  self.user\_ref = db.collection('users').document(username)

ستخدم للوصول إلى مجموعة collection

.document('معرّف')للوصول إلى مستند معيّن داخل المجموعة.

هذا لا يحتوي على البيانات نفسها، بل هو **مرجع (Reference)** إلى موقع المستند في قاعدة بيانات Firebase (Firestore).

مثال توضيحي:  
إذا كان username = "ahmed"  
فـ user\_ref يُشير إلى المستند:

collection: "users"

document ID: "ahmed"

username = "ahmed"

self.user\_ref = db.collection('users').document(username)

print("Document ID:", self.user\_ref.id) Document ID: ahmed

print("Document Path:", self.user\_ref.path) Document Path: users/ahmed

doc = self.user\_ref.get() .get() لجلب نسخة من المستند.

يرجع كائن من النوع DocumentSnapshot.

خاصية من DocumentSnapshot، تتحقق هل المستند موجود فعلًا.

if doc.exists:

print("Document data:", doc.to\_dict()) Document data: {'password': '1234', 'type': 'student'} لتحويل بيانات المستند إلى قاموس Python (dict).

else:

print("No data found for this user.")

.delete() لحذف المستند من قاعدة البيانات:

.add(data) لإضافة مستند جديد بمعرف عشوائي داخل مجموعة:

.stream()لجلب جميع المستندات من مجموعة:

self.user\_ref.set({

            'password': password,

            'type': user\_type

        })

.set(data) لإنشاء أو تعديل بيانات المستند.

| **الدالة / الخاصية** | **وظيفتها** |
| --- | --- |
| collection(name) | الدخول لمجموعة (جدول) |
| document(id) | تحديد مستند داخل المجموعة |
| .set(data) | إنشاء أو تعديل مستند |
| .add(data) | إنشاء مستند بمعرف عشوائي |
| .get() | جلب نسخة من المستند |
| .exists | التحقق هل المستند موجود |
| .to\_dict() | تحويل بيانات المستند إلى قاموس |
| .delete() | حذف المستند |
| .stream() | جلب كل المستندات من مجموعة |

 if all(key in data for key in ['category', 'question', 'answer']):

                    qustion.append(data) #  يتحقق مما إذا كانت جميع المفاتيح المطلوبة موجودة في البيانات (category, question, answer)

| **التعبير** | **المعنى** |
| --- | --- |
| key in data | هل المفتاح موجود في القاموس؟ |
| for key in [...] | كرر العملية لكل مفتاح في القائمة |
| all(...) | هل جميع النتائج كانت صحيحة؟ |

            self.data = pd.DataFrame(qustion) #  ثم يحول قائمة الأسئلة إلى DataFrame ويخزنها في self.data

يقوم بتحويل قائمة qustion (التي تحتوي على مجموعة من القواميس) إلى جدول منظم (مثل Excel أو جدول SQL).

لأن مكتبة pandas توفر أدوات قوية جدًا للتعامل مع الجداول:

self.data['category'].unique() # الحصول على التصنيفات المختلفة

self.data[self.data['category'] == 'القبول'] # تصفية الأسئلة حسب المجال

self.data.to\_csv('file.csv') # حفظ البيانات في ملف CSV

self.data = self.data.apply(lambda x: x.astype(str) if x.dtype == object else x)

تحويل **كل الأعمدة النصية** (أي التي نوعها object) إلى **سلاسل نصية صريحة (**str**)** داخل جدول البيانات (DataFrame) لضمان التناسق.

def load\_users(self):

        users = self.user\_db.get\_all\_users()

        self.users\_table.setRowCount(len(users))

        يضبط عدد صفوف الجدول (QTableWidget) على عدد المستخدمين الموجودين.

مثال: إذا عندك 3 مستخدمين → الجدول سيصبح فيه 3 صفوف.

        for row, (username, user\_data) in enumerate(users.items()):

e() تعطيك:

row → رقم الصف (0، 1، 2...)

username → اسم المستخدم (مثل "admin")

user\_data → القيم المرتبطة بالمستخدم (مثل {"password": "1234", "type": "student"})

items() ترجع **قائمة من tuples**، كل tuple يحتوي على:

(المفتاح, القيمة)

            self.users\_table.setItem(row, 0, QTableWidgetItem(username))

            user\_type = "مسؤول" if user\_data['type'] == 'admin' else "طالب"

            self.users\_table.setItem(row, 1, QTableWidgetItem(user\_type))

            # Don't allow deleting admin user

            if user\_data['type'] == 'admin':

                self.users\_table.setItem(row, 2, QTableWidgetItem("غير مسموح"))

            else:

                delete\_btn = QPushButton("حذف")

                delete\_btn.setStyleSheet("""

                    QPushButton {

                        background-color: #f44336;

                        color: white;

                        padding: 5px;

                        border-radius: 3px;

                        border: none;

                    }

                    QPushButton:hover {

                        background-color: #d32f2f;

