequency (TF-IDF)

تكرار المصطلح- تكرار المُستنَد العكسي هو طريقة تُستخدم لتحديد أهمية الرموز في المُستنَد.



شكل 3.13: الكلمات والمصطلحات الواردة في المستند

$$\text{تكرار المُستنَد العكسي} = \frac{\text{عدد المُستنَدات في الأصل النصّي}}{\text{عدد المُستنَدات التي تحتوي على المصطلح}}$$

$$\text{تكرار المصطلح} = \frac{\text{عدد مرات ظهور المصطلح في المُستنَد}}{\text{عدد الكلمات في المُستنَد}}$$

القيمة = تكرار المصطلح × تكرار المُستنَد العكسي

أداة TfidfVectorizer

توفر مكتبة **سكليرن** (Sklearn) أداة تدعم هذا النوع من البرمجة الاتجاهية **لتكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF). يمكن استخدام أداة TfidfVectorizer لتمثيل عبارة باستخدام المتَّجهات.

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Train a TF-IDF model with the IMDb training dataset
vectorizer_tf = TfidfVectorizer(min_df=10)
vectorizer_tf.fit(X_train_text_annotated)
X_train_tf = vectorizer_tf.transform(X_train_text_annotated)
```

يمكن الآن إدخال أداة التمثيل بالمتَّجهات في مُصنَّف بايز الساذج لبناء خط أنابيب نموذج تنبُّؤ جديد وتطبيقه على بيانات اختبار IMDb:

```
# train a new Naive Bayes Classifier on the newly vectorized data.
model_tf =MultinomialNB()
model_tf.fit(X_train_v2, Y_train)

# create a new prediction pipeline.
prediction_pipeline_tf = make_pipeline(vectorizer_tf, model_tf)

# get predictions using the new pipeline.
predictions_tf = prediction_pipeline_tf.predict(X_test_text_annotated)

# print the achieved accuracy.
accuracy_score(Y_test, predictions_tf)
```

```
0.8858
```

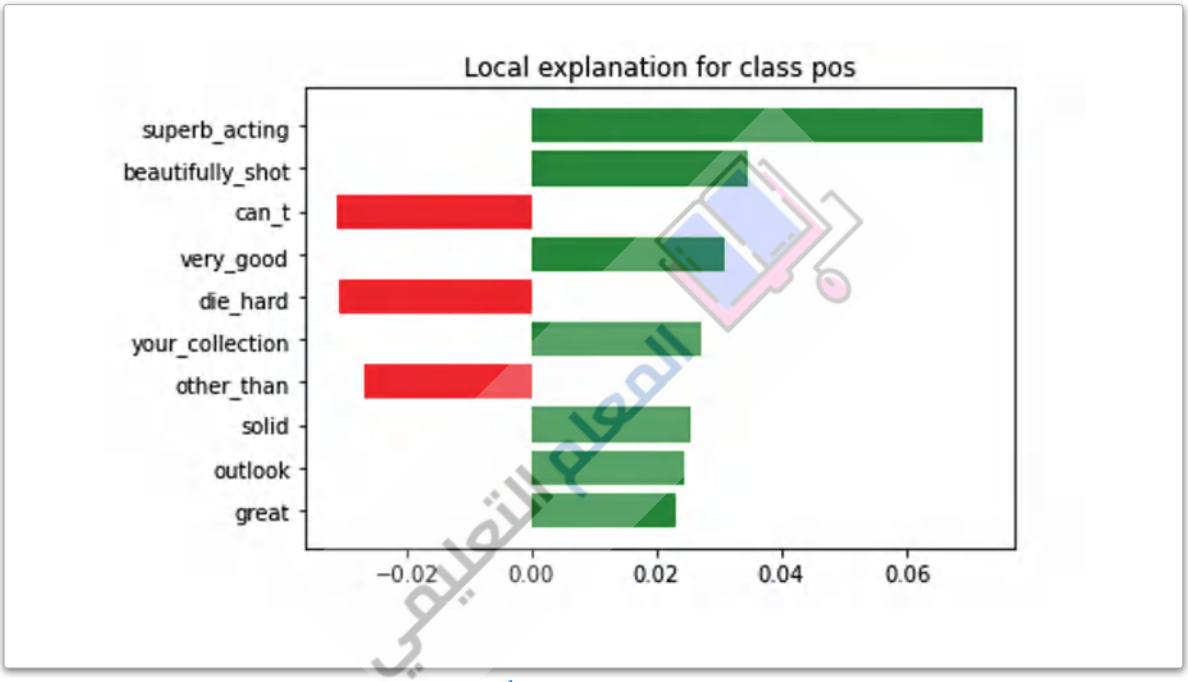
يحقق خط الأنابيب الجديد دقة تصل إلى 88.58%، وهو تحسُّن كبير بالمقارنة مع الدقة السابقة التي وصلت إلى 84.68%. يمكن الآن استخدام النموذج المُحسَّن لإعادة النظر في مثال الاختبار الذي تم تصنيفه بشكل خاطئ بواسطة النموذج الأول:

```
# get the review example that confused the previous algorithm
mistake_example_annotated=X_test_text_annotated[4600]

print('\nReview:',mistake_example_annotated)

# get the correct labels of this example.
print('\nCorrect Label:', class_names[Y_test[4600]])

# get the prediction probabilities for this example.
print('\nPrediction Probabilities for neg, pos:',prediction_pipeline_
tf.predict_proba([mistake_example_annotated]))
```

```
Review: i_personally thought the movie was_pretty good very_good acting by tadanobu_
asano of ichi_the_killer fame i really can_t say much about the story but there_were
parts that confused me a little_too much and overall i_thought the movie was just too
lengthy other_than that however the movie contained superb_acting great fighting and
a lot of the locations were beautifully_shot great effects and a lot of sword play
another solid effort by tadanobu_asano in my_opinion well i really can_t say anymore
about the movie but if_you re only outlook on asian_cinema is crouching_tiger hidden_
dragon or house of flying_daggers i_would suggest_you trying to rent_it but if_you re
a die_hard asian_cinema fan i_would say this has to be in your_collection very_good
japanese film

Correct Label: pos

Prediction Probabilities for neg, pos: [[0.32116538 0.67883462]]
```

يتنبأ خط الأنابيب الجديد بشكل صحيح بالقيمة الإيجابية لهذا التقييم. يَستخدم المقطع البرمجي التالي مُفسِّر **النموذج المحايد المحلي القابل للتفسير والشرح** (LIME) لتفسير المنطق وراء هذا التنبؤ:

```
# create an explainer.
explainer_tf = LimeTextExplainer(class_names=class_names)

# explain the prediction  of the second pipeline for this example.
exp = explainer_tf.explain_instance(mistake_example_annotated, prediction_
pipeline_tf.predict_proba, num_features=10)

# visualize the results.
fig = exp.as_pyplot_figure()
```



شكل 3.14: تأثير الكلمة في مزيج تكرار المصطلح- تكرار المُستنَد العكسي ومصنّف بايز الساذج

تؤكد النتائج أن خط الأنابيب الجديد يتبع منطقًا أكثر ذكاءً. فهو يُحدد بشكل صحيح المشاعر الإيجابية للعبارات مثل: beautifully_shot (لقطة _ جميلة)، و superb_acting (تمثيل_رائع)، وvery good (جيد جدًا)، ولا يمكن تضليله باستخدام الكلمات التي جعلت خط الأنابيب الأول يتنبأ بنتائج خاطئة.

يمكن تحسين أداء خط الأنابيب لنموذج التنبُّؤ بطرق متعددة، بإستبدال مصنف بايز البسيط بطرق أكثر تطورًا مع ضبط متغيراتها لزيادة احتمالاتها. وثمَّة خيار آخر يتلخص في استخدام تقنيات البرمجة الاتجاهية البديلة التي لا تستند إلى تكرار الرمز، مثل تضمين الكلمات و النصوص، وسيُستعرض ذلك في الدرس التالي.

تمرينات

1

حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي:	صحيحة	خاطئة
1. في التعلُّم الموجَّه، تُستخدم مجموعات البيانات المُعنوَنة لتدريب النموذج.	✓	○
2. البرمجة الاتجاهية هي تقنية لتحويل البيانات من تنسيق متَّجَه رقمي إلى بيانات أولية.	○	✓
3. تتطلب المصفوفة المتباعدة ذاكرة أقل بكثير من المصفوفة الكثيفة.	✓	○
4. تُستخدم خوارزمية مُصنَّف بايز الساذج لبناء خط أنابيب التنبؤ.	✓	○
5. تكرار الكلمة في المُستنَد يُعدُّ التمثيل الدقيق الوحيد لأهمية هذه الكلمة.	○	✓

2 اشرح لماذا تتطلب المصفوفة الكثيفة مساحة من الذاكرة أكبر من المصفوفة المتباعدة.

لأنه غير فعال من حيث استخدام الذاكرة والسبب يعود إلى أن الغالبية العظمى من الادخالات في هذه المصفوفة تساوي صفر.

3 حلِّل كيف يُستخدَم العاملان الرّياضيّان في تكرار المصطلح- تكرار المُستنَد العكسي (TF-IDF) لتحديد أهمية الكلمة في النص.

يستخدم مقياس تكرار المصطلح تكرار المستند (TF-IDF) صيغة رياضية بسيطة لتحديد أهمية الرموز في المستند بناء على عاملين هما:

-تكرار الرمز في المستند بقياس عدد مرات ظهوره في المستند مقوماً على إجمالي عدد الرموز في جميع المستندات.

-تكرار المستند العكسي للرمز المحسوب بقسمة إجمالي عدد المستندات في مجموعة البيانات على عدد المستندات التي تحتوي عل الرمز

4 لديك X_train_text وهي عبارة عن مصفوفة numPy تتضمن مستندًا واحدًا في كل صف. لديك كذلك مصفوفة ثانية Y_train تتضمن قيم المُستنَدات في X_train_text. أكمل المقطع البرمجي التالي بحيث يمكن استخدام تكرار المصطلح- تكرار المُستنَد العكسي (TF-IDF) لتمثيل البيانات بالمتَّجهات، وتدريب نموذج تصنيف MultinomialNB على الإصدار المُمَثَّل بالمتَّجهات، ثم تجميع أداة التمثيل بالمتَّجهات ونموذج التصنيف في خط أنابيب تنبؤ واحد:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X-train_text) # fits the vectorizer on the training data
X_train = vectorizer.Transform(X_train_text) # uses the fitted vectorizer to vectorize the data
model_MNB=MultinomialNB() # a Naive Bayes Classifier
model_MNB.fit(X_train, Y-train) # fits the classifier on the vectorized training data
prediction_pipeline = make_pipeline(Vectorizer, Model_MNB)
```

5 أكمل المقطع البرمجي التالي بحيث يمكنه بناء مُفسِّر نصوص النموذج المحايد المحلي القابل للتفسير والشرح (LIME) لخط أنابيب التنبؤ الذي قمت ببنائه في التدريب السابق، واستخدم المُفسِّر لتفسير التنبؤ على مثال لنصٍ آخر.

```
from Lime.lime_text import LimeTextExplainer
text_example="I really enjoyed this movie, the actors were excellent"
class_names=['neg','pos'] # creates a local explainer for explaining individual predictions
explainer = LimeTextExplainer(class_names=class_names) # explains the prediction for this example
exp = explainer.explain_instance(text_example.lower(),prediction_pipeline.explain instance,
num_features=10) # focuses the explainer on the 10 most influential features
print(exp.asflpyplotfifigure()) # prints the words with the highest influence on the prediction
```

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثاني

التعلُّم غير الموجَّه

استخدام التعلُّم غير الموجَّه لفهم النصوص

Unsupervised Learning to Understand Text

التعلُّم غير الموجَّه هو نوع من تعلُّم الآلة، يستخدم فيه النموذج بيانات غير مُعَنونة، حيثُ يُقدِّم له مجموعة من الأمثلة التي يتولى البحث فيها عن الأنماط والعلاقات بين البيانات من تلقاء نفسه. وفي سياق فهم النص، يمكن استخدام التعلُّم غير الموجَّه في تحديد الهياكل والأنماط الكامنة ضمن مجموعة بيانات المُستنَدات النصية. هناك العديد من التقنيات المختلفة التي يمكن استخدامها في التعلُّم غير الموجَّه للبيانات النصية، بما في ذلك **خوارزميات التجميع** (Clustering Algorithms)، و**تقنيات تقليص الأبعاد** (Dimensionality Reduction Techniques)، و**النماذج التوليدية** (Generative Models). تُستخدم خوارزميات التجميع لضم المُستنَدات المتشابهة معًا، بينما تُستخدم تقنيات تقليص الأبعاد لتقليص أبعاد البيانات وتحديد الخصائص الهامة. ومن ناحية أخرى، تُستخدم النماذج التوليدية لتعلُّم التوزيع الأساسي للبيانات وتوليد نص جديد مشابه لمجموعة البيانات الأصلية.

التعلُّم غير الموجَّه (Unsupervised Learning):
في التعلُّم غير الموجَّه، يُزوَّد النموذج بكميات كبيرة من البيانات غير المُعنوَنة ويتوجب عليه البحث عن الأنماط في البيانات غير المُتراكبة من خلال الملاحظة والتجميع.

تقليص الأبعاد (Dimensionality Reduction):
تقنية تقليص الأبعاد هي إحدى تقنيات تعلُّم الآلة وتحليل البيانات المُستخدَمة لتقليص عدد الخصائص (الأبعاد) في مجموعة البيانات مع الاحتفاظ بأكبر قدر ممكن من المعلومات.

خوارزميات التجميع Clustering Algorithms

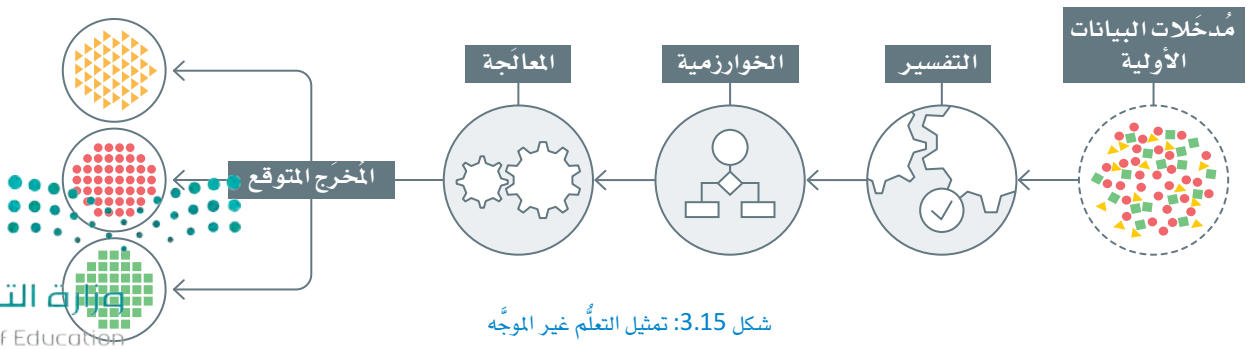
يمكن لخوارزميات التجميع تجميع العملاء المتشابهين استنادًا إلى السلوكيات أو الديموغرافيا، أو سجل المشتريات؛ لأغراض التسويق المُستهدَف وزيادة معدلات الاحتفاظ بالعملاء.

تقنيات تقليص الأبعاد
Dimensionality Reduction Techniques

تُستخدم تقنيات تقليص الأبعاد في ضغط الصورة لتقليل عدد وحدات البيكسل فيها مما يساعد على تقليص حجم البيانات اللازمة لتمثيلها مع الحفاظ على خصائصها الرئيسة.

النماذج التوليدية Generative Models

تُستخدم النماذج التوليدية في تطبيقات الكشف عن الاختلاف؛ حيث تُحدِّد الاختلافات في البيانات بتعلُّم الأنماط الطبيعية للبيانات باستخدام النموذج التوليدي.