                    }

                """)

                delete\_btn.clicked.connect(lambda \_, u=username: self.delete\_user(u))

                self.users\_table.setCellWidget(row, 2, delete\_btn)

يتفكك هكذا:

lambda → تعريف دالة مجهولة

\_ → هذا المتغير يستقبل إشارة signal التي يرسلها الزر عند الضغط (نحن لا نستخدمه لذلك وضعنا \_)

u=username → هنا نثبت قيمة username الحالية في متغير u وقت إنشاء الـ lambda

(هذه الحركة مهمة حتى لا تتغير القيمة بعد انتهاء اللوب)

self.delete\_user(u) → الأمر الذي سيتم تنفيذه عند الضغط على الزر

class TabukUniversityQA:

    def \_\_init\_\_(self,data):

        self.arabic\_stopwords = set([

            'في', 'من', 'إلى', 'على', 'عن', 'أن', 'لا', 'ما', 'هذا', 'هذه',

            'ذلك', 'هؤلاء', 'التي', 'الذي', 'الذين', 'كان', 'يكون', 'و', 'ف',

            'ثم', 'أو', 'أم', 'لكن', 'إن', 'إذا', 'هل', 'إلا', 'قد', 'حيث',

            'بين', 'حتى', 'عند', 'بعض', 'كل', 'أي', 'هنا', 'هناك', 'مع', 'هو',

            'هي', 'هم', 'كيف', 'متى', 'أين', 'لماذا', 'كم', 'فيها', 'له', 'لها',

            'عليه', 'عليها', 'إليه', 'إليها', 'فيه', 'منه', 'منها', 'إياه', 'إياها',

            'عما', 'مما', 'غير', 'سوى', 'حين', 'الآن', 'قط', 'أمس', 'اليوم', 'غدا',

            'قبل', 'بعد'

        ])

        هذه قائمة بالكلمات الشائعة في اللغة العربية التي لا تضيف معنى مهم عند البحث أو المقارنة (مثل: "في"، "من"، "إلى").

نخزنها في set لأنها أسرع للبحث مقارنة بالقائمة (list).

سيتم استخدامها لاحقًا في دالة \_clean\_arabic\_text لإزالتها من النصوص.

        self.vectorizer = TfidfVectorizer(ngram\_range=(1, 2), min\_df=2, max\_df=0.8)

TfidfVectorizer يحول النصوص إلى تمثيل عددي (Vector) بحيث الكلمات الأكثر تميزًا تأخذ وزن أكبر.

ngram\_range=(1, 2) → يستخدم كلمات فردية (unigrams) وأزواج كلمات (bigrams).

min\_df=2 → يتجاهل أي كلمة لا تظهر على الأقل في وثيقتين.

max\_df=0.8 → يتجاهل أي كلمة تظهر في أكثر من 80% من الوثائق (لأنها غالبًا كلمة عامة جدًا).

TF\_IDF

ماذا تفعل البارامترات هنا؟

ngram\_range=(1, 2)

تعني: استخرج **Unigrams** (كلمة واحدة) و **Bigrams** (كل زوج متتالٍ من كلمتين).

مثال على نص: "شروط القبول بجامعة تبوك"

Unigrams: ["شروط", "القبول", "بجامعة", "تبوك"]

Bigrams: ["شروط القبول", "القبول بجامعة", "بجامعة تبوك"]

لماذا نضيف الأزواج (Bigrams) بجانب المفردات؟

-فهم السياق بدقة أكبر:الكلمة "جامعة" وحدها شيء، و"جامعة تبوك" شيء محدد أكثر.

لو اعتمدنا فقط على "جامعة"، ممكن النموذج يخلط بين جامعة تبوك وجامعات أخرى.

-تمييز المعاني المتشابهة:مثال: "شروط القبول" ≠ "رسوم القبول"

مثال عملي

لو عندك السؤالين:

"أين تقع جامعة تبوك؟"

"أين تقع كلية الهندسة؟"

أين, تقع, جامعة, تبوك, كلية, الهندسة :ممكن يحصل تشابه عالي لأن "أين" و"تقع" موجودة في الاثنين.

أين تقع, تقع جامعة, جامعة تبوك, تقع كلية, كلية الهندسة:الآن النموذج يفرق بين "جامعة تبوك" و"كلية الهندسة".

الفائدة: الـ bigrams تحفظ **المعنى المركّب** (مثل “شروط القبول”) بدل تفتيته لكلمتين منفصلتين قد تفقد السياق.

min\_df=2

تجاهل أي n‑gram لا يظهر في وثيقتين أو أكثر.

يقلّل الضجيج (كلمات نادرة/أخطاء إملائية) التي لا تساعد في التعميم.

مثال: لو لديك 100 سؤال/إجابة كوثائق تدريب، وظهرت عبارة “رقم الجوال الأكاديمي” في وثيقة واحدة فقط → سيتم تجاهلها من القاموس.

max\_df=0.8

تجاهل أي n‑gram يظهر في أكثر من 80% من الوثائق.