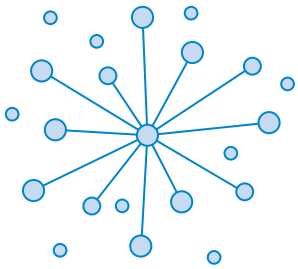


شكل 3.15: تمثيل التعلُّم غير الموجَّه

وإحدى المزايا الرئيسة لاستخدام التعلُّم غير الموجَّه هي أنه يمكن استخدامه للكشف عن الأنماط والعلاقات التي قد لا تبدو واضحة على الفور للمراقب البشري. وقد يكون هذا مفيدًا بشكل خاص في فهم مجموعات البيانات الكبيرة المكونة من النصوص غير المُتراكبة، حيث يكون التحليل اليدوي غير عملي. في هذه الوحدة، ستَستخدم مجموعة بيانات متوافرة للعامّة من المقالات الإخبارية من **هيئة الإذاعة البريطانية** (BBC) بواسطة **جرين وكوننجهام**، (Greene & Cunningham، 2006) لتوضيح بعض التقنيات الرئيسة للتعلُّم غير الموجَّه. يُستخدم المقطع البرمجي التالي لتحميل مجموعة البيانات، المُنظّمة في خمسة مجلدات إخبارية مختلفة تمثل مقالات من أقسام إخبارية مختلفة، هي: الأعمال التجارية، والسياسة، والرياضة، والتقنية، والترفيه. لن تستخدم القيم الخمسة في توجيه أي من الخوارزميات المُستخدَمة في هذه الوحدة. وبدلًا من ذلك، ستُستخدم فقط لأغراض التصوير والمصادقة. يتضمن كل مجلد إخباري مئات الملفات النصية، وكل ملف يتضمن محتوى مقالة واحدة محددة. وقد حُمِّلت مجموعة البيانات بالفعل إلى **مفكرة جوبيتر** (Jupyter Notebook) وستقوم لبنة التعليمات البرمجية بفتح واستخراج كل المُستنَدات والقيم المطلوبة في تركيبتين لبيانات القوائم، على التوالي.

العنقود (Cluster):

العنقود هو مجموعة من الأشياء المتشابهة. وفي تعلُّم الآلة، يشير التجميع (Clustering) إلى عملية تجميع البيانات غير المُعنوَنة في عناقيد متجانسة.



شكل 3.16: تمثيل عنقود

BBC open dataset

https://www.kaggle.com/datasets/shivamkushwaha/bbc-full-text-document-classification

D. Greene and P. Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering", Proc. ICML 2006. All rights, including copyright, in the content of the original articles are owned by the BBC.

```
# used to list all the files and subfolders in a given folder
from os import listdir
# used for generating random number
import random shuffling lists

bbc_docs=[] # holds the text of the articles
bbc_labels=[] # holds the news section for each article

for folder in listdir('bbc'): # for each news-section folder
    for file in listdir('bbc/'+folder): # for each text file in this folder

        # open the text file, use encoding='utf8' because articles may include non-ascii characters
        with open('bbc/'+folder+'/'+file,encoding='utf8',errors='ignore') as f:
            bbc_docs.append(f.read()) # read the text of the article and append to the docs list
        # use the name of the folder (news section) as a label for this doc
        bbc_labels.append(folder)
# shuffle the docs and labels lists in parallel
merged = list(zip(bbc_docs, bbc_labels)) # link the two lists
random.shuffle(merged) # shuffle them in parallel (with the same random order)
bbc_docs, bbc_labels = zip(*merged) # separate them again into individual lists.
```

تجميع المُستنَدات Document Clustering

الآن بعد تحميل مجموعة البيانات فإن الخطوة التالية هي تجربة عدة طرق غير موجَّهة، ومنها: التجميع الذي يُعدّ الطريقة غير الموجَّهة الأكثر شهرة في هذا النطاق. وبالنّظر إلى مجموعة من المُستنَدات غير المُعنوَنة، سيكون الهدف هو تجميع الوثائق المتشابهة معًا، وفي الوقت نفسه الفصل بين الوثائق غير المتشابهة.

تجميع المُستنَدات (Document Clustering):

تجميع المُستنَدات هو طريقة تُستخدم لتجميع المُستنَدات النصيّة في عناقيد بناءً على تشابه محتواها.

جدول 3.2: العوامل التي تُحدد جودة النتائج

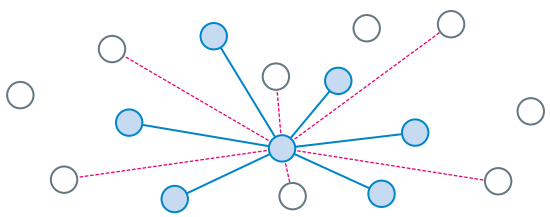
1	**طريقة تمثيل البيانات بالمتَّجهات.** على الرغم من أن تقنية **تكرار المصطلح- تكرار المُستنَد العكسي** (TF-IDF) أثبتت كفاءتها وفعاليتها في هذا المجال، إلا أنّك ستتعرف في هذه الوحدة على مزيد من البدائل الأكثر تطورًا وتعقيدًا.
2	**التعريف الدقيق للتشابه بين مستند وآخر.** بالنسبة للبيانات النصيّة المُمثلة بالمتَّجهات، تكون مقاييس المسافة الإقليدية وجيب التمام هما الأكثر شيوعًا. سيُستخدم الأول في الأمثلة المشروحة في هذه الوحدة.
3	**عدد العناقيد المُختارة.** يوفر **التجميع التكتلي** (Agglomerative Clustering - AC) طريقة واضحة لتحديد العدد المناسب من العناقيد ضمن مجموعة محددة من البيانات، وهو التحدي الرئيس الذي يواجه مهام التجميع.

تحديد عدد العناقيد
Selecting the Number of Clusters

تحديد العدد الصحيح للعناقيد هو خطوة ضرورية ضمن مهام التجميع. للأسف، تعتمد الغالبية العظمى من خوارزميات التجميع على المُستخدِم في تحديد عدد العناقيد الصحيحة ضمن المُدخَلات. ربما يكون للعدد المُحدد تأثيرًا كبيرًا على جودة النتائج وقابليتها للتفسير، ولكن هناك العديد من المقاييس أو المؤشرات التي يمكن استخدامها لتحديد عدد العناقيد.

- إحدى الطرائق الشائعة هي استخدام مقياس **التراص** (Compactness). يمكن القيام بذلك عن طريق حساب مجموع المسافات بين النقاط ضمن كل عنقود، وتحديد عدد العناقيد الذي يقلل من هذا المجموع إلى الحد الأدنى.
- هناك طريقة أخرى تتلخص في مقياس **الفصل** (Separation) بين العناقيد، مثل متوسط المسافة بين النقاط في العناقيد المختلفة،وبناء عليه، يتم تحديد عدد العناقيد الذي يرفع من هذا المتوسط.

وبشكل عملي، غالبًا ما تتعارض المنهجيات المذكورة بالأعلى مع بعضها من حيث التوصية بأرقام مختلفة، ويمثّل هذا تحدّيًا مشتركًا عند التعامل مع البيانات النصية بشكلٍ خاص، فعادةً ما يصعُب تمييز تركيبها.



شكل 3.17: آلة حساب المسافات بين النقاط

المسافة الإقليدية (Euclidean Distance):

المسافة الإقليدية هي مسافة الخط المستقيم بين نقطتين في فضاء متعدد الأبعاد. وتُحسب بالجذر التربيعي لمجموع مربعات الفروقات بين الأبعاد المناظرة للنقاط. تُستخدم المسافة الإقليدية في التجميع لقياس التشابه بين نقطتي بيانات.

مسافة جيب التمام (Cosine Distance):

تستخدم مسافة جيب التمام لقياس التشابه في جيب التمام بين نقطتي البيانات. فهي تحسب جيب تمام الزاوية بين متَّجهين يمثلان نقاط البيانات، وتُستخدَم عادةً في تجميع البيانات النصيّة. وتقع قيمة جيب التمام بين -1 و 1؛ حيث تشير القيمة -1 إلى الاتجاه العكسي، بينما تشير القيمة 1 إلى الاتجاه نفسه.

في التعلُّم غير الموجَّه، يشير عدد العناقيد إلى عدد المجموعات أو التصنيفات التي تنقسم إليها البيانات بواسطة الخوارزمية. ويُعدُّ تحديد عدد العناقيد الصحيح أمرًا مهمًا لأنه يؤثر على دقة النتائج وقابليتها للتفسير. إذا كان عدد العناقيد كبيرًا للغاية، فإنّ المجموعات ستكون محدّدةً جدًا وبدون معنى. في حين أنه إذا كان عدد العناقيد منخفضًا للغاية، فإنّ المجموعات ستكون ممتدة على نطاق واسع جدًا، ولن تستنبط التركيب الأساسي للبيانات. من الضروري تحقيق التوازن بين توفير عددٍ كافٍ من العناقيد لاستنباط أنماط ذات معنى، وألا تكون كثيرة في الوقت نفسه بالقدر الذي يجعل النتائج مُعقدة للغاية وغير مفهومة.

التجميع الهرمي (Hierarchical Clustering):

التجميع الهرمي هو خوارزمية التجميع المُستخدَمة لتجميع البيانات في عناقيد بناءً على التشابه. في التجميع الهرمي، تُنظّم نقاط البيانات في تركيب يشبه الشجرة، حيث تكون كل عُقدة بمثابة عنقود، وتكون العُقدة الأم هي نقطة التقاء العُقد المتفرعة منها.

يُستخدَم المقطع البرمجي التالي لاستيراد مكتبات محددة تُستخدَم في التجميع الهرمي من بدايته حتى نهايته:

```
# used for tfi-df vectorization, as seen in the previous unit
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import AgglomerativeClustering # used for agglomerative clustering

# used to visualize and support hierarchical clustering tasks
import scipy.cluster.hierarchy as hierarchy

# set the color palette to be used by the 'hierarchy' tool.
hierarchy.set_link_color_palette
(['blue','green','red','yellow','brown','purple','orange','pink','black'])

import matplotlib.pyplot as plt # used for general visualizations
```

البرمجة الاتجاهية للنصوص Text Vectorization

تتطلب العديد من طرق التعلُّم غير الموجَّه تمثيل النصّ الأوليّ بالمتَّجَهات في تنسيق رقميّ، كما تمّ عرضه في الوحدة السابقة، ويستخدم المقطع البرمجي التالي أداة TfidfVectorizer التي أُستخدمت في الدرس السابق لهذا الغرض:

```
vectorizer = TfidfVectorizer(min_df=10) # apply tf-idf vectorization, ignore words that
appear in more than 10 docs.

text_tfidf=vectorizer.fit_transform(bbc_docs) # fit and transform in one line

text_tfidf
```

```
<2225x5867 sparse matrix of type '<class 'numpy.float64'>'
        with 392379 stored elements in Compressed Sparse Row format>
```

الآن تحوَّلت بيانات النص إلى تنسيق رقمي متباعد كما أُستخدمت في الدرس السابق.

يَستخدم المقطع البرمجي التالي أداة TSENVisualizer من مكتبة yellowbrick لإسقاط وتصوير النصوص المُمثلة بالمتَّجهات في فضاء ثنائي الأبعاد:

```
%%capture
!pip install yellowbrick
from yellowbrick.text import TSNEVisualizer
```

تقليص الأبعاد Dimensionality Reduction

يكون تقليص الأبعاد مفيدًا في العديد من التطبيقات مثل:

- **تصوير البيانات عالية الأبعاد:** من الصعب تصوير البيانات في فضاء عالي الأبعاد، ولذلك تُقلّص الأبعاد ليسهل تصوير البيانات وفهمها في هذه الحالة.
- **تبسيط النموذج:** النموذج ذو الأبعاد الأقل يكون أبسط وأسهل فهمًا، ويستغرق وقتًا أقل في عملية التدريب.
- **تحسين أداء النموذج:** يُساعد تقليص الأبعاد في التخلص من التشويش وتكرار البيانات، مما يُحسّن أداء النموذج.

تضمين المجاور العشوائي الموزع على شكل T t-Distributed Stochastic Neighbor Embedding (T-SNE)

خوارزمية تضمين المجاور العشوائي الموزَّع على شكل T (T-SNE) هي خوارزمية تعلُّم الآلة غير الموجَّه المُستخدَمة لتقليص الأبعاد.

جدول 3.3: تقنيات تقليص الأبعاد

التقنية	الوصف	مثال التطبيق العملي
تحديد الخصائص (Feature selection)	تحديد الخصائص يتضمن تحديد مجموعة فرعية من الخصائص الرئيسة.	تحتوي مجموعات البيانات الطبية على مئات من أعمدة البيانات ذات الصلة بحالة المريض. يمكن لعدد قليل من هذه الخصائص مساعدة النموذج في التشخيص السليم لحالة المريض. بينما تكون السمات الأخرى غير مرتبطة بالتشخيص وقد تُشتت النموذج، وتحديد الخصائص يتجاهل كل الخصائص بإستثناء الأكثر تميزًا منها.
تحويل الخصائص (Feature transformation)	يتضمن تحويل الخصائص تجميع الخصائص الأصلية أو تحويلها لإنشاء مجموعة جديدة من الخصائص، واستبدال الخصائص الرئيسة إذا لم تكن هناك حاجة إليها.	إذا توقَّع النموذج إقامة المريض في المستشفى، يمكن إنشاء خصائص إضافية للنموذج باستخدام الخصائص الحالية لسجلات الحالة الطبية للمريض. على سبيل المثال، حساب عدد الفحوصات المخبرية المطلوبة على مدار الأسبوع الماضي، أو عدد الزيارات على مدار الشهر الماضي. وهناك مثال آخر، وهو: حساب مساحة المستطيل بإستخدام ارتفاعه وعرضه.
التعلُّم المتشعِّب (Manifold learning)	تقنيات التعلُّم المتشعِّب، مثل **تضمين المجاور العشوائي الموزَّع على شكل T (T-SNE) والتقريب والإسقاط المتشعِّب المنتظم** (Uniform Manifold Approximation and Projection –UMAP) هي تقنيات التعلُّم غير الموجَّه التي تهدف إلى الحفاظ على تركيب البيانات في الفضاء منخفض الأبعاد.	يمكن لهذه التقنيات تحويل صورة عالية الأبعاد إلى فضاء منخفض الأبعاد مع الحفاظ على الخصائص والتركيب الأساسيين لها. ونظرًا لأن هذا يقلص من المساحة المطلوبة، فإنه يمكن تخزين وإرسال هذا التمثيل وإعادة بناء الصورة الأصلية مع خسارة أقل قدر من المعلومات.

إحدى الخصائص الرئيسة لتقنية **تضمين المجاور العشوائي الموزَّع على شكل T** (T-SNE) هي محاولة الحفاظ على التركيب المحلي للبيانات قدر الإمكان، حتى تتقارب نقاط البيانات الشبيهة في التمثيل منخفض الأبعاد، ويتحقق ذلك بتقليص التباعد بين التوزيعين المحتملين: توزيع البيانات عالية الأبعاد، وتوزيع البيانات منخفضة الأبعاد.