لأن الكلمات شديدة الشيوع (مثل “جامعة”، “طالب”، “سؤال”) لا تميّز وثيقة عن أخرى.

هذا يُشبه مرشّح “stop words” ولكن بطريقة معتمدة على الإحصاء داخل مجموعتك.

كيف يعمل الـ Vectorizer خطوة بخطوة داخل مشروعك؟

(بعد ما تنادي fit على نصوص combined\_text)

بناء القاموس: يقرأ كل الوثائق، يستخرج unigrams و bigrams، ثم يطبّق مرشّحات min\_df و max\_df ليقرر ما الكلمات/العبارات التي ستدخل في المفردات (vocabulary).

حساب IDF لكل مصطلح في القاموس.

لاحقًا عند transform:

يحسب TF لكل مصطلح في الوثيقة.

يضرب TF بـ IDF → يعطيك متّجه TF‑IDF.

المتجهات الناتجة (لمجمل الأسئلة/الأجوبة) تُخزَّن في self.tfidf\_matrix وتُستخدم لاحقًا مع cosine similarity لإيجاد أقرب الإجابات.

لماذا هذه الإعدادات مناسبة للغة العربية هنا؟

bigrams تحافظ على التركيبات الشائعة (شروط القبول، بوابة القبول)، وهذا مهم لأن المعنى غالبًا في التعبير المركّب.

min\_df=2 يزيل ضجيج العبارات النادرة.

max\_df=0.8 يزيل العموميات التي لا تميّز سؤالًا عن آخر.

ملاحظة عملية

**بعد تعريف الـ** vectorizer:

يتم استدعاء self.vectorizer.fit(self.data['combined\_text']) في train\_model.

ثم self.vectorizer.transform(...) لإنتاج المتّجهات.

بعدها تُستخدم cosine\_similarity لمقارنة سؤال المستخدم مع قاعدة الأسئلة/الأجوبة.

<https://chatgpt.com/s/t_6895774b53208191b373f758afe70dc1>

تمام—هذا السطر يهيّئ مُحوِّل نصوص اسمه ****TfidfVectorizer**** مع إعدادات محددة:

        self.data = data  # DataFrame containing the questions and answers

data هو DataFrame يحتوي أعمدة:

        self.tfidf\_matrix = None

لاحقًا ستخزن هنا المصفوفة العددية التي تمثل كل سؤال+إجابة بعد تحويلهم عبر TF-IDF.

        self.preprocess\_data()

ينظف النصوص (إزالة التشكيل، الحروف الخاصة، والكلمات الشائعة) ويضيف أعمدة جديدة في self.data:

processed\_question

processed\_answer

combined\_text (دمج السؤال مع الإجابة في نص واحد).

        self.train\_model()

يأخذ النصوص المعالجة ويحوّلها إلى متجهات رقمية (TF-IDF) ويخزنها في self.tfidf\_matrix.

هذا يجعل الكائن جاهزًا مباشرة للإجابة على الأسئلة.

        print("Data loaded successfully")

def preprocess\_data(self):

        self.data['processed\_question'] = self.data['question'].apply(self.\_clean\_arabic\_text)

هذه الدالة تجهز البيانات قبل تدريب نموذج TF-IDF:

تنظف النصوص من الرموز والكلمات الشائعة.

تحفظ النسخة النظيفة في أعمدة جديدة.

تدمج السؤال والإجابة في نص واحد (لأننا نريد أن يعتمد البحث على الاثنين معًا).

        self.data['processed\_answer'] = self.data['answer'].apply(self.\_clean\_arabic\_text)

        self.data['combined\_text'] = self.data['processed\_question'] + " " + self.data['processed\_answer']

دمج النصين مع مسافة " " بينهما.

الهدف: عند تدريب TF-IDF، النموذج يرى السياق الكامل للسؤال مع الإجابة، مما يحسّن المطابقة.

--الأسباب الرئيسية لدمج السؤال + الإجابة

1-زيادة كمية البيانات لكل سجل

نموذج TF-IDF يعتمد على تكرار الكلمات عبر جميع المستندات.

إذا أخذنا السؤال فقط، فقد يكون قصيرًا جدًا (مثل: "متى يبدأ التسجيل؟") وبالتالي كمية الكلمات قليلة جدًا.

بدمج الإجابة، يصبح لدينا محتوى أكبر وأغنى.

-2

تحسين فهم العلاقة بين السؤال والإجابة

بعض الأسئلة قد تحتوي على كلمات عامة جدًا، لكن الإجابة تضيف تفاصيل مهمة تساعد على التمييز.

مثال:

سؤال: "ما هي الرسوم؟"

إجابة: "الرسوم الدراسية لجامعة تبوك هي 5000 ريال للفصل."