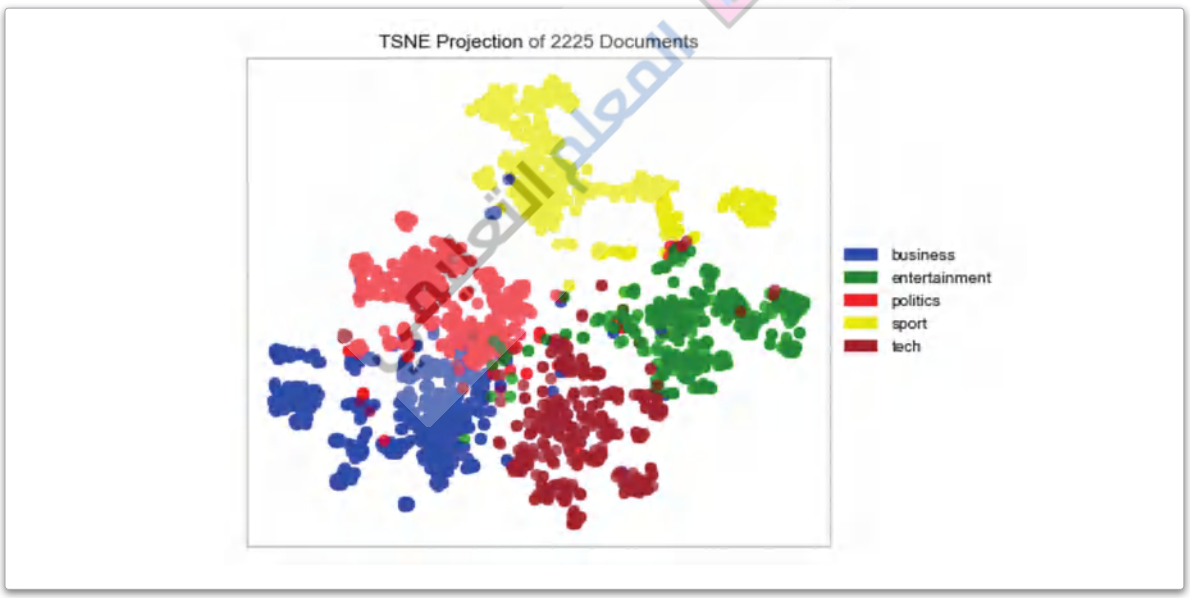
مجموعة بيانات هيئة الإذاعة البريطانية المُمثلة بالمتَّجهات تُصنَّف بالتأكيد كبيانات عالية الأبعاد، لأنها تتضمن بُعدًا مستقلًا أي **عمودًا** (Column) لكل كلمة فريدة تظهر في البيانات. يُحسب العدد الإجمالي للأبعاد كما يلي:

```
print('Number of unique words in the BBC documents vectors:',
                    len(vectorizer.get_feature_names_out()))
```

```
Number of unique words in the BBC documents vectors: 5867
```

يُستخدَم المقطع البرمجي التالي لإسقاط 5,867 بُعدًا في محورين فقط وهما محوري X و Y في الرسم البياني. يُستخدَم المقطع البرمجي التالي لتصميم مُخطَّط الانتشار حيث يمثل كل لون أحد الأقسام الإخبارية الخمسة.

```
tsne = TSNEVisualizer(colors=['blue','green','red','yellow','brown'])
tsne.fit(text_tfidf,bbc_labels)
tsne.show();
```



شكل 3.18: إسقاط تضمين المجاور العشوائي الموزَّع على شكل T (T-SNE)

يَستخدِم هذا التصور قيمة ground-truth (بيانات الحقيقة المعتمدة) من **القسم الإخباري** (News Section) في كل مستند للكشف عن انتشار كل قيمة في إسقاط فضاء البرمجة الاتجاهية ثنائي الأبعاد. يوضِح الشكل أنه على الرغم من ظهور بعض الشوائب في فراغات مُحدَّدة من فضاء البيانات، إلا أن الأقسام الإخبارية الخمسة منفصلة بشكل جيد. وسنستعرض لاحقًا البرمجة الاتجاهية المُحسَّنة للحد من هذه الشوائب.

التجميع التكتلي Agglomerative Clustering (AC)

التجميع التكتلي (AC) هو الطريقة الأكثر انتشارًا وفعاليةً في هذا الفضاء، فمن خلالها يمكن التغلّب على هذا التحدي بتوفير طريقة واضحة لتحديد العدد المناسب من العناقيد. يستند **التجميع التكتلي** (AC) إلى منهجية التصميم من أسفل إلى أعلى، حيث تبدأ بحساب المسافة بين كل أزواج نقاط البيانات، ثم اختيار النقطتين الأقرب ودمجهما في عنقود واحد. تتكرر هذه العملية حتى تُدمج كل نقاط البيانات في عنقود واحد، أو حتى الوصول إلى العدد المطلوب من العناقيد.

شكل 3.19: التجميع التكتلي (AC)

دالة Linkage( ) *fx*

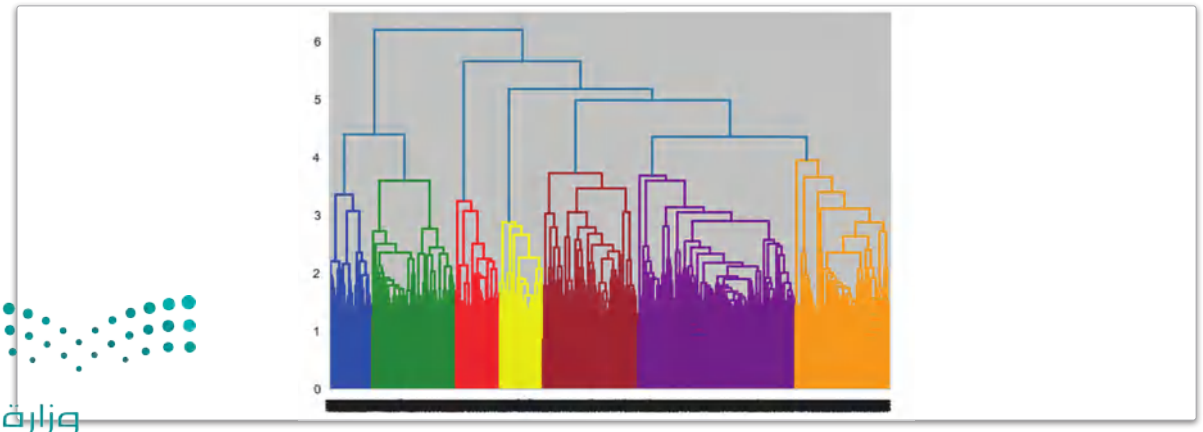
تُنفذ لغة البايثون **التجميع التكتلي** (AC) باستخدام دالة linkage( ).

يجب توفير متغيرين لدالة linkage( ):

- البيانات النصيّة المُمثلة بالمتَّجهات، ويمكن استخدام دالة toarray( ) لتحويل البيانات إلى تنسيق كثيف يمكن لهذه الدالة أن تتعامل معه.
- مقياس المسافة الذي يجب استخدامه لتحديد العناقيد التي ستُدمج أثناء عملية التجميع التكتلي. تتوفر عدة خيارات من مقاييس المسافة للاختيار من بينها وفقًا لمتطلبات وتفضيلات المُستخدِم، مثل المسافة **الإقليدية** (Euclidian)، ومسافة **مانهاتن** (Manhattan)... إلخ، ولكن في هذا المشروع ستستخدم طريقة **وارد** (**ward**) القياسية.

يستخدم المقطع البرمجي التالي دالة linkage( ) من الأداة **الهرمية** (Hierarchy) الواردة بالأعلى لتطبيق هذه العملية على بيانات هيئة الإذاعة البريطانية المُمثَّلة بالمتَّجهات:

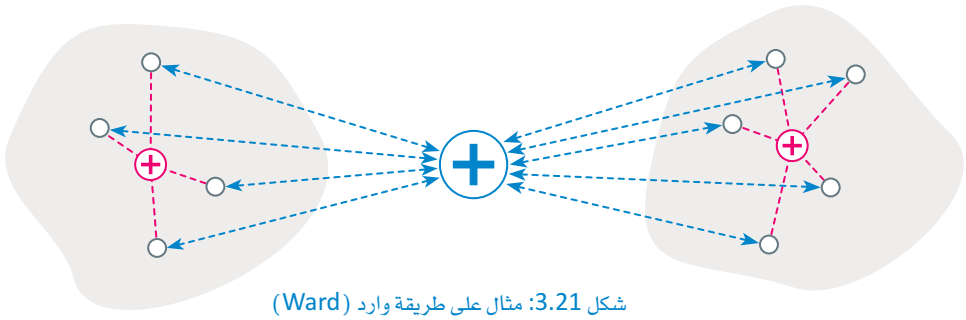
```
plt.figure() # create a new empty figure

# iteratively merge points and clusters until all points belong to a single cluster
# return the linkage of the produced tree
linkage_tfidf=hierarchy.linkage(text_tfidf.toarray(),method='ward')

# visualize the linkage
hierarchy.dendrogram(linkage_tfidf)

# show the figure
plt.show()
```



شكل 3.20: الرسم الشجري الهرمي لبيانات هيئة الإذاعة البريطانية

مسافة وارد Ward Distance

يَستخدِم المثال أعلاه طريقة وارد (Ward) القياسية لقياس المسافة للمتغير الثاني. تستند مسافة وارد (Ward) إلى مفهوم التباين داخل العنقود، وهو مجموع المسافات بين النقاط في العنقود. في كل تكرار، تُقيِّم الطريقة كل عملية دمج ممكنة بحساب التباين داخل العنقود قبل عمليّة الدمج، وبعدها، ثم تبدأ عملية الدمج التي تحقِّق أقل ارتفاع في التباين. أظهرت مسافة وارد (Ward) نتائج جيدة في معالجة البيانات النصيّة، بالرغم من وجود العديد من الخيارات الأخرى.



شكل 3.21: مثال على طريقة وارد (Ward)

الرسم الشجري (Dendrogram):
الرسم الشجري هو رسم تخطيطي تفرعي يوضح العلاقة الهرمية بين البيانات، ويأتي عادة في صورة أحد مُخرَجات التجميع الهرمي.

الرسم الشجري في الشكل 3.20 يعرض طريقة واضحة لتحديد عدد العناقيد. في هذا المثال، تقترح المكتبة استخدام 7 عناقيد، مع تمييز كل عنقود بلون مختلف. قد يتبنى المُستخدِم هذا المُقترح أو يَستخدِم الرسم الشجري لاختيار رقم مختلف. على سبيل المثال، دُمِّج اللونين الأزرق والأخضر في آخر خطوة مع مجموعة العناقيد لكل الألوان الأخرى. وهكذا، سيؤدي اختيار 6 عناقيد إلى دمج اللونين الأرجواني والبرتقالي، بينما اختيار 5 عناقيد سيؤدي إلى دمج اللونين الأزرق والأخضر.

يتبنى المقطع البرمجي التالي مقترحات الأداة ويَستخدِم أداة التجميع التكتلي من مكتبة **سكليرن** (Sklearn) لتقسيم المُخطَّط الشجري بعد إنشاء العناقيد السبع:

```
AC_tfidf=AgglomerativeClustering(linkage='ward',n_clusters=7) # prepare the tool, set the number of clusters.

AC_tfidf.fit(text_tfidf.toarray()) # apply the tool to the vectorized BBC data.

pred_tfidf=AC_tfidf.labels_ # get the cluster labels.

pred_tfidf
```

```
array([6, 2, 4, ..., 6, 3, 5], dtype=int64)
```

لاحظ أن قيمة ground-truth (بيانات الحقيقة المعتمدة) من **القسم الإخباري** (News Section) في كل مستند لم تُستخدَم على الإطلاق في هذه العملية. وبدلًا من ذلك، عولجت عملية التجميع استنادًا إلى نص محتوى كل وثيقة على حده. إنَّ قيم بيانات الحقيقة المعتمدة مفيدة في التطبيق العملي، فهي تتيح التحقق من صحة نتائج التجميع. وقيم بيانات الحقيقة المعتمدة الحالية موجودة في قائمة bbc_labels (قيم_ هيئة الإذاعة البريطانية).

يَستخدِم المقطع البرمجي التالي قيم بيانات الحقيقة المعتمدة وثلاثة دوال مختلفة لتسجيل النقاط من مكتبة **سكليرن** (Sklearn) لتقييم جودة تجميع البيانات:

- تكون قيم **مؤشر التجانس** (Homogeneity Score) بين 0 و 1 ويمكن زيادة هذه القيم عندما تكون كل النقاط في كل عنقود لها قيمة بيانات الحقيقة المعتمدة. وبالمثل، يحتوي كل عنقود على نقاط البيانات وحيدة التصنيف.
- تكون قيمة **مؤشر راند المُعدل** (Adjusted Rand Score) بين 0.5- و 1.0 ويمكن زيادة هذه القيم عندما تقع كل نقاط البيانات ذات القيم نفسها في العنقود نفسه وكل نقاط البيانات ذات القيم المختلفة في عناقيد مختلفة.
- تكون قيمة **مؤشر الاكتمال** (Completeness Score) بين 0 و 1 ويمكن زيادة هذه القيمة بتعيين كل نقاط البيانات من تصنيف مُحدد في العنقود نفسه.

```
from sklearn.metrics import homogeneity_score,adjusted_rand_
score,completeness_score

print('\nHomogeneity score:',homogeneity_score(bbc_labels,pred_tfidf))
print('\nAdjusted Rand score:',adjusted_rand_score(bbc_labels,pred_tfidf))
print('\nCompleteness score:',completeness_score(bbc_labels,pred_tfidf))
```

```
Homogeneity score: 0.6224333236569846

Adjusted Rand score: 0.4630492696176891

Completeness score: 0.5430590192420555
```

المؤشر أقرب إلى 1 وهذا يعني أن مجموعة النصوص في العنقود تنتمي إلى قيمة واحدة.

المؤشر أقرب إلى 1 وهذا يعني إنشاء روابط أفضل بين العناقيد والقيم؛ كلٌّ على حده.

لاستكمال تحليل البيانات، يُعاد تجميع البيانات باستخدام 5 عناقيد، بالتساوي مع العدد الحقيقي لقيم ground-truth (بيانات الحقيقة المعتمدة):

```
AC_tfidf=AgglomerativeClustering(linkage='ward',n_clusters=5)
AC_tfidf.fit(text_tfidf.toarray())
pred_tfidf=AC_tfidf.labels_

print('\nHomogeneity score:',homogeneity_score(bbc_labels,pred_tfidf))
print('\nAdjusted Rand score:',adjusted_rand_score(bbc_labels,pred_tfidf))
print('\nCompleteness score:',completeness_score(bbc_labels,pred_tfidf))
```

```
Homogeneity score: 0.528836079209762

Adjusted Rand score: 0.45628412883628383

Completeness score: 0.6075627851312266
```

نظرًا لقدرة التجميع الهرمي على إيجاد العدد الحقيقي من القيم، وتوفير مؤشر اكتمال أكثر دقة، ستحصل على عملية تجميع أفضل من حيث تمثيل البيانات.

على الرغم من أن نتائج المؤشر تُظهر أن التجميع التكتلي باستخدام البرمجة الاتجاهية **لتكرار المُستنَد العكسي** (TF-IDF) تحقق نتائج معقولة، إلا أنّه لا يزال بالإمكان تحسين دقة عملية التجميع. سيوضح القسم التالي كيف يمكن أن نحقق نتائج مبهرة باستخدام تقنيات البرمجة الاتجاهية المُستنِدة على الشبكات العصبية.

اصطلاح-تكرار

البرمجة الاتجاهية للكلمات باستخدام الشبكات العصبية
Word Vectorization with Neural Networks

البرمجة الاتجاهية **لتكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF) تستند إلى حساب تكرار الكلمات ومعالجتها عبر المُستنَدات في مجموعة البيانات. بالرغم من أن هذا يحقق نتائج جيدة، إلا أنّ القيود الكبيرة تعيب الطرائق المستنِدة إلى التكرار. فهي تتجاهل تمامًا العلاقة الدلالية بين الكلمات. على سبيل المثال، على الرغم من أن كلمتي trip (نزهة) و journey (رحلة) مترادفتان، إلا أنّ البرمجة الاتّجاهيّة المُستندة إلى التّكرار ستتعامل معهما باعتبارهما كلمتان منفصلتان تمامًا ولهما خصائص مستقلة. وبالمثل، بالرغم من أن كلمتَي apple (تفاحة) و fruit (فاكهة) مترابطتان دلاليًا؛ لأن التفاح نوع من الفاكهة إلا أنّ ذلك لن يؤخذ بعين الاعتبار أيضًا.

تؤثر هذه القيود كثيرًا على التطبيقات التي تستخدم هذا النوع من البرمجة الاتجاهية. فكِّر في الجملتين التاليتين:

- I have a very high fever، so I have to visit a doctor (لديّ حمّى شديدة، ويجب عليّ زيارة الطبيب).
- My body temperature has risen significantly، so I need to see a healthcare professional (ارتفعت درجة حرارة جسمي كثيرًا، ويجب عليّ زيارة أخصائي الرعاية الصحية).

بالرغم من أن الجملتين تصفان الحالة نفسها إلا أنهما لا تتشاركان أي كلمات دلالية. ولذلك، ستفشل خوارزميات التجميع المُستنِدة إلى **تكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF) أو أي برمجة اتجاهية (تستند إلى التكرار) في رؤية التشابه بين الكلمات، ومن المحتمل ألا تضعها في نفس العنقود.

نموذج الكلمة إلى المتَّجَه Word2Vec

يمكن معالجة هذه القيود بالطرائق التي تأخذ بعين الاعتبار التشابه الدلالي بين الكلمات. إحدى الطرق الشهيرة المُتبعة في هذا الصدد هي **نموذج الكلمة إلى المتَّجَه** (Word2Vec) التي تَستخدِم بُنية تَستند إلى الشبكات العصبية. يَستند **نموذج الكلمة إلى المتَّجَه** (Word2Vec) إلى فكرة أن الكلمات المتشابهة دلاليًا تُحاط بكلمات مماثلة في السياق نفسه. ولذلك، نجد الشبكات العصبية تستخدم التضمين الخفي لكل كلمة للتنبؤ بالسياق، مع ضرورة إنشاء الروابط بين الكلمات والتضمينات الشبيهة. عمليًا، يخضع **نموذج الكلمة إلى المتَّجَه** (Word2Vec) للتدريب المُسبَق على ملايين المُستنَدات لتعلُّم التضمين عالي الدقة للكلمات. يمكن تحميل النماذج المُدرَّبة مسبَقًا واستخدامها في التطبيقات المُستندة إلى النصوص. يَستخدِم المقطع البرمجي التالي مكتبة **جينسم** (Gensim) لتحميل نموذج شهير مُدرَّب مسبَقًا على مجموعة كبيرة جدًا من **أخبار قوقل** (Google News):

```
import gensim.downloader as api
model_wv = api.load('word2vec-google-news-300')
fox_emb=model_wv['fox']
print(len(fox_emb))
```

```
300
```

هذا النموذج يربط كل كلمة بتضمين مكوّن من 300 بُعد.

الكلمات المُستبعَدة (Stopwords):

الكلمات المُستبعَدة هي كلمات شائعة في اللغات تُستبعد عادةً أثناء المعالجة المُسبَقة للنصوص ضمن مهام **معالجة اللغات الطبيعية** (NLP) مثل البرمجة الاتجاهية للكلمات. هذه الكلمات تشمل أدوات التعريف، وحروف العطف، وحروف الجر، والكلمات التي لا تكون مفيدة لتحديد معنى النصّ، أو سياقه.

التضمين (Embedding):

التضمين يُعبِّر عن الكلمات أو الرموز في فضاء المتَّجَه المستمر حيث ترتبط الكلمات المتشابهة دلاليًا مع النقاط القريبة.

الأبعاد العشرة الأولى للتضمين العددي لكلمة fox (ثعلب) موضحة بالأسفل:

```
fox_emb[:10]
```

```
array([-0.08203125, -0.01379395, -0.3125    , -0.04125977,  0.05493164,
       -0.12988281, -0.10107422, -0.00164795,  0.15917969,  0.12402344],
      dtype=float32)
```

يَستخدِم النموذج تضمينات الكلمات لتقييم درجة التشابه. فكِّر في المثال التالي حيث تُظهر المقارنة بين كلمة car (السيارة) والكلمات الأخرى درجة التشابة من خلال تناقص قيم التشابة. علمًا بأن قيم التشابه تقع دومًا بين 0 و 1.

```
pairs = [
    ('car', 'minivan'),
    ('car', 'bicycle'),
    ('car', 'airplane'),
    ('car', 'street'),
    ('car', 'apple'),
]
for w1, w2 in pairs:
    print(w1, w2, model_wv.similarity(w1, w2))
```

```
car minivan 0.69070363
car bicycle 0.5364484
car airplane 0.42435578
car street 0.33141237
car apple 0.12830706
```

يُمكن استخدام المقطع البرمجي التالي للعثور على الكلمات الخمسة المشابهة لإحدى الكلمات:

```
print(model_wv.most_similar(positive=['apple'], topn=5))
```

```
[('apples', 0.720359742641449), ('pear', 0.6450697183609009),
('fruit', 0.6410146355628967), ('berry', 0.6302295327186584), ('pears',
0.613396167755127)]
```

يُمكن استخدام التصوير في التحقق من صحة تضمينات هذا النموذج المُدرَّب مُسبقًا، ويُمكن تحقيق ذلك عبر:

- تحديد نماذج الكلمات من مجموعة بيانات هيئة الإذاعة البريطانية.
- استخدام **تضمين المجاور العشوائي الموزَّع على شكل T** (T-SNE) لتخفيض التضمين ذي الـ 300 بعدٍ لكل كلمة إلى نقطة ثنائية الأبعاد.
- تصوير النقاط في مُخطَّط الانتشار في الفضاء ثنائي الأبعاد.

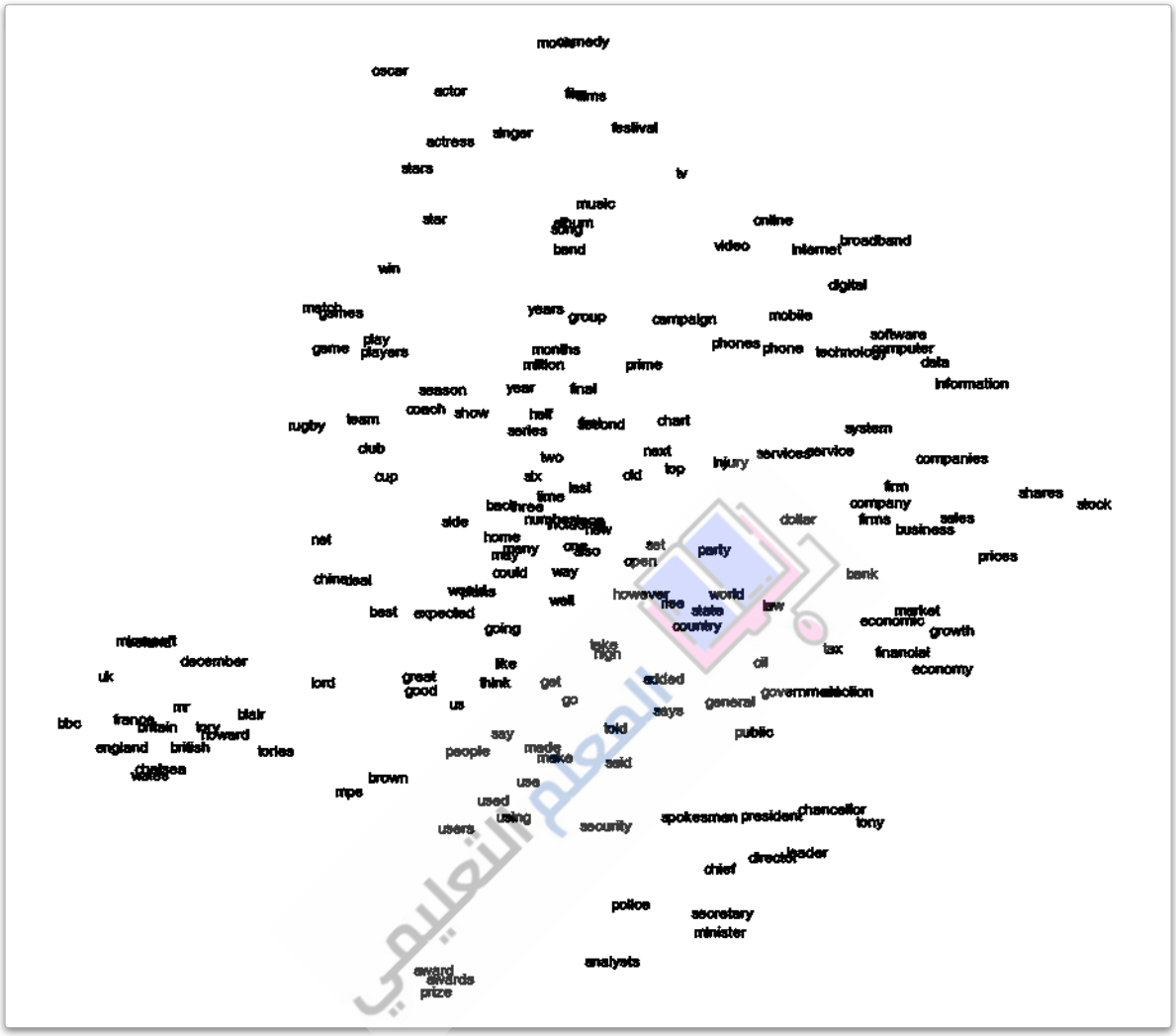
```
%%capture
import nltk # import the nltk library for nlp.
import re # import the re library for regular expressions.
import numpy as np # used for numeric computations
from collections import Counter # used to count the frequency of elements in a given list
from sklearn.manifold import TSNE # Tool used for Dimensionality Reduction.

# download the 'stopwords' tool from the nltk library. It includes very common words for different languages
nltk.download('stopwords')

from nltk.corpus import stopwords # import the 'stopwords' tool.

stop=set(stopwords.words('english')) # load the set of english stopwords.
```

تُستخدَم الدالة الآتية لاحقًا لتحديد عينة من الكلمات التمثيلية من مجموعة بيانات هيئة الإذاعة البريطانية. يُحدِّد المقطع البرمجي الكلمات الخمسين الأكثر تكرارًا على وجه التحديد من الأقسام الإخبارية الخمسة لهيئة الإذاعة البريطانية مع استثناء **الكلمات المُستبعَدة** (Stopwords) وهي الكلمات الإنجليزية الشائعة جدًا والكلمات التي لم تُضمَّن في **نموذج الكلمة إلى المتَّجِه** (Word2Vec) المُدرَّب مسبقًا.

بعض الكلمات الإنجليزية الشائعة التي تعتبر كلمات مُستبعَدة (Stopwords) هي a (أ) و the (ال) و is (يكون) و are (يكونون).

```
def get_sample(bbc_docs:list,
               bbc_labels:list
               ):

    word_sample=set() # a sample of words from the BBC dataset

    # for each BBC news section
    for label in ['business', 'entertainment', 'politics', 'sport', 'tech']:

        # get all the words in this news section, ignore stopwords.
        # for each BBC doc and for each word in the BBC doc
        # if the word belongs to the label and is not a stopword and is included in the Word2Vec model
        label_words=[word for i in range(len(bbc_docs))
                     for word in re.findall(r'\b\w\w+\b',bbc_docs[i].lower())
                         if bbc_labels[i]==label and
                            word not in stop and
                            word in model_wv]

        cnt=Counter(label_words) # count the frequency of each word in this news section.

        # get the top 50 most frequent words in this section.
        top50=[word for word,freq in cnt.most_common(50)]
        # add the top50 words to the word sample.
        word_sample.update(top50)


    word_sample=list(word_sample) # convert the set to a list.
    return word_sample

word_sample=get_sample(bbc_docs,bbc_labels)
```

وأخيرًا، ستَستخدِم طريقة **تضمين المجاور العشوائي الموزَّع على شكل T** (T-SNE) لتخفيض التضمينات ذات الـ 300 بعدٍ للكلمات في العينة ضمن النقاط ثنائية الأبعاد. بعدها، تُمثَّل النقاط في مُخطَّط انتشار بسيط.



شكل 3.22: تمثيل الكلمات الأكثر تكرارًا من مجموعة بيانات هيئة الإذاعة البريطانية

يُثبت المُخطَّط أن تضمينات **نموذج الكلمة إلى المتَّجه** (Word2Vec) تستنبط الارتباطات الدلالية بين الكلمات، كما يتضح من مجموعات الكلمات الواضحة مثل:

- economy (الاقتصاد)، economic (الاقتصادية)، business (الأعمال)، financial (المالية)، sales (المبيعات)، bank (المصرف)، firm (الشركة)، firms (الشركات).
- Internet (الإنترنت)، mobile (الهاتف المحمول)، phones (الهواتف)، phone (الهاتف)، broadband (النطاق العريض)، online (متصل)، digital (رقمي).
- actor (ممثل)، actress (ممثلة)، film (فيلم)، comedy (كوميدي)، films (أفلام)، festival (مهرجان)، band (فرقة)، movie (فيلم).
- game (لعبة)، team (فريق)، match (مباراة)، players (لاعبون)، coach (مدرِّب)، injury (إصابة)، club (نادي)، rugby (الرجبي).

البرمجة الاتجاهية للجُمل باستخدام التعلُّم العميق
Sentence Vectorization with Deep Learning

على الرغم من إمكانية استخدام **نموذج الكلمة إلى المتَّجَه** (Word2Vec) في نمذجة الكلمات الفردية، يتطلب التجميع البرمجة الاتجاهية للنص بأكمله. إحدى الطرائق الأكثر شهرة لتحقيق ذلك هي تمثيلات **ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) المُستنِدة إلى منهجية التعلُّم العميق.

تمثيلات الترميز ثنائية الاتجاه من المحولات
Bidirectional Encoder Representations from Transformers (BERT)

تمثيلات الترميز ثنائية الاتجاه من المحولات (BERT) هي نموذج تمثيل لغوي قوي طورته شركة قوقل، ويعدُّ التدريب المُسبَق والضبط الدقيق عاملان رئيسان وراء قدرة **تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT) على تطبيق نقل التعلُّم، أي القدرة على الاحتفاظ بالمعلومات حول مشكلة ما والاستفادة منها في حلِّ مشكلة أخرى، ويتمّ التدريب المُسبَق عبر تغذية النموذج بكمية هائلة من البيانات غير المُعنوَنة لعدة مهام، مثل التنبؤ اللغوي المُقنَّع (إخفاء الكلمات العشوائية في مُدخَلات النصوص والمُهِمَّة هي التنبؤ بهذه الكلمات). يُهيِّئ نموذج **تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT) المتغيرات المُدرَّبة مُسبقًا للضبط الدقيق، كما تُستخدَم مجموعات البيانات المُعنوَنة من المهام النهائية لضبط دقة عمل النموذج، ويكون لكل مُهِمَّة نهائية نماذج دقيقة منفصلة، برغم أنها مُهيَّئة بالمتغيرات المُدرَّبة نفسها مسبقًا. على سبيل المثال، تختلف عملية الضبط الدقيق لنموذج تحليل المشاعر عن نموذج الإجابة على الأسئلة. ومن المهم معرفة أن الفروقات في بنية النماذج تصبح ضئيلة أو منعدمة بعد خطوة ضبط الدقة.

تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات SBERT

تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT) هي الإصدار المُعدَّل من **تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT). تُدرَّب **تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT) مثل **نموذج الكلمة إلى المتَّجَه** (Word2Vec) للتنبؤ بالكلمات بناءً على سياق الجُمل الواردة بها. ومن ناحية أخرى، تُدرَّب **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) للتنبؤ بما إذا كانت جملتان متشابهتين دلاليًا. تُستخدَم **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) لإنشاء تضمينات لأجزاء النصوص الأطول من الجُمل، مثل الفقرات، أو النصوص القصيرة، أو المقالات في مجموعة بيانات هيئة الإذاعة البريطانية محل الدراسة في هذه الوحدة. بالرغم من أن النماذج الثلاث تستند جميعها إلى الشبكات العصبية، إلا أن **تمثيلات الترميز ثنائية الاتجاه من المحولات** (BERT) و**تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) تتبعان بُنية مختلفة بشكل كبير وأكثر تعقيدًا من **نموذج الكلمة إلى المتَّجَه** (Word2Vec).