الدمج يعطي: "ما هي الرسوم الرسوم الدراسية لجامعة تبوك هي 5000 ريال للفصل"

-3رفع دقة البحث عن الإجابة

عند استلام سؤال جديد من المستخدم، نريد أن يجد النموذج الإجابة الصحيحة حتى لو كانت صيغة السؤال مختلفة.

إذا كان النموذج يعرف محتوى الإجابة أيضًا، يمكن أن يتعرف على التشابه من خلال الكلمات الموجودة في الإجابة.

-4التعامل مع الأسئلة المتشابهة ذات الإجابات المختلفة

قد يكون لدينا أكثر من سؤال متشابه لكن بإجابات مختلفة، ودمج الإجابة مع السؤال يجعل النموذج يفرق بينها.

مثال:

سؤال: "ما هي شروط القبول في البكالوريوس؟" + إجابة البكالوريوس.

سؤال: "ما هي شروط القبول في الماجستير؟" + إجابة الماجستير.

  def \_clean\_arabic\_text(self, text):

        if not isinstance(text, str):

النص مش من نوع String (مثلاً NaN أو رقم)، نرجّع نص فاضي "" عشان نتجنب الأخطاء لاحقًا.

            return ""

        text = text.replace('أ', 'ا').replace('إ', 'ا').replace('آ', 'ا')

توحيد أشكال الألف → كل أنواع الألف المختلفة تصبح حرف "ا".

        text = text.replace('ة', 'ه').replace('ى', 'ي')

        text = re.sub(r'[^\w\s\u0600-\u06FF،؛؟]', ' ', text)

[...] → مجموعة الحروف المسموح بها أو غير المسموح بها.

^ داخل الأقواس → يعني "ليس" (أي استبعاد ما يلي).

\w → أي حرف أو رقم أو الشرطة السفلية \_ (بالـ Unicode).

\s → أي مسافة بيضاء (مثل space, tab, newline).

\u0600-\u06FF → جميع الحروف العربية في جدول Unicode.

أي حرف ليس حرفًا أو رقمًا أو مسافة أو حرفًا عربيًا أو علامة ترقيم عربية، استبدله بمسافة.

،؛؟ → علامات الترقيم العربية (الفاصلة العربية، الفاصلة المنقوطة، وعلامة الاستفهام).

يحذف أي رموز أو حروف غير عربية أو غير مسموحة، مع الإبقاء على:

الحروف العربية (\u0600-\u06FF)

الشرطة السفلية \_ إذا كانت موجودة

الفراغات

علامات الترقيم العربية (، ؛ ؟)

مثال: "شروط القبول! @ جامعة" → "شروط القبول جامعة"

        text = re.sub(r'[\u064b-\u0652\u0640]', '', text)

شرح النمط [\u064b-\u0652\u0640]

[...] → مجموعة الحروف المطلوب استهدافها

\u064b-\u0652 → هذا نطاق (Range) في Unicode يمثل حركات التشكيل العربية:

064B → تنوين فتح ً

064C → تنوين ضم ٌ

064D → تنوين كسر ٍ

064E → فتحة َ

064F → ضمة ُ

0650 → كسرة ِ

0651 → شدة ّ

0652 → سكون ْ

\u0640 → الكشيدة ـ (علامة مد تستخدم في الكتابة للزخرفة فقط).

يحذف التشكيل والمدود:

الفتحة، الضمة، الكسرة، السكون، التنوين… إلخ.

المدّة (ـ).

        text = re.sub(r'[a-zA-Z\d]', '', text)

يحذف أي حروف لاتينية أو أرقام.

مثال: "جامعة Tabuk 2024" → "جامعة "

هذا السطر يحذف الرموز وعلامات الترقيم من النص، ويستبدلها بمسافة " ".

re.sub(pattern, replacement, text) من مكتبة re (التعبيرات النمطية).

تقوم بـ البحث والاستبدال في النص. pattern = النمط اللي نبحث عنه. replacement = الشيء اللي نستبدله به (هنا مسافة " ").

r'[^\w\s]'

\w = أي حرف أو رقم (بالإنجليزية أو بالعربية).

\s = أي مسافة بيضاء (مثل الفراغ).

^ = نفي (يعني "ليس").

[^\w\s] = كل شيء ما عدا الحروف والأرقام والمسافات.

✅ أي شيء مثل:

علامات الترقيم: ؟ ! . ، ; :

رموز: @ # $ % ^ & \* ( ) [ ]

سيتم حذفه.

مثال عملي:

        text = re.sub(r'\s+', ' ', text).strip()

يحوّل أي تكرار للمسافات إلى مسافة واحدة فقط.

ثم يحذف المسافات من بداية ونهاية النص.

        words = [word for word in text.split() if word not in self.arabic\_stopwords]

يقسم النص إلى كلمات.