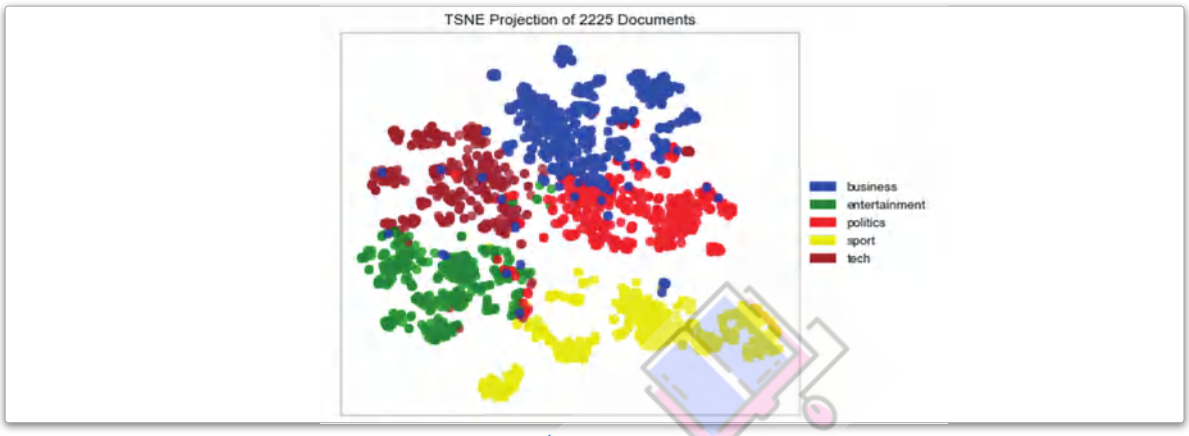
مكتبة الجُمل والمحولات Sentence_transformers Library

تُطبق مكتبة **الجُمل والمحولات** (sentence_transformers) الوظائف الكاملة لنموذج تمثيلات **ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT). تأتي المكتبة بالعديد من نماذج **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) المُدرَّبة مُسبقًا؛ كلٌ منها مُدرَّب على مجموعة بيانات مختلفة ولتحقيق أهداف مختلفة. يعمل المقطع البرمجي التالي على تحميل أحد النماذج العامة الشهيرة المُدرَّبة مُسبقًا، ويستخدمها لإنشاء تضمينات للمستندات في مجموعة بيانات هيئة الإذاعة البريطانية:

```
%%capture
!pip install sentence_transformers
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('all-MiniLM-L6-v2') # load the pre-trained model.

text_emb = model.encode(bbc_docs) # embed the BBC documents.
```

لقد استخدمت في وقت سابق في هذه الوحدة أداة تضمين المجاور العشوائي الموزع على شكل T والتي هي (TSNEVisualizer)، لتصوير المُستندات المُمثلة بالمتُّجَهات المُنتجة باستخدام أداة تكرار المصطلح-تكرار المستند العكسي (TF-IDF). يمكن الآن استخدامها للتضمينات المُنتَجة بواسطة **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT):

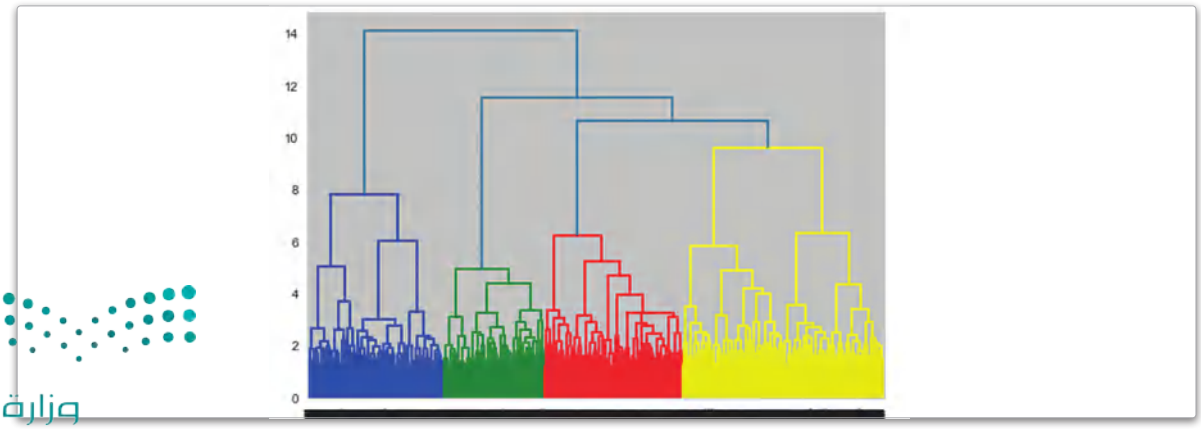
```
tsne = TSNEVisualizer(colors=['blue','green','red','yellow','brown'])
tsne.fit(text_emb,bbc_labels)
tsne.show();
```



شكل 3.23: إسقاط تضمين المجاور العشوائي الموزَّع على شكل T (T-SNE) للتضمينات المُنتجة بواسطة تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT)

يوضح الشكل أن **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) تؤدي إلى فصل أكثر وضوحًا للأقسام الإخبارية المختلفة مع عدد أقل من الشوائب من **تكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF). الخطوة التالية هي استخدام التضمينات لتدريب خوارزمية التجميع التكتلي:

```
plt.figure() # create a new figure.

# iteratively merge points and clusters until all points belong to a single cluster. Return the the linkage of the produced tree.
linkage_emb=hierarchy.linkage(text_emb, method='ward')

hierarchy.dendrogram(linkage_emb) # visualize the linkage.
plt.show() # show the figure.
```



شكل 3.24: الرسم الشجري الهرمي لتمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT)

كما هو موضّح في الشكل 3.24، فإن أداة الرسم الشجري تشير إلى 4 عناقيد، كل واحد منها مُميز بلون مختلف.
يَستخدِم المقطع البرمجي التالي هذا المقترح لحساب العناقيد وحساب مقاييس التقييم:

```
AC_emb=AgglomerativeClustering(linkage='ward',n_clusters=4)
AC_emb.fit(text_emb)
pred_emb=AC_emb.labels_

print('\nHomogeneity score:',homogeneity_score(bbc_labels,pred_emb))
print('\nAdjusted Rand score:',adjusted_rand_score(bbc_labels,pred_emb))
print('\nCompleteness score:',completeness_score(bbc_labels,pred_emb))
```

```
Homogeneity score: 0.6741395570357063

Adjusted Rand score: 0.6919474005627763

Completeness score: 0.7965514907905805
```

إذا كانت البيانات قد تم إعادة تجميعها باستخدام العدد الصحيح من 5 عناقيد، فالعنقود الأصفر المُحدد بالشكل أعلاه سينقسم إلى اثنين، وستكون النتائج على النحو التالي:

```
AC_emb=AgglomerativeClustering(linkage='ward',n_clusters=5)
AC_emb.fit(text_emb)
pred_emb=AC_emb.labels_

print('\nHomogeneity score:',homogeneity_score(bbc_labels,pred_emb))
print('\nAdjusted Rand score:',adjusted_rand_score(bbc_labels,pred_emb))
print('\nCompleteness score:',completeness_score(bbc_labels,pred_emb))
```

```
Homogeneity score: 0.7865655030556284

Adjusted Rand score: 0.8197670431956582

Completeness score: 0.7887580797775077
```

تُظهِر النتائج أن استخدام **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) في البرمجة الاتجاهية للنصوص يَنتج عنه نتائج تجميع مُحسَّنة بالمقارنة مع **تكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF). إذا كان عدد العناقيد هو 5 **لتكرار المصطلح-تكرار المُستنَد العكسي** (TF-IDF) (القيمة الصحيحة) و4 عناقيد **لتمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT)، فإن المقاييس الثلاثة **لتمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) لا تزال هي الأعلى بفارق كبير. ثم تزداد الفجوة إذا كان العدد 5 لكلٍ من الطريقتين.

وهذا يُعدُّ دليلًا على إمكانات الشبكات العصبية، التي تسمح لها بُنيتها المتطورة بفهم الأنماط الدلالية المُعقدة في البيانات النصيّة.

تمرينات

1

حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي:	صحيحة	خاطئة
1. في التعلُّم غير الموجَّه، تُستخدم مجموعات البيانات المُعنوَنة لتدريب النموذج.	○	✓
2. يتطلب التعلُّم غير الموجَّه البرمجة الاتجاهية للبيانات.	✓	○
3. تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT) تُعدُّ أفضل من تكرار المصطلح-تكرار المستند العكسي (TF-IDF) للبرمجة الاتجاهية للكلمات.	✓	○
4. يَتبع التجميع التكتلي منهجية التصميم من أعلى إلى أسفل لتحديد العناقيد.	○	✓
5. تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT) مُدرَّبة للتنبؤ بما إذا كانت جملتان مختلفتين دلاليًا.	○	✓

2 استعرِض بعض التطبيقات التي يُستخدم فيها تقليص الأبعاد. وصِف التقنيات المستخدمة فيه.

تحديد الخصائص - التعلم المتشعب

3 اشرح وظائف البرمجة الاتجاهية لقياس تكرار المصطلح-تكرار المستند العكسي (TF-IDF).

البرمجة الاتجاهية لتكرار المصطلح تكرار المستند العكسي استند إلى حساب تكرار الكلمات ومعالجتها عبر المستندات في مجموعة البيانات.

4 لديك مصفوفة numPy تدعى 'Docs' تتضمن مستندًا نصيًّا واحدًا في كل صف. لديك كذلك مصفوفة labels تتضمن قيم كل مستند في Docs. أكمل المقطع البرمجي التالي بحيث تستخدم نموذج تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT) المُدرَّب مُسبقًا لحساب تضمينات كل الوثائق في Docs ثم استخدم أداة TSNEVisualizer تضمين المجاور العشوائي الموزَّع على شكل T لتصوير التضمينات في الفضاء ثنائي الأبعاد، باستخدام لون مختلف لكل واحد من القيم الأربعة المحتملة:

```
from sentence_transformers import SentenceTransformer
from Yellowbrick.text import TSNEVisualizer model = SentenceTransformer('all-MiniLM-L6-v2') # loads the pre-trained model.
docs_emb = model.TSNEVisualizer(Docs) # embeds the docs
tsne = colors(fit=['blue','green','red','yellow'])
tsne.text_tfidf(bbc_labels, labels)
tsne.show();
```

5 أكمل المقطع البرمجي التالي بحيث تَستخدم نموذج الكلمة إلى المتَّجه (Word2Vec) لاستبدال كل كلمة في إحدى الجُمل بأخرى تكون أكثر شبهًا بها:

```
import gensim.downloader as apl
import re
model_wv = apl.load('word2vec-google-news-300')
old_sentence='My name is John and I like basketball.'
new_sentence=''
for word in re.findall(r'\b\w\w+\b',old_sentence.lower()):
        replacement=model_wv.most_similar(positive=['apple'], topn=1)[0]
        new_sentence+= replacement
sentence=new_sentence.strip()
```

رابط الدرس الرقمي

www.ien.edu.sa

الدرس الثالث

توليد النص

توليد اللغات الطبيعية Natural Language Generation (NLG)

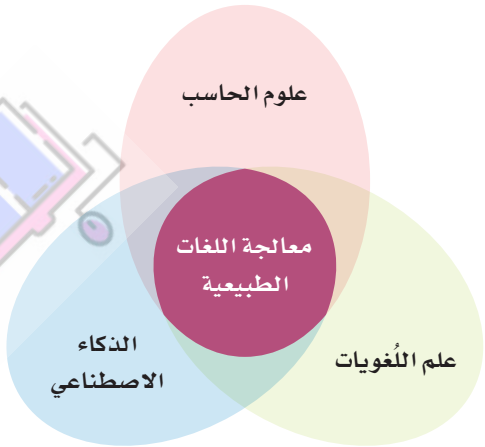
توليد اللغات الطبيعية (NLG) هو أحد فروع **معالجة اللغات الطبيعية** (NLP) التي تركِّز على توليد النصوص البشرية باستخدام خوارزميات الحاسب. الهدف من **توليد اللغات الطبيعية** (NLG) هو توليد اللغات المكتوبة أو المنطوقة بصورة طبيعية ومفهومة للبشر دون الحاجة إلى تدخل بشري. توجد العديد من المنهجيات المختلفة لتوليد اللغات الطبيعية، مثل المنهجيات المُستنِدة إلى القوالب، والمُستنِدة إلى القواعد، والمُستنِدة إلى تعلُّم الآلة.

معالجة اللغات الطبيعية (Natural Language Processing-NLP):

معالجة اللغات الطبيعية (NLP) هو أحد فروع الذكاء الاصطناعي الذي يمنح أجهزة الحاسب القدرة على محاكاة اللغات البشرية الطبيعية.

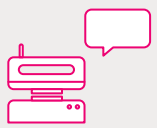
توليد اللغات الطبيعية (Natural Language Generation-NLG):

توليد اللغات الطبيعية (NLG) هي عملية توليد النصوص البشرية باستخدام **الذكاء الاصطناعي** (AI).



شكل 3.25: مُخطَّط فِنّ (Venn) لمعالجة اللغات الطبيعية (NLP)

جدول 3.4: تأثير توليد اللغات الطبيعية

	يُستخدم **توليد اللغات الطبيعية** (NLG) لتوليد المقالات والتقارير الإخبارية، والمحتوى المكتوب آليًا مما يوفّر الوقت، ويساعد الأشخاص في التركيز على المهام الإبداعية أو المهام عالية المستوى.
	يمكن الاستفادة من ذلك في تحسين كفاءة وفعالية روبوت الدردشة لخدمة العملاء وتمكينه من تقديم ردود طبيعية ومفيدة لأسئلتهم واستفساراتهم.
	يمكن الاستفادة من **توليد اللغات الطبيعية** (NLG) في تحسين إمكانية الوصول لذوي الإعاقة أو لذوي الحواجز اللغوية، بتمكينهم من التواصل مع الآلات بطريقة طبيعية وبديهية تناسبهم.

هناك أربع أنواع من **توليد اللغات الطبيعية** (NLG):

توليد اللغات الطبيعية المبني على القوالب Template-Based NLG

يتضمن توليد اللغات الطبيعية المبنيّ على القوالب استخدام قوالب مُحدَّدة مُسبقًا تحدد بنية ومحتوى النص المتولِّد. تُزَوّد هذه القوالب بمعلومات مُحدَّدة لتوليد النص النهائي. تُعدُّ هذه المنهجية بسيطة نسبيًا وتحقق فعالية في توليد النصوص للمهام المُحدَّدة والمُعرَّفة جيدًا. من ناحية أخرى، قد تواجه صعوبة مع المهام المفتوحة أو المهام التي تتطلب درجة عالية من التباين في النص المُوَلَّد. على سبيل المثال، قالب تقرير حالة الطقس ربما يبدو كما يلي: Today in **[city]**, it is **[temperature]** degrees with **[weather condition]**. (اليوم في [**المدينة**]، درجة الحرارة هي [**درجة الحرارة**] مئوية و [**حالة الطقس**].).

توليد اللغات الطبيعية المبني على القواعد Rule-Based NLG

يَستخدم توليد اللغات الطبيعية المبنيّ على القواعد مجموعة من القواعد المُحدَّدة مُسبقًا لتوليد النص. قد تحدد هذه القواعد طريقة تجميع الكلمات والعبارات لتشكيل الجُمل، أو كيفية اختيار الكلمات وفقًا للسياق المُستخدَمة فيه. عادةً تُستخدم هذه القواعد لتصميم روبوت الدردشة لخدمة العملاء. قد يكون من السهل تطبيق الأنظمة المبنية على القواعد. وفي بعض الأحيان قد تتسم بالجمود ولا تُولِّد مُخرَجات تبدو طبيعية.

توليد اللغات الطبيعية المبني على الاختيار Selection-Based NLG

يتضمن توليد اللغات الطبيعية المبنيّ على الاختيار تحديد مجموعة فرعية من الجُمل أو الفقرات لإنشاء ملخّص للنصّ الأصلي الأكبر حجمًا. بالرغم من أن هذه المنهجية لا تُولِّد نصوصًا جديدة، إلا أنها مُطبقَّة عمليًا على نطاق واسع؛ وذلك لأنّها تأخذ العينات من مجموعة من الجُمل المكتوبة بواسطة البشر، يمكن الحد من مخاطرة توليد النصوص غير المُتنبئ بها أو ضعيفة البنية. على سبيل المثال، مُولِّد تقرير الطقس المبنيّ على الاختيار قد يضم قاعدة بيانات من العبارات مثل: It is hot outside (الطقس حار بالخارج)، و The temperature is rising (درجة الحرارة ترتفع)، و Expect sunny skies (تنبؤات بطقس مُشمس).

توليد اللغات الطبيعية المبني على تعلُّم الآلة Machine Learning-Based NLG

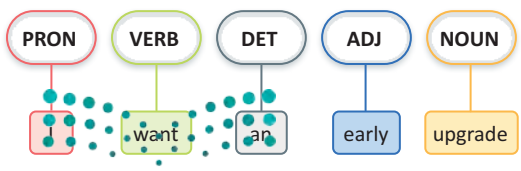
يتضمن توليد اللغات الطبيعية المبنيّ على تعلُّم الآلة تدريب نموذج تعلُّم الآلة على مجموعة كبيرة من بيانات النصوص البشرية. يتعلّم النموذج أنماط النصّ وبنيته، ومن ثم يمكنه توليد النص الجديد الذي يشبه النص البشري في الأسلوب والمحتوى. قد تكون المنهجية أكثر فعالية في المهام التي تتطلب درجة عالية من التباين في النص المُوَلَّد. وقد تتطلب المنهجية مجموعات أكبر من بيانات التدريب والموارد الحسابية.

استخدام توليد اللغات الطبيعية المبني على القوالب Using Template-Based NLG

توليد اللغات الطبيعية المبنيّ على القوالب بسيط نسبيًا وقد يكون فعالًا في توليد النصوص للمهام المُحدَّدة والمُعرَّفة مثل إنشاء التقارير أو توصيف البيانات. إحدى مميزات توليد اللغات الطبيعية المبنيّ على القوالب هو سهولة التطبيق والصيانة. يُصمِّم الأشخاص القوالب، دون الحاجة إلى خوارزميات تعلُّم الآلة المُعقدَة أو مجموعات كبيرة من بيانات التدريب. وهذا يجعل توليد اللغات الطبيعية المبنيّ على القوالب هو الخيار المناسب للمهام التي تكون ذات بنية ومحتوى نصّ محدّدين، دون الحاجة إلى إجراء تغييرات كبيرة. تَستند قوالب **توليد اللغات الطبيعية** (NLG) إلى أي بُنية لغوية مُحدَّدة مُسبقًا. إحدى الممارسات الشائعة هي إنشاء القوالب التي تتطلب كلمات بوسوم محددة كجزء من الكلام لإدراجها في الفراغات المُحدَّدة ضمن الجملة.

وسوم أقسام الكلام Part of Speech (POS) Tags

وسوم **أقسام الكلام** (Part Of Speech)، التي تُعرَّف كذلك باسم وسوم POS هي قيم تُخصَّص للكلمات في النص للإشارة إلى البناء النحوي للكلمات، أو جزء الكلام في الجملة. على سبيل المثال، قد تكون الكلمة اسمًا أو فعلًا أو صفةً أو ظرفًا، إلخ، وتُستخدَم وسوم أقسام الكلام في **معالجة اللغات الطبيعية** (NLP) لتحليل بنية النصّ وفهم معناه.



شكل 3.26: مثال على عملية وسم أقسام الكلام

تحليل بِناء الجُمل Syntax Analysis

يُستخدم تحليل بِناء الجُمل عادةً إلى جانب وسوم **أقسام الكلام** (POS) في توليد اللغات الطبيعية المبنيّ على القوالب لضمان قدرة القوالب على توليد النصوص الواقعية. يتضمن تحليل بِناء الجُمل التعرّف على أجزاء الكلام في الجُمل، والعلاقات بينها لتحديد البناء النحوي للجُملة. تتضمن الجُملة أنواعًا مختلفة من عناصر بناء الجُملة. مثل :

- **الفعل** (Predicate) هو قسم الجُملة الذي يحتوي على الفعل. وهو عادةً يعبر عمّا يقوم به الفاعل أو عمّا يحدث.
- **الفاعل** (Subject) هو قسم الجُملة الذي يُنفّذ الفعل.
- **المفعول به** (Direct Object) هو اسم أو ضمير يشير إلى الشخص أو الشيء الذي يتأثر مباشرةً بالفعل.

يَستخدِم المقطع البرمجي التالي مكتبة **ووندروورد** (Wonderwords) التي تتبع منهجية بِناء الجُمل لعرض بعض الأمثلة على توليد اللغات الطبيعية المبنيّ على القوالب.

```
%%capture

!pip install wonderwords
# used to generate template-based randomized sentences
from wonderwords.random_sentence import RandomSentence

# make a new generator with specific words
generator=RandomSentence(
                         # specify some nouns
                         nouns=["lion", "rabbit", "horse","table"],
                         verbs=["eat","run","laugh"], # specify some verbs.
                         adjectives=['angry','small']) # specify some adjectives.

# generates a sentence with the following template: [subject (noun)] [predicate (verb)]
generator.bare_bone_sentence()
```

```
'The table runs.'
```

```
# generates a sentence with the following template:
# the [(adjective)] [subject (noun)] [predicate (verb)] [direct object (noun)]
generator.sentence()
```

```
'The small lion runs rabbit.'
```

توضح الأمثلة بالأعلى أنه، بينما يُستخدَم توليد اللغات الطبيعية المبنيّ على القوالب لتوليد الجُمل وفق بُنية مُحدَّدة ومُعتمدة مُسبقًا، إلا أنّ هذه الجُمل قد لا تكون ذات مغزىً عملي. وعلى الرغم من إمكانية تحسين دقة النتائج إلى حدٍ كبير بتحديد قوالب متطورة ووضع المزيد من القيود على استخدام المفردات، إلا أن هذه المنهجية غير عملية لتوليد النصوص الواقعية على نطاقٍ واسع. فبدلًا من إنشاء القوالب المُحدَّدة مُسبقًا، تُستخدَم المنهجية الأخرى لتوليد اللغات الطبيعية القائمة على القوالب البنيةَ والمفرداتِ نفسها المُكوِّنة لأي جملة حقيقية كقالب ديناميكي متغير. تتبنى دالة paraphrase( ) هذه المنهجية.

دالة Paraphrase()

تُقسِّم الدالة في البداية النص المُكوَّن من فقرة إلى مجموعة من الجُمل. ثم تحاول استبدال كل كلمة في الجُملة بكلمة أخرى متشابهة دلاليًا. يُقيَّم التشابه الدلالي بواسطة **نموذج الكلمة إلى المتَّجه** (Word2Vec) الذي درسته في الدرس السابق. قد يوصي نموذج الكلمة إلى المتَّجه (Word2Vec) باستبدال الكلمة في الجملة بكلمة أخرى مشابهة لها، مثل: استبدال apple (تفاحة) بـ apples (تفاح)، ولتجنب مثل هذه الحالات تُستخدم دالة مكتبة fuzzywuzzy الشهيرة لتقييم تشابه المُفردات بين الكلمة الأصلية والكلمة البديلة.

الدالة نفسها مُوضحَّة بالأسفل:

```
def paraphrase(text:str, # text to be paraphrased
               stop:set, # set of stopwords
               model_wv,# Word2Vec Model
               lexical_sim_ubound:float, # upper bound on lexical similarity
               semantic_sim_lbound:float # lower bound on semantic similarity
              ):

    words=word_tokenize(text) # tokenizes the text to words

    new_words=[] # new words that will replace the old ones.

    for word in words: # for every word in the text

        word_l=word.lower() # lower-case the word.

        # if the word is a stopword or is not included in the Word2Vec model, do not try to replace it.
        if word_l in stop or word_l not in model_wv:
            new_words.append(word) # append the original word

        else: # otherwise

            # get the 10 most similar words, as per the Word2Vec model.
            # returned words are sorted from most to least similar to the original.
            # semantic similarity is always between 0 and 1.
            replacement_words=model_wv.most_similar(positive=[word_l],
topn=10)
            # for each candidate replacement word
            for rword, sem_sim in replacement_words:
                # get the lexical similarity between the candidate and the original word.
                # the partial_ratio function returns values between 0 and 100.
                # it compares the shorter of the two words with all equal-sized substrings
                # of the original word.
                lex_sim=fuzz.partial_ratio(word_l,rword)

                # if the lexical sim is less than the bound, stop and use this candidate.
                if lex_sim<lexical_sim_ubound:
                    break
```

fuzz تشير إلى مكتبة fuzzywuzzy

```
            # quality check: if the chosen candidate is not semantically similar enough to
            # the original, then just use the original word.
            if sem_sim<semantic_sim_lbound:
                new_words.append(word)
            else: # use the candidate.
                new_words.append(rword)

    return ' '.join(new_words) # re-join the new words into a single string and return.
```

المُخرَج هو إصدار مُعاد صياغته من النص المُدخَل.

يُستخدَم المقطع البرمجي التالي لاستيراد كل الأدوات اللازمة لدعم دالة ( )paraphrase وفي المربع الأبيض أدناه، تحصل على مُخرَج طريقة **إعادة الصياغة** (Paraphrase) للنص المُسند إلى المتغير text:

```
%%capture

import gensim.downloader as api # used to download and load a pre-trained Word2Vec model
model_wv = api.load('word2vec-google-news-300')

import nltk
# used to split a piece of text into words. Maintains punctuations as separate tokens
from nltk import word_tokenize
nltk.download('stopwords') # downloads the stopwords tool of the nltk library
# used to get list of very common words in different languages
from nltk.corpus import stopwords
stop=set(stopwords.words('english')) # gets the list of english stopwords

!pip install fuzzywuzzy[speedup]
from fuzzywuzzy import fuzz

text='We had dinner at this restaurant yesterday. It is very close to my
house. All my friends were there, we had a great time. The location is
excellent and the steaks were delicious. I will definitely return soon, highly
recommended!'
# parameters: target text, stopwords, Word2Vec model, upper bound on lexical similarity, lower bound
on semantic similarity
paraphrase(text, stop, model_wv, 80, 0.5)
```

```
'We had brunch at this eatery Monday. It is very close to my bungalow. All
my acquaintances were there, we had a terrific day. The locale is terrific
and the tenderloin were delicious. I will certainly rejoin quickly, hugely
advised!'
```

كما في المنهجيات الأخرى المُستنِدة إلى القوالب، يمكن تحسين النتائج بإضافة المزيد من القيود لتصحيح بعض البدائل الأقل وضوحًا والمذكورة في الأعلى. ومع ذلك، يوضح المثال أعلاه أنه يُمكن باستخدام هذه الدالة البسيطة توليد نصوص واقعية للغاية.

استخدام توليد اللغات الطبيعية المبني على الاختيار
Using Selection-Based NLG

في هذا القسم، ستستعرض منهجية عملية لاختيار نموذج من الجُمل الفرعية من وثيقة مُحدَّدة. هذه المنهجية تُجسِد استخدام ومزايا توليد اللغات الطبيعية المبنيّ على الاختيار يستند إلى لبنتين رئيسيتين:

- **نموذج الكلمة إلى المتَّجه** (Word2Vec) المُستخدَم لتحديد أزواج الكلمات المتشابهة دلاليًا.
- مكتبة Networkx الشهيرة ضمن لغة البايثون المُستخدَمة لإنشاء ومعالجة أنواع مختلفة من بيانات الشبكة.

النَّص المُدخَل الذي سيُستخدم في هذا الفصل هو مقالة إخبارية نُشرت بعد المباراة النهائية لكأس العالم 2022.

```
# reads the input document that we want to summarize
with open('article.txt',encoding='utf8',errors='ignore') as f: text=f.read()

text[:100] # shows the first 100 characters of the article
```

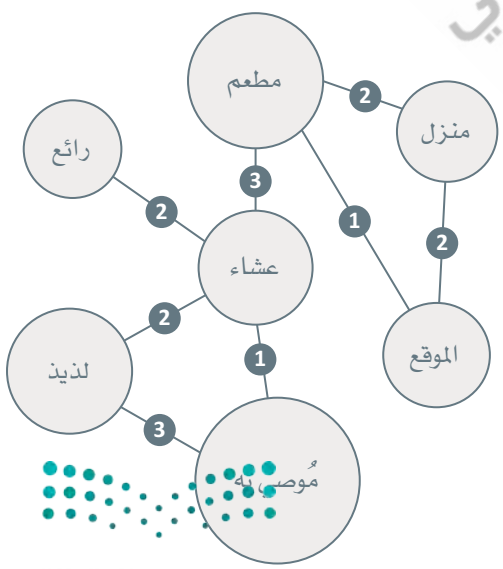
```
'It was a consecration, the spiritual overtones entirely appropriate.
Lionel Messi not only emulated '
```

في البداية، يُرمَّز النص باستخدام مكتبة re والتعبير النمطي نفسه المُستخدَم في الوحدات السابقة:

```
import re # used for regular expressions

# tokenize the document, ignore stopwords, focus only on words included in the Word2Vec model.
tokenized_doc=[word for word in re.findall(r'\b\w\w+\b',text.lower()) if word
not in stop and word in model_wv]

# get the vocabulary (set of unique words).
vocab=set(tokenized_doc)
```

مكتبة Networkx

يمكن الآن نمذجة مفردات المُستنَد في **مُخطَّط موزون** (Weighted Graph). تُوفر مكتبة Networkx في لغة البايثون مجموعة واسعة من الأدوات لإنشاء وتحليل المُخطَّطات. في توليد اللغات الطبيعية المبنيّ على الاختيار، يُساعد تمثيل مفردات الوثيقة في مُخطَّط موزون في تحديد العلاقات بين الكلمات وتسهيل اختيار العبارات والجُمل ذات الصلة. في المُخطَّط الموزون، تُمثل كل عُقدة كلمةً أو مفهومًا، وتُمثل الحواف بين العُقد العلاقات بين هذه المفاهيم. تُعبر الأوزان على الحواف عن قوة هذه العلاقات، مما يسمح لنظام توليد اللغات الطبيعية بتحديد المفاهيم الأقوى ارتباطًا. عند توليد النصوص، يُستخدم المُخطَّط الموزون للبحث عن العبارات والجُمل استنادًا إلى العلاقات بين الكلمات. على سبيل المثال، قد يَستخدِم النظام المُخطَّط للبحث عن الكلمات والعبارات الأكثر ارتباطًا لوصف كيان مُحدَّد ثم استخدام هذه الكلمات لتحديد الجُملة الأكثر ملاءمةً من قاعدة بيانات النظام.



شكل 3.27: مثال على مُخطَّط موزون لـ Networkx

دالة Build_graph( ) *fx*

تُستخدم دالة Build_graph( ) مكتبة NetworkX لإنشاء مُخطَّط يتضمن:

- عُقدة واحدة لكل كلمة ضمن مفردات محددة.
- حافة بين كل كلمتين. الوزن على الحافة يساوي التشابه الدلالي بين الكلمات، المحسوب بواسطة أداة Doc2Vec وهي أداة معالجة اللغات الطبيعية المُخصصة لتمثيل النصّ كمتَّجَه وهي تعميم لمنهجية **نموذج الكلمة إلى المتَّجَه** (Word2Vec).

تَرسم الدالة مخطّطًا ذا عُقدة واحدة لكل كلمة في المفردات المُحدَّدة. توجد كذلك حافة بين عُقدتين إذا كان تشابه **نموذج الكلمة إلى المتَّجَه** (Word2Vec) أكبر من الحد المُعطى.

```
# tool used to create combinations (e.g. pairs, triplets) of the elements in a list
from itertools import combinations
import networkx as nx # python library for processing graphs

def build_graph(vocab:set, # set of unique words
                model_wv # Word2Vec model
               ):
    # gets all possible pairs of words in the doc
    pairs=combinations(vocab,2)

    G=nx.Graph() # makes a new graph

    for w1,w2 in pairs: # for every pair of words w1, w2
        sim=model_wv.similarity(w1, w2) # gets the similarity between the two words
        G.add_edge(w1,w2,weight=sim)

    return G

# creates a graph for the vocabulary of the World Cup document
G=build_graph(vocab,model_wv)
# prints the weight of the edge (semantic similarity) between the two words
G['referee']['goalkeeper']
```

```
{'weight': 0.40646762}
```

وبالنظر إلى ذلك المُخطَّط المبني على الكلمة، يمكن تمثيل مجموعة من الكلمات المتشابهة دلاليًا في صورة عناقيد من العُقد المتصلة معًا بواسطة حواف عالية الوزن. يُطلق على عناقيد العُقد كذلك **المجتمَعات** (Communities). مُخرَج المُخطَّط هو مجموعة بسيطة من الرؤوس والحواف الموزونة. لم تُجرى عملية التجميع حتى الآن لإنشاء المجتمَعات. في الشكل 3.28 تُستخدم ألوان مختلفة لتمييز المجتمَعات في المُخطَّط المذكور بالمثال السابق.

شكل 3.28: المجتمَعات في المُخطَّط

خوارزمية لوفان Louvain Algorithm

تتضمن مكتبة Networkx العديد من الخوارزميات لتحليل المُخطَّط والبحث عن المجتمَعات. واحدة من الخيارات الأكثر فعالية هي خوارزمية لوفان التي تعمل عبر تحريك العُقد بين المجتمَعات حتى تجد بُنية المجتمع التي تمثل الربط الأفضل في الشبكة الضمنية.

fx دالة Get_communities( )

تَستخدم الدالة الآتية خوارزمية لوفان للبحث عن المجتمَعات في المُخطَّط المبنيّ على الكلمات. تَحسب الدالة كذلك مؤشر الأهمية لكل مجتمع على حده. ثم تكون المُخرَجات في صورة قاموسين:

- word_to_community الذي يربط الكلمة بالمجتمع.
- community_scores الذي يربط المجتمع بدرجة الأهمية.