يحذف الكلمات الموجودة في قائمة self.arabic\_stopwords (الكلمات الشائعة مثل "في"، "من"، "إلى").

        return ' '.join(words)

def train\_model(self):

        if self.data is None:

قبل أي تدريب، نتأكد أن self.data فيها بيانات.

            raise ValueError("No data loaded")

        self.vectorizer.fit(self.data['combined\_text'])

fit هنا تقوم بـ:

قراءة جميع النصوص.

استخراج كل الكلمات (و الـ n-grams إذا محددة).

بناء قاموس Vocabulary للكلمات.

حساب قيم IDF لكل كلمة.

        self.tfidf\_matrix = self.vectorizer.transform(self.data['combined\_text'])

بعد بناء القاموس، نحول كل نص في combined\_text إلى متجه أرقام يمثل وزن كل كلمة بناءً على TF-IDF.

الناتج self.tfidf\_matrix هو مصفوفة كبيرة أبعادها:

fit

يبني القاموس (vocabulary\_) للكلمات.

يحسب IDF لكل كلمة.

لا ينتج المصفوفة الرقمية النهائية.

transform

يستخدم القاموس و قيم IDF لحساب TF-IDF لكل مستند.

ينتج المصفوفة الرقمية الجاهزة للاستخدام في حساب التشابه.

<https://chatgpt.com/s/t_68957f77fbf88191a5c563f6fd0e6168>

مثال

عندنا جدول أسئلة وأجوبة، ننظفها، ندمجها، ثم ندربها بـ [TF-IDF.](TF.docx)

البيانات الخام (قبل التنظيف)

| **question** | **answer** |
| --- | --- |
| ما هي شروطُ القبول؟ | يجب التسجيل عبر البوابة الإلكترونية. |
| كم رسومُ الدراسة؟ | الرسوم هي 5000 ريال لكل فصل دراسي. |

بعد التنظيف (\_clean\_arabic\_text)

| processed\\_question | processed\\_answer |

| ------------------- | ------------------------- |

| شروط قبول | تسجيل عبر بوابه الكترونيه |

| رسوم دراسه | رسوم 5000 ريال لكل فصل |

الدمج (combined\_text)

شروط قبول تسجيل عبر بوابه الكترونيه

رسوم دراسه رسوم 5000 ريال لكل فصل

5️⃣ النتيجة (تصوّر مبسّط لمصفوفة TF-IDF)

| **المصطلح** | **شروط** | **قبول** | **تسجيل** | **رسوم** | **دراسه** | **...** |
| --- | --- | --- | --- | --- | --- | --- |
| **Doc 1** (شروط قبول...) | 0.5 | 0.5 | 0.4 | 0.0 | 0.0 | ... |
| **Doc 2** (رسوم دراسه...) | 0.0 | 0.0 | 0.0 | 0.6 | 0.6 | ... |

https://chatgpt.com/s/t\_6895831ce224819195b99205e1c4f50b

ask\_question(self, question, threshold=0.25): threshold: الحد الأدنى لقيمة التشابه المقبولة (افتراضي 0.25)

        try:

            if not question or not isinstance(question, str):

                return "الرجاء إدخال سؤال صحيح"

            clean\_question = self.\_clean\_arabic\_text(question)

نفس خطوات التنظيف المستخدمة أثناء التدريب (توحيد الحروف، حذف التشكيل، إزالة stopwords… إلخ) لضمان تطابق المعالجة بين بيانات التدريب وسؤال المستخدم.

            if len(clean\_question.split()) < 2:

                return "السؤال قصير جدًا، يرجى تقديم المزيد من التفاصيل"

            results = self.get\_most\_similar(clean\_question)

ينادي الميثود التالية get\_most\_similar اللي:

تحوّل السؤال إلى متجه TF‑IDF

تقارن مع self.tfidf\_matrix باستخدام cosine similarity

ترجع قائمة بأفضل النتائج (السؤال/الإجابة/التشابه/المجال)

            if not results or results[0]['similarity'] < threshold:

لو ما فيه نتائج، أو أعلى تشابه أقل من العتبة 0.25 → نرجّع رسالة اعتذار (علشان ما نعرض إجابة “أي كلام”).

الـ threshold مهم عشان نضمن جودة المطابقة. تقدر ترفعه لو تبي نتائج أدق وأقل.

                return "عذرًا، لا أمتلك إجابة كافية على هذا السؤال. يرجى الرجوع إلى الموقع الرسمي لجامعة تبوك لمزيد من المعلومات."

            answer = results[0]['answer']

            if not answer.strip():

                return "عذرًا، الإجابة فارغة في قاعدة البيانات"

            return f"{answer}\n\n(هذه الإجابة متعلقة بمجال: {results[0]['category']})"

        except Exception as e:

            print(f"Full error: {e}")

            return "حدث خطأ غير متوقع أثناء معالجة سؤالك. يرجى المحاولة مرة أخرى."

def get\_most\_similar(self, query, top\_n=3):

        try:

            query\_vec = self.vectorizer.transform([query])

أخذ نص السؤال (بعد التنظيف في ask\_question) ويحوله إلى **متجه أوزان** باستخدام نفس الـ vectorizer المدرب.:fit → للتدريب الأولي على البيانات المرجعية (المستندات القديمة).

transform → لتحويل أي نص جديد (سؤال) إلى متجه TF-IDF باستخدام القاموس و الـ IDF التي تم تعلمها مسبقًا.