الدرجة تساوي مجموع تكرار الكلمات في المجتمع. على سبيل المثال، إذا كان المجتمع يتضمن ثلاثة كلمات تظهر 5 و8 و6 مرات في النصّ، فإنّ مؤشّر المجتمع حينئذٍ يساوي 19. ومن ناحية المفهوم، يمثل المؤشر جزءًا من النصّ الذي يضُمُّه المجتمع.

```
from networkx.algorithms.community import louvain_communities
from collections import Counter # used to count the frequency of elements in a list

def get_communities( G, # the input graph
                     tokenized_doc:list): # the list of words in a tokenized document

     # gets the communities in the graph
    communities=louvain_communities(G, weight='weight')
    word_cnt=Counter(tokenized_doc)# counts the frequency of each word in the doc

    word_to_community={}# maps each word to its community

    community_scores={}# maps each community to a frequency score

    for comm in communities: # for each community
         # convert it from a set to a tuple so that it can be used as a dictionary key.
        comm=tuple(comm)
        score=0  # initialize the community score to 0.

        for word in comm: # for each word in the community

            word_to_community[word]=comm # map the word to the community

            score+=word_cnt[word] # add the frequency of the word to the community's score.

        community_scores[comm]=score # map the community to the score.

    return word_to_community, community_scores
```

```
word_to_community, community_scores = get_communities(G,tokenized_doc)
word_to_community['player'][:10] # prints 10 words from the community of the word 'team'
```

```
('champion',
 'stretch',
 'finished',
 'fifth',
 'playing',
 'scoring',
 'scorer',
 'opening',
 'team',
 'win')
```

الآن بعد ربط كل الكلمات بالمجتمع، وربط المجتمع بمؤشر الأهمية، ستكون الخطوة التالية هي استخدام هذه المعلومات لتقييم أهمية كل جملة في المُستنَد الأصلي. دالة ( )evaluate_sentences مُصمَّمة لهذا الغرض.

fx دالة ( )Evaluate_sentences

تبدأ الدالة بتقسيم المُستنَد إلى جُمل. ثم حساب مؤشر الأهمية لكل جُملة، استنادًا إلى الكلمات التي تتضمنها. تكتسب كل كلمة مؤشر الأهمية من المجتمع الذي تنتمي إليه.

على سبيل المثال، لديك جملة مكونة من خمسة كلمات w1، w2، w3، w4، w5. الكلمتان w1 وw2 تنتميان إلى مجتمع بمؤشر قيمته 25، والكلمتان w3 وw4 تنتميان إلى مجتمع بمؤشر قيمته 30، والكلمة w5 تنتمي إلى مجتمع بمؤشر قيمته 15. مجموع مؤشرات الجُمل هو 25+25+30+30+15=125. تَستخدِم الدالة بعد ذلك هذه المؤشرات لتصنيف الجُمل في ترتيب تنازلي، من الأكثر إلى الأقل أهمية.

```
from nltk import sent_tokenize # used to split a document into sentences

def evaluate_sentences(doc:str, # original document
                       word_to_community:dict,# maps each word to its community
                       community_scores:dict, # maps each community to a score
                       model_wv): # Word2Vec model

    # splits the text into sentences
    sentences=sent_tokenize(doc)
    scored_sentences=[]# stores (sentence, score) tuples

    for raw_sent in sentences: # for each sentence

        # get all the words in the sentence, ignore stopwords and focus only on words that are in the
Word2Vec model.
        sentence_words=[word
        for word in re.findall(r'\b\w\w+\b',raw_sent.lower()) # tokenizes
        if word not in stop and  # ignores stopwords
```

```
                word in model_wv] # ignores words that are not in the Word2Vec model

        sentence_score=0 # the score of the sentence

        for word in sentence_words: # for each word in the sentence

            word_comm=word_to_community[word] # get the community of this word
            sentence_score+=community_scores[word_comm] # add the score of this
community to the sentence score.

        scored_sentences.append((sentence_score,raw_sent)) # stores this sentence and
its total score

    # scores the sentences by their score, in descending order
    scored_sentences=sorted(scored_sentences,key=lambda x:x[0],reverse=True)

    return scored_sentences


scored_sentences=evaluate_sentences(text,word_to_community,community_
scores,model_wv)
len(scored_sentences)
```

```
61
```

يتضمن المُستنَد الأصلي إجمالي 61 جُملة، ويُستخدَم المقطع البرمجي التالي للعثور على الجُمل الثلاثة الأكثر أهمية من بين هذه الجُمل:

```
for i in range(3):
    print(scored_sentences[i],'\n')
```

```
(3368, 'Lionel Messi not only emulated the deity of Argentinian football,
Diego Maradona, by leading the nation to World Cup glory; he finally
plugged the burning gap on his CV, winning the one title that has eluded
him – at the fifth time of asking, surely the last time.')

(2880, 'He scored twice in 97 seconds to force extra-time; the first a
penalty, the second a sublime side-on volley and there was a point towards
the end of regulation time when he appeared hell-bent on making sure that
the additional period would not be needed.')

(2528, 'It will go down as surely the finest World Cup final of all time,
the most pulsating, one of the greatest games in history because of how
Kylian Mbappé hauled France up off the canvas towards the end of normal
time.')
```

```
print(scored_sentences[-1]) # prints the last sentence with the lowest score
print()
print(scored_sentences[30]) # prints a sentence at the middle of the scoring scale
```

```
(0, 'By then it was 2-0.')

(882, 'Di María won the opening penalty, exploding away from Ousmane
Dembélé before being caught and Messi did the rest.')
```

النتائج تؤكد أن هذه المنهجية تُحدِّد بنجاح الجُمل الأساسية التي تستنبط النقاط الرئيسة في المُستنَد الأصلي، مع تعيين مؤشرات أقل للجُمل الأقل دلالةً. تُطبَّق المنهجية نفسها كما هي لتوليد ملخّص لأي وثيقة مُحدَّدة.

استخدام توليد اللغات الطبيعية المبني على القواعد لإنشاء روبوت الدردشة

Using Rule-Based NLG to Create a Chatbot

في هذا القسم، ستُصمِّم **روبوت دردشة** (Chatbot) وفق المسار المُحدَّد الموصي به بالجمع بين قواعد المعرفة الرئيسة للأسئلة والأجوبة والنموذج العصبي **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT)، ويشير هذا إلى أن نقل التعلُّم المُستخدَم في **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) له البنية نفسها كما في **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) all-MiniLM-L6-v2 وسوف يهيَّأ بشكل دقيق لمُهِمَّة أخرى غير تحليل المشاعر، وهي: توليد اللغات الطبيعية.

1. تحميل نموذج تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات المُدرَّب مُسبقًا

Load the Pre-Trained SBERT Model

الخطوة الأولى هي تحميل نموذج **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) المُدرَّب مُسبقًا:

```
%%capture
from sentence_transformers import SentenceTransformer, util
model_sbert = SentenceTransformer('all-MiniLM-L6-v2')
```

2. إنشاء قاعدة معرفة بسيطة Create a Simple Knowledge Base

الخطوة الثانية هي إنشاء قاعدة معرفة بسيطة لتحديد النص البرمجي المكون من الأسئلة والأجوبة التي يستخدمها روبوت الدردشة. يتضمن النص البرمجي 4 أسئلة (السؤال 1 إلى 4) والأجوبة على كل سؤال (الإجابة 1 إلى 4). كل إجابة مكونة من مجموعة من الخيارات كل خيار يتكون من قيمتين فقط، تُمثِّل القيمة الثانية السؤال التالي الذي يستخدمه روبوت الدردشة. إذا كان هذا هو السؤال الأخير، ستصبح القيمة الثانية خالية. هذه الخيارات تمثل الإجابات الصحيحة المحتملة على الأسئلة المعنية بها. على سبيل المثال، الإجابة على السؤال الثاني لها خياران محتملان [None،"Python"] and [None،"Java"](["جافا"،لا يوجد] و ["البايثون"،لا يوجد]). كل خيار مُكون من قيمتين:

- النص الحقيقي للإجابة المقبولة مثل: Java (جافا) أو Courses on Marketing (دورات تدريبية في التسويق).
- مُعرِّف يشير إلى السؤال التالي الذي سيطرحه روبوت الدردشة عند تحديد هذا الخيار. على سبيل المثال، إذا حدَّد المُستخدِم خيار ["3"،"Courses on Engineering"] (["دورات تدريبية في الهندسة"، "3"])كإجابة على السؤال الأول، يكون السؤال التالي الذي سيطرحه روبوت الدردشة هو السؤال الثالث.

يمكن توسيع قاعدة المعرِفة البسيطة لتشمل مستويات أكثر من الأسئلة والأجوبة، وتجعل روبوت الدردشة أكثر ذكاءً.

```
QA={
       "Q1":"What type of courses are you interested in?",
       "A1":[["Courses in Computer Programming","2"],
             ["Courses in Engineering","3"],
             ["Courses in Marketing","4"]],

       "Q2":"What type of Programming Languages are you interested in?",
       "A2":[["Java",None],["Python",None]],

       "Q3":"What type of Engineering are you interested in?",
       "A3":[["Mechanical Engineering",None],["Electrical Engineering",None]],

       "Q4":"What type of Marketing are you interested in?",
       "A4":[["Social Media Marketing",None],["Search Engine
Optimization",None]]
}
```

fx دالة Chat()

في النهاية، تُستخدَم دالة Chat() لمعالجة قاعدة المعرِفة وتنفيذ روبوت الدردشة. بعد طرح السؤال، يقرأ روبوت الدردشة رد المُستخدِم.

- إن كان الرد مشابهًا دلاليًا لأحد خيارات الإجابات المقبولة لهذا السؤال، يُحدَّد ذلك الخيار وينتقل روبوت الدردشة إلى السؤال التالي.
- إن لم يتشابه الرد مع أيٍ من الخيارات، يُطلب من المُستخدِم إعادة صياغة الرد.

تُستخدَم دالة **تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات** (SBERT) لتقييم مؤشر التشابه الدلالي بين الرد وكل الخيارات المُرشَّحة. يُعدُّ الخيار متشابهًا إذا كان المؤشر أعلى من مُتغير الحد الأدنى sim_lbound .

```
import numpy as np # used for processing numeric data

def chat(QA:dict, # the Question-Answer script of the chatbot
         model_sbert, # a pre-trained SBERT model
         sim_lbound:float): # lower bound on the similarity between the user's response and the
closest candidate answer

    qa_id='1' # the QA id

    while True: # an infinite loop, will break in specific conditions

        print('>>',QA['Q'+qa_id]) # prints the question for this qa_id
        candidates=QA["A"+qa_id] # gets the candidate answers for this qa_id

        print(flush=True) # used only for formatting purposes
        response=input() # reads the user's response

        # embed the response
        response_embeddings = model_sbert.encode([response], convert_to_
tensor=True)
        # embed each candidate answer. x is the text, y is the qa_id. Only embed x.
```

```
        candidate_embeddings = model_sbert.encode([x for x,y in candidates],
convert_to_tensor=True)

        # gets the similarity score for each candidate
        similarity_scores = util.cos_sim(response_embeddings, candidate_
embeddings)

        # finds the index of the closest answer.
        # np.argmax(L) finds the index of the highest number in a list L
        winner_index=np.argmax(similarity_scores[0])

        # if the score of the winner is less than the bound, ask again.
        if similarity_scores[0][winner_index]<sim_lbound:
            print('>> Apologies, I could not understand you. Please rephrase
your response.')
            continue

        # gets the winner (best candidate answer)
        winner=candidates[winner_index]

        # prints the winner's text
        print('\n>> You have selected:',winner[0])
        print()

        qa_id=winner[1] # gets the qa_id for this winner

        if qa_id==None: # no more questions to ask, exit the loop
            print('>> Thank you, I just emailed you a list of courses.')
            break
```

أنظر إلى التفاعلين التاليين بين روبوت الدردشة والمُستخدِم:

التفاعل الأول

```
chat(QA,model_sbert, 0.5)
```

```
>> What type of courses are you interested in?
marketing courses
>> You have selected: Courses on Marketing
>> What type of Marketing are you interested in?
seo
>> You have selected: Search Engine Optimization
>> Thank you, I just emailed you a list of courses.
```

في التفاعل الأول، يفهم روبوت الدردشة أن المُستخدِم يبحث عن دورات تدريبية في التسويق. وكذلك، روبوت الدردشة ذكي بالقدر الكافي ليفهم أن المصطلح SEO يشبه دلاليًا مصطلح Search Engine Optimization (تحسين محركات البحث) مما يؤدي إلى إنهاء المناقشة بنجاح.

التفاعل الثاني

```
chat(QA,model_sbert, 0.5)
```

```
>> What type of courses are you interested in?

cooking classes
>> Apologies, I could not understand you. Please rephrase your response.
>> What type of courses are you interested in?

software courses

>> You have selected: Courses on Computer Programming

>> What type of Programming Languages are you interested in?

C++

>> You have selected: Java

>> Thank you, I just emailed you a list of courses.
```

في التفاعل الثاني، يفهم روبوت الدردشة أن Cooking Classes (دروس الطهي) لا تشبه دلاليًا الخيارات الموجودة في قاعدة المعرفة. وهو ذكي بالقدر الكافي ليفهم أن Software courses (الدورات التدريبية في البرمجة) يجب أن ترتبط بخيار Courses on Computer Programming (الدورات التدريبية في برمجة الحاسب). الجزء الأخير من التفاعل يسلط الضوء على نقاط الضعف: يربط روبوت الدردشة بين رد المُستخدِم C++ و Java. على الرغم من أن لغتي البرمجة مرتبطتان بالفعل ويمكن القول بأنهما أكثر ارتباطًا من لغتي البايثون و C++، إلا أن الرد المناسب يجب أن يُوضح أن روبوت الدردشة لا يتمتع بالدراية الكافية للتوصية بالدورات التدريبية في لغة C++. إحدى الطرائق لمعالجة هذا القصور هي استخدام التشابه بين المفردات بدلًا من التشابه الدلالي للمقارنة بين الردود والخيارات ذات الصلة ببعض الأسئلة.

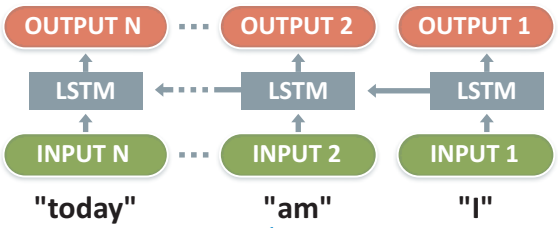
استخدام تعلُّم الآلة لتوليد نص واقعي

Using Machine Learning to Generate Realistic Text

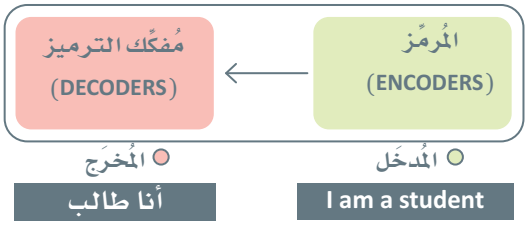
الطرائق الموضحة في الأقسام السابقة تَستخدِم القوالب، والقواعد، أو تقنيات التحديد لتوليد النصوص للتطبيقات المختلفة. في هذا القسم، ستتعرَّف على أحدث تقنيات تعلُّم الآلة المُستخدَمة في **توليد اللغات الطبيعية** (NLG).

جدول 3.5: تقنيات تعلُّم الآلة المُتقدمة المُستخدَمة في توليد اللغات الطبيعية

التقنية	الوصف
شبكة الذاكرة المُطوّلة قصيرة المدى (Long Short-Term Memory – LSTM)	تتكون شبكة **الذاكرة المُطوّلة قصيرة المدى** (LSTM) من خلايا **ذاكرة** (Memory Cells) مرتبطة ببعض. عند إدخال سلسلة من البيانات إلى الشبكة، تتولى معالجة كل عنصر في السلسلة واحدًا تلو الآخر، وتُحدِّث الشبكة خلايا الذاكرة لتوليد مُخرَج لكل عنصر على حده. شبكات **الذاكرة المُطوّلة قصيرة المدى** (LSTM) تناسب مهام **توليد اللغات الطبيعية** (NLG) لقدرتها على الاحتفاظ بالمعلومات من سلاسل البيانات (مثل التعرّف على الكلام أو الكتابة اليدوية) ومعالجة تعقيد اللغات الطبيعية.
النماذج المبنية على المحولات (Transformer-Based Models)	النماذج المبنية على المحولات هي تلك التي تفهم اللغات البشرية وتولدها، وتَستند هذه النماذج في عملها إلى تقنية **الانتباه الذاتي** (Self-Attention) التي تمكّنها من فهم العلاقات بين الكلمات المختلفة في الجُمل.