يقوم بتحويل السؤال الجديد (query) إلى متجه أرقام بناءً على القاموس (Vocabulary) ووزن الكلمات (IDF) الذي تعلمه من البيانات.

الشكل

['جامعة', 'تبوك', 'شروط', 'القبول', 'رسوم']

"ما هي رسوم القبول في جامعة تبوك"

فـ query\_vec سيكون متجه TF-IDF مثل:

[0.4, 0.2, 0.0, 0.3, 0.5]كل خانة تمثل وزن الكلمة المقابلة في القاموس.

            similarities = cosine\_similarity(query\_vec, self.tfidf\_matrix).flatten()

قارن متجه السؤال مع كل الصفوف في self.tfidf\_matrix (الأسئلة والإجابات الموجودة في القاعدة).

cosine\_similarity يعطي درجة من 0 إلى 1:

1 → متطابق تمامًا

0 → لا يوجد أي تشابه

flatten() تجعلها مصفوفة 1D (قائمة عادية).ي

query\_vec: متجه واحد يمثل السؤال الجديد.

self.tfidf\_matrix: مصفوفة كبيرة حجمها:

عدد المستندات × حجم القاموس

[

[0.5, 0.1, 0.0, 0.3, 0.2], ← المستند 1

[0.0, 0.4, 0.5, 0.2, 0.0], ← المستند 2

[0.3, 0.2, 0.0, 0.4, 0.5] ← المستند 3

]

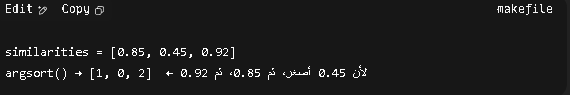
similarities = [0.85, 0.45, 0.92]

كل رقم هو نسبة التشابه مع مستند معين.

            top\_indices = similarities.argsort()[-top\_n:][::-1]

ترتيب النتائج واختيار الأفضل

argsort() ترجع فهارس العناصر مرتبة تصاعديًا



[-top\_n:] يأخذ آخر n فهارس (أعلى الدرجاoمثلاً top\_n = 2:

[0, 2]

[::-1] يعكس الترتيب ليصبح تنازلي (من الأعلى إلى الأقل).

[2, 0]

الشكل النهائي:

top\_indices = [2, 0]

  return [{

                'question': self.data.iloc[idx]['question'],

                'answer': self.data.iloc[idx]['answer'],

                'similarity': similarities[idx],

                'category': self.data.iloc[idx]['category']

            } for idx in top\_indices if similarities[idx] > 0.1]

for idx in top\_indices → يمر على كل فهرس (index) موجود في قائمة top\_indices التي تم ترتيبها من الأعلى للأقل تشابه.

if similarities[idx] > 0.1 → شرط فلترة: لا نضيف إلا النتائج التي درجة التشابه لها أكبر من 0.1. حتى يتفادى الضوضاء.

{...} → لكل نتيجة مطابقة، ننشئ قاموس يحتوي على معلومات السؤال والإجابة والتشابه والمجال.

يمر على كل idx في top\_indices:

idx = 2

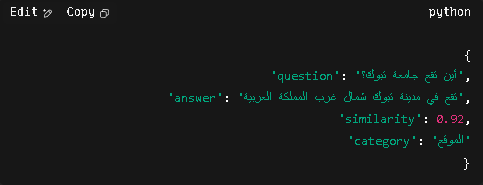
self.data.iloc[2]['question'] → "أين تقع جامعة تبوك؟"

self.data.iloc[2]['answer'] → "تقع في مدينة تبوك شمال غرب المملكة العربية"

similarities[2] → 0.92

self.data.iloc[2]['category'] → "الموقع"

ينتج قاموس:



[مهمممممممممم](فهمت قصدك 👍، أنت تقول.docx)

مثال:

https://chatgpt.com/s/t\_689749a32628819192d8f3499a56bca2

        except Exception as e:

            print(f"Similarity error: {e}")

            return None

class CreateAccountDialog(QDialog):

    def \_\_init\_\_(self, user\_db, parent=None):

        super().\_\_init\_\_(parent)

#هذا يضمن أن خصائص النافذة الحوارية في PyQt (مثل الإغلاق، الترتيب، التعامل مع الـ parent) يتم تهيئتها.

        self.user\_db = user\_db

        self.setWindowTitle("إنشاء حساب جديد")

        self.setMinimumWidth(400)

        self.setLayoutDirection(Qt.RightToLeft)

        self.init\_ui()

    def init\_ui(self):

        # Set green background

        with open("style/MainWindows.qss", "r", encoding="utf-8") as f:

            create\_account\_style = f.read()

        self.setStyleSheet(create\_account\_style)

        layout = QVBoxLayout()

        layout.setAlignment(Qt.AlignRight)

        self.setLayout(layout)

        form = QFormLayout()

        form.setLabelAlignment(Qt.AlignRight)

        form.setFormAlignment(Qt.AlignRight)

        form.setSpacing(15)

        self.username\_input = QLineEdit()

        self.username\_input.setPlaceholderText("اسم المستخدم")

        form.addRow(QLabel("اسم المستخدم:"), self.username\_input)

        self.password\_input = QLineEdit()

        self.password\_input.setPlaceholderText("كلمة المرور")

        self.password\_input.setEchoMode(QLineEdit.Password)

        form.addRow(QLabel("كلمة المرور:"), self.password\_input)

        self.confirm\_password\_input = QLineEdit()

        self.confirm\_password\_input.setPlaceholderText("تأكيد كلمة المرور")

        self.confirm\_password\_input.setEchoMode(QLineEdit.Password)

        form.addRow(QLabel("تأكيد كلمة المرور:"), self.confirm\_password\_input)

        self.user\_type\_combo = QComboBox()

        self.user\_type\_combo.addItems(["طالب"])

        form.addRow(QLabel("نوع المستخدم:"), self.user\_type\_combo)

        layout.addLayout(form)

        buttons = QDialogButtonBox(QDialogButtonBox.Ok | QDialogButtonBox.Cancel)

        buttons.accepted.connect(self.validate\_input)

        buttons.rejected.connect(self.reject)

        layout.addWidget(buttons)

    def validate\_input(self):

        username = self.username\_input.text().strip()

        password = self.password\_input.text().strip()

        confirm\_password = self.confirm\_password\_input.text().strip()

| **نوع الأداة** | **طريقة جلب النص** |
| --- | --- |
| QLineEdit | .text() |
| QTextEdit | .toPlainText() أو .toHtml() |
| QComboBox | .currentText() |
| QSpinBox | .value() |
| QDateEdit / QTimeEdit | .date() / .time() |

        user\_type = "student" if self.user\_type\_combo.currentText() == "طالب" else "admin"

        if not username or not password:

            QMessageBox.warning(self, "تحذير", "الرجاء إدخال اسم المستخدم وكلمة المرور")

            return

        if len(username) < 3:

            QMessageBox.warning(self, "تحذير", "اسم المستخدم يجب أن يكون 3 أحرف على الأقل")

            return

        if len(password) < 4:

            QMessageBox.warning(self, "تحذير", "كلمة المرور يجب أن تكون 4 أحرف على الأقل")

            return

        if password != confirm\_password:

            QMessageBox.warning(self, "تحذير", "كلمة المرور وتأكيدها غير متطابقين")

            return

        if self.user\_db.add\_user(username, password, user\_type):

            QMessageBox.information(self, "نجاح", "تم إنشاء الحساب بنجاح")

            self.accept()

        else:

            QMessageBox.warning(self, "تحذير", "اسم المستخدم موجود مسبقًا")

class LoginPage(QWidget):

| **الخاصية** | **QWidget** | **QDialog** |
| --- | --- | --- |
| نوع النافذة | أساسية/عامة | منبثقة (Dialog) |
| يدعم accept/reject | ❌ لا | ✅ نعم |
| الاستخدام الشائع | النوافذ الرئيسية/الصفحات | إدخال بيانات، تأكيد، تحذير |
| التحكم في التطبيق | لا يوقف النوافذ الأخرى | يمكنه إيقاف النوافذ الأخرى (Modal) |

    def \_\_init\_\_(self, parent=None):

        super().\_\_init\_\_(parent)

        self.parent = parent

        self.user\_db = UserDatabase()

        self.setLayoutDirection(Qt.RightToLeft)

        self.init\_ui()

    def init\_ui(self):

        layout = QVBoxLayout()

        layout.setAlignment(Qt.AlignCenter)

        layout.setContentsMargins(50, 50, 50, 50)

        layout.setSpacing(30)

        # Logo

        logo\_label = QLabel()

        try:

            response = requests.get("https://iconape.com/wp-content/files/vf/193315/png/193315.png")

            logo\_pixmap = QPixmap()

            logo\_pixmap.loadFromData(response.content)

            logo\_pixmap = logo\_pixmap.scaled(180, 180, Qt.KeepAspectRatio, Qt.SmoothTransformation)

            logo\_label.setPixmap(logo\_pixmap)

        except:

            logo\_label.setText("جامعة تبوك")

            logo\_label.setFont(QFont('Arial', 24, QFont.Bold))

            logo\_label.setStyleSheet("color: #2e7d32;")

        logo\_label.setAlignment(Qt.AlignCenter)

        layout.addWidget(logo\_label)