شكل 3.29: الذاكرة المُطوّلة قصيرة المدى



شكل 3.30: المُحوِّل

المُحوِّلات Transformers

المُحوِّلات مناسبة لمهام توليد اللغات الطبيعية لقدرتها على معالجة البيانات المُدخَلة المُتسلسلة بكفاءة. في نموذج المُحوِّلات، تُمرَّر البيانات المُدخَلة عبر المُرمِّز الذي يُحوّل المُدخَلات إلى تمثيل مستمر. ثم يُمرَّر التمثيل عبر مُفكِّك الترميز الذي يُولِّد التسلسل المُخرَج. إحدى الخصائص الرئيسة لهذه النماذج هي استخدام آليات الانتباه التي تسمح للنموذج بالتركيز على الأجزاء المُهِمَّة من التسلسل في حين تتجاهل الأجزاء الأقل دلالةً. أظهرت نماذج المُحوِّلات كفاءة في توليد النص عالي الدقة للعديد من مهام توليد اللغات الطبيعية بما في ذلك ترجمة الآلة، والتلخيص، والإجابة على الأسئلة.

نموذج الإصدار الثاني من المُحوِّل التوليدي مُسبَق التدريب OpenAI GPT-2 Model

في هذا القسم، سوف تستخدم **الإصدار الثاني من نموذج المُحوِّل التوليدي مُسبَق التدريب** (GPT-2) وهو نموذج لغوي قوي طورته شركة أوبن أي آي (OpenAI) لتوليد النصوص المُستنِدة إلى النص التلقيني المُدخَل بواسطة المُستخدِم. **الإصدار الثاني من المُحوِّل التوليدي مُسبق التدريب** (GPT-2 Generative Pre-training Transformer 2) مُدرَّب على مجموعة بيانات تضم أكثر من ثمان ملايين صفحة ويب ويتميز بالقدرة على إنشاء النصوص البشرية بعدَّة لغات وأساليب. بُنية **الإصدار الثاني من المُحوِّل التوليدي مُسبق التدريب** (GPT-2) المبنية على المُحوِّل تسمح بتحديد **التبعيَّات** (Dependencies) بعيدة المدى وتوليد النصوص المُتَّسقة، وهو مُدرَّب للتنبؤ بالكلمة التالية وفقًا لكل الكلمات السابقة ضمن النص، وبالتالي، يمكن استخدام النموذج لتوليد نصوص طويلة جدًا عبر التنبؤ المستمر وإضافة المزيد من الكلمات.

```
%%capture
!pip install transformers
!pip install torch
import torch # an open-source machine learning library for neural networks, required for GPT2.
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# initialize a tokenizer and a generator based on a pre-trained GPT2 model.

# used to:
# -encode the text provided by the user into tokens
# -translate (decode) the output of the generator back to text
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')

# used to generate new tokens based on the inputted text
generator = GPT2LMHeadModel.from_pretrained('gpt2')
```

يُقدَّم النص التالي كأساس يستند إليه **الإصدار الثاني من المُحوِّل التوليدي مُسبق التدريب** (GPT-2):

```
text= "We had dinner at this restaurant yesterday. It is very close to my
house. All my friends were there, we had a great time. The location is
```

```
excellent and the steaks were delicious. I will definitely return soon, highly
recommended!'

# encodes the given text into tokens
encoded_text = tokenizer.encode(text, return_tensors='pt')

# use the generator to generate more tokens.
# do_sample=True prevents GPT-2 from just predicting the most likely word at every step.
generated_tokens = generator.generate(encoded_text,
                                      max_length=200) # max number of new tokens to
generate
#decode the generates tokens to convert them to words
# skip_special_tokens=True is used to avoid special tokens such as '>' or '-' characters.
print(tokenizer.decode(generated_tokens[0], skip_special_tokens=True))
```

```
We had dinner at this restaurant yesterday. It is very close to my house.
All my friends were there, we had a great time. The location is excellent
and the steaks were delicious.I will definitely return soon, highly
recommended!

I've been coming here for a while now and I've been coming here for a while
now and I've been coming here for a while now and I've been coming here for
a while now and I've been coming here for a while now and I've been coming
here for a while now and I've been coming here for a while now and I've
been coming here for a while now and I've been coming here for a while now
and I've been coming here for a while now and I've been coming here for a
while now and I've been coming here for a while now and I've been coming
here for a while now and I've been coming here for a while now and I've
been coming here for a while now and
```

```
# use the generator to generate more tokens.
# do_sample=True prevents GPT-2 from just predicting the most likely word at every step.
generated_tokens = generator.generate(encoded_text,
                                      max_length=200, # max number of new tokens to
generate
                                      do_sample=True)

print(tokenizer.decode(generated_tokens[0],skip_special_tokens=True))
```

```
We had dinner at this restaurant yesterday. It is very close to my house.
All my friends were there, we had a great time. The location is excellent
and the steaks were delicious.I will definitely return soon, highly
recommended!

If you just found this place helpful. If you like to watch videos or
go to the pool while you're there, go for it! Good service - I'm from
Colorado and love to get in and out of this place. The food was amazing!
Also, we were happy to see the waitstaff with their great hands - I went
for dinner. I ordered a small side salad (with garlic on top), and had a
slice of tuna instead. When I was eating, I was able to get up and eat my
salad while waiting for my friend to pick up the plate, so I had a great
time too. Staff was welcoming and accommodating. Parking is cheap in this
neighborhood, and it is in the neighborhood that it needs to
```

يحقّق هذا مُخرَجات أكثر تنوعًا، مع الحفاظ على دقة وسلامة النص المولَّد، حيث يستخدِم النص مفردات غنية وهو سليم نحويًا. يسمح **الإصدار الثاني من المُحوِّل التوليدي مُسبق التدريب** (GPT-2) بتخصيص المُخرَج بشكل أفضل. يتضح ذلك عند استخدام مُتغير temperature (درجة الحرارة) الذي يسمح للنموذج بتقبل المزيد من المخاطر بل وأحيانًا اختيار بعض الكلمات الأقل احتمالًا. القيم الأعلى لهذا المُتغير تؤدي إلى نصوص أكثر تنوعًا. مثل:

```
# Generate tokens with higher diversity
generated_tokens = generator.generate(
    encoded_text, max_length=200, do_sample=True, temperature=2.0)

print(tokenizer.decode(generated_tokens[0], skip_special_tokens=True))
```

```
We had dinner at this restaurant yesterday. It is very close to my house.
All my friends were there, we had a great time. The location is excellent
and the steaks were delicious.I will definitely return soon, highly
recommended!

Worth a 5 I thought a steak at a large butcher was the end story!! We were
lucky. The price was cheap!! That night though as soon as dinner was on
my turn that price cut completely out. At the tail area they only have
french fries or kiwifet - no gravy - they get a hard egg the other day too
they call kawif at 3 PM it will be better this summer if I stay more late
with friends. When asked it takes 2 or 3 weeks so far to cook that in this
house. Once I found a place it was great. Everything I am waiting is just
perfect as usual....great prices especially at one where a single bite
would suffice or make more as this only runs on the regular hours
```

ومع ذلك، إذا كانت درجة الحرارة مرتفعة للغاية، فإنّ النموذج سيتجاهل الإرشادات الأساسية التي تظهر في **المُدخَل الأوليّ** (Original Seed) ويُولِّد مُخرجًا أقل واقعية وليس له معنى:

```
# Too high temperature leads to divergence in the meaning of the tokens
generated_tokens = generator.generate(
    encoded_text, max_length=200, do_sample=True, temperature=4.0)

print(tokenizer.decode(generated_tokens[0], skip_special_tokens=True))
```

```
We had dinner at this restaurant yesterday. It is very close to my house.
All my friends were there, we had a great time. The location is excellent
and the steaks were delicious.I will definitely return soon, highly
recommended! It has the nicest ambagas of '98 that I like; most Mexican.
And really nice steak house; amazing Mexican atmosphere to this very
particular piece of house I just fell away before its due date, no surprise
my 5yo one fell in right last July so it took forever at any number on
it being 6 (with it taking two or sometimes 3 month), I really have found
comfort/affability on many more restaurants when ordering.If you try at
it they tell ya all about 2 and three places will NOT come out before they
close them/curry. Also at home i would leave everything until 1 hour but
sometimes wait two nights waiting for 2+ then when 2 times you leave you
wait in until 6 in such that it works to
```

تمرينات

1

حدِّد الجملة الصحيحة والجملة الخاطئة فيما يلي:	صحيحة	خاطئة
1. توليد اللغات الطبيعية المبنيّ على تعلُّم الآلة يتطلب مجموعات كبيرة من بيانات التدريب والموارد الحسابية.	✓	○
2. الفعل هو نوع من وسوم أقسام الكلام (POS).	✓	○
3. في تحليل بِناء الجُمل لتوليد اللغات الطبيعية المبنيّ على القوالب، يُستخدَم التحليل بصورة منفصلة عن وسوم أقسام الكلام (POS).	○	✓
4. المجتمَعات هي عناقيد العُقد التي تُمثِّل الكلمات المختلفة دلاليًا.	○	✓
5. يصبح روبوت الدردشة أكثر ذكاءً كلما ازداد عدد مستويات الأسئلة والأجوبة المُضافة إلى قاعدة المعرِفة.	✓	○

2 قارن بين المنهجيات المختلفة لتوليد اللغات الطبيعية (NLG).

توليد اللغات الطبيعية المبني على القوالب يتضمن استخدام قوالب محددة مسبقاً تحديد بنية ومحتوى النص المتولد.
توليد اللغات الطبيعية المبني على القواعد يستخدم مجموعة من القواعد المحددة مسبقاً لتوليد النص.
توليد اللغات الطبيعية المبني على الاختيار : يتضمن مجموعة فرعية من الجمل أو الفقرات لإنشاء ملخص للنص الأصلي الأكبر حجماً.
توليد اللغات الطبيعية المبني على تعلم الآلة يتضمن تدريب نموذج تعلم الآلة على مجموعة كبيرة من بيانات النصوص البشرية،

3 حدِّد ثلاث تطبيقات مختلفة لتوليد اللغات الطبيعية (NLG).

يستخدم لتوليد المقالات والتقارير الاخبارية والمحتوى المكتوب آلياً مما يوفر الوقت ويساعد الأشخاص في التركيز على المهام الإبداعية أو المهام عالية المستوى.
يمكن الاستفادة منه في تحسين كفاءة وفعالية روبوت الدردشة لخدمة العملاء وتمكينه من تقديم ردود طبيعية ومقيدة لأسئلتهم واستفساراتهم.
يمكن الاستفادة منه في تحسين إمكانية الوصول لذوي الإعاقة أو لذوي الحواجز اللغوية. يمكنهم من التواصل مع الآلات بطريقة طبيعية وبديهية تناسبهم.

4 أكمل المقطع البرمجي التالي حتى تقبل الدالة build_graph() مفردات مُحدَّدة من الكلمات ونموذج الكلمة إلى المتَّجَه (Word2Vec) المُدرَّب لرسم مُخطَّط ذي عُقدة واحدة لكل كلمة في المفردات المُحدَّدة. يجب أن يحتوي المُخطَّط على حافة بين عُقدتين إذا كان تشابه نموذج الكلمة إلى المتَّجَه (Word2Vec) أكبر من مستوى التشابه المُعطى. يجب ألا تكون هناك أوزان على الحواف.

```
from itertools import combinations # tool used to create combinations
import networkx as nx # python library for processing graphs
def build_graph(vocab:set, # set of unique words
                model_wv, # Word2Vec model
                similarity_threshold:float
               ):
    pairs=combinations(vocab, 2) # gets all possible pairs of words in the vocabulary
    G=nx.Graph() # makes a new graph
    for w1,w2 in pairs: # for every pair of words w1,w2
        sim=model_wv.similarity(w1, w2)# gets the similarity between the two words
        if Sim>similarity_threshold:
            G.add_edge(w1,w2)
    return G
```

5 أكمل المقطع البرمجي التالي حتى تَستخدِم الدالة get_max_sim() نموذج تمثيلات ترميز الجُمل ثنائية الاتجاه من المحولات (SBERT) للمقارنة بين جُملة مُحدَّدة my_sentence وكل الجُمل الواردة في قائمة أخرى من الجُمل L. يجب أن تُعيد الدالة الجُملة ذات مُؤشر التشابه الأعلى من L1 إلى my_sentence.

```
from sentence_transformers import SentenceTransformer, util

from itertools import combinations # tool used to create combinations

model_sbert = SentenceTransformer ('all-MiniLM-L6-v2')

def get_max_sim(L1,my_sentence):

    # embeds my_sentence

    my_embedding = model_sbert, encode([my_sentence], convert_to_tensor=True)

    # embeds the sentences from L2

    L_embeddings = model_sbert.encode(L, convert_to_tensor=True)

    similarity_scores = util.cos_sim(my_embedding, L_embedding)

    winner_index=np.argmax(similarity_scores[0])

    return L1[winner_index]
```

المشروع

تصنيف النص هو عملية مكونة من خطوتين تشمل:

الخطوة الأولى: استخدام مجموعة من نصوص التدريب ذات القيم (التصنيفات) المعروفة لتدريب نموذج التصنيف.

الخطوة الثانية: استخدام نموذج التدريب للتنبؤ بالقيم لكل نصّ في مجموعة بيانات الاختبار. القيم في مجموعة بيانات الاختبار إما غير معروفة أو مخبَّأة وتُستخدم لاحقًا في عملية التحقق.

يجب تمثيل النصوص في كل من مجموعات بيانات التدريب والاختبار بالمتَّجهات قبل استخدامها. تُستخدَم أدوات CountVectorizer أو TfidfVectorizer من مكتبة **سكليرن** (Sklearn) في البرمجة الاتجاهية.

تُقدِّم مكتبة **سكليرن** (Sklearn) في لغة البايثون قائمة طويلة من نماذج التصنيف. مثل:

GradientBoostingClassifier( ) <

DecisionTreeClassifier( ) <

RandomForestClassifier( ) <

مهمتك هي استخدام مجموعة بيانات التدريب IMDB المُستخدَمة في هذا الدرس لتدريب النموذج الذي يحقق أعلى درجة من الدقة على مجموعة بيانات الاختبار IMDB (imdb_data/imdb_ test.csv). يمكنك تحقيق ذلك عبر:

1. استبدال المُصنِّف MultinomialNB بنماذج تصنيف أخرى من مكتبة **سكليرن** (Sklearn) مثل الموضحة بالأعلى.
2. إعادة تشغيل المفكرة التفاعلية لديك بعد الاستبدال، لحساب دقة كل نموذج جديد بعد تجربته.
3. إنشاء تقرير للمقارنة بين دقة كل النماذج التي جرَّبتها وتحديد النموذج الذي حقق نتائجَ دقيقة.

ماذا تعلّمت

- تصنيف النص باستخدام نماذج التعلُّم غير الموجَّه.
- تحليل النص باستخدام نماذج التعلُّم الموجَّه.
- استخدام نماذج تعلُّم الآلة لتوليد اللغات الطبيعية.
- برمجة روبوت دردشة بسيط.

المصطلحات الرئيسة

Part of Speech (POS) Tags	وسوم أقسام الكلام
Sentiment Analysis	تحليل المشاعر
Supervised Learning	التعلُّم الموجَّه
Syntax Analysis	تحليل بِناء الجُمل
Tokenization	التقسيم
Transfer Learning	التعلُّم المنقول
Unsupervised Learning	التعلُّم غير الموجَّه
Vectorization	البرمجة الاتجاهية

Black-Box predictors	متنبئات الصندوق الأسود
Chatbot	روبوت الدردشة
Cluster	عنقود
Dendrogram	الرسم الشجري
Dimensionality Reduction	تقليص الأبعاد
Document Clustering	تجميع المُستنَدات
Natural Language Generation	توليد اللغات الطبيعية
Natural Language Processing	معالجة اللغات الطبيعية