        # Title

        title = QLabel("Ubot Academic Assistant")

        title.setFont(QFont('Arial', 18, QFont.Bold))

        with open("style/stylesheetTitle.qss", "r", encoding="utf-8") as f:

            title\_style = f.read()

        title.setStyleSheet(title\_style)

        title.setAlignment(Qt.AlignCenter)

        layout.addWidget(title)

        # Login Form

        form = QFormLayout()

        form.setFormAlignment(Qt.AlignRight)

        form.setLabelAlignment(Qt.AlignRight)

        form.setHorizontalSpacing(20)

        form.setVerticalSpacing(15)

        self.username = QLineEdit()

        self.username.setPlaceholderText("اسم المستخدم")

        self.username.setMinimumWidth(300)

        self.username.setMinimumHeight(40)

        form.addRow(QLabel("اسم المستخدم:"), self.username)

        self.password = QLineEdit()

        self.password.setPlaceholderText("كلمة المرور")

        self.password.setEchoMode(QLineEdit.Password)

        self.password.setMinimumWidth(300)

        self.password.setMinimumHeight(40)

        form.addRow(QLabel("كلمة المرور:"), self.password)

        layout.addLayout(form)

        # Login Button

        login\_btn = QPushButton("تسجيل الدخول")

        login\_btn.setFont(QFont('Arial', 14))

        login\_btn.setMinimumHeight(45)

        login\_btn.clicked.connect(self.login)

        layout.addWidget(login\_btn)

        # Create Account Button

        create\_account\_btn = QPushButton("إنشاء حساب جديد")

        create\_account\_btn.setFont(QFont('Arial', 12))

        create\_account\_btn.setStyleSheet("""

            QPushButton {

                color: #ACADB9;

                border: none;

                padding: 5px;

                background-color: transparent;

            }

            QPushButton:hover {

                text-decoration: underline;

            }

        """)

        create\_account\_btn.clicked.connect(self.show\_create\_account)

        layout.addWidget(create\_account\_btn, alignment=Qt.AlignCenter)

        self.setLayout(layout)

    def login(self):

        username = self.username.text().strip()

        password = self.password.text().strip()

        if not username or not password:

            QMessageBox.warning(self, "تحذير", "الرجاء إدخال اسم المستخدم وكلمة المرور")

            return

        user\_type = self.user\_db.validate\_user(username, password)

        if user\_type:

            self.parent.current\_user = username

            self.parent.user\_type = user\_type

            self.parent.switch\_to\_main\_app()

        else:

            QMessageBox.warning(self, "تحذير", "اسم المستخدم أو كلمة المرور غير صحيحة")

    def show\_create\_account(self):

        dialog = CreateAccountDialog(self.user\_db, self)

        dialog.exec\_()

exec\_() يشغل النافذة بأسلوب Modal (نافذة حوارية مغلقة):

المستخدم لا يستطيع التفاعل مع نافذة تسجيل الدخول قبل إغلاق نافذة إنشاء الحساب.

تنتظر حتى يضغط المستخدم OK أو Cancel.

ترجع القيمة:

QDialog.Accepted لو المستخدم أكّد العملية (self.accept() في النافذة الفرعية).

QDialog.Rejected لو المستخدم ألغى (self.reject() أو أغلق النافذة).

class QAApp(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.user\_db = UserDatabase()

        self.db\_manager = DatabaseManager()

        self.qa\_engine = TabukUniversityQA(self.db\_manager.data)

        self.current\_user = None

        self.user\_type = None

        self.user\_info\_label = None

        # Set green theme

        with open("style/MainWindows.qss", "r", encoding="utf-8") as f:

            main\_style = f.read()

        self.setStyleSheet(main\_style)

        # Set RTL direction for the main window

        self.setLayoutDirection(Qt.LeftToRight)

        # Create stacked widget for multiple pages

        self.stacked\_widget = QStackedWidget()

QStackedWidget هي حاوية (Container) تسمح لك بوضع أكثر من صفحة (QWidget) فوق بعض، لكن تظهر صفحة واحدة فقط في كل لحظة.

ميزة QStackedWidget هي أنه بإمكانك التنقل بين الصفحات بسهولة عن طريق:

في مشروعك، استخدمناها لاحتواء:

صفحة تسجيل الدخول (login\_page)

الصفحة الرئيسية (main\_page)

وبكذا نستطيع التنقل بين الصفحتين بدون إنشاء نافذة جديدة.

        self.stacked\_widget.setLayoutDirection(Qt.RightToLeft)

        self.setCentralWidget(self.stacked\_widget)

        # Create pages

        self.login\_page = LoginPage(self)

        self.main\_page = QWidget()

        self.main\_page.setLayoutDirection(Qt.RightToLeft)

        # Add pages to stacked widget

        self.stacked\_widget.addWidget(self.login\_page)

        self.stacked\_widget.addWidget(self.main\_page)

        self.init\_main\_ui()

        # Start with login page

        self.stacked\_widget.setCurrentWidget(self.login\_page)

        self.setWindowTitle("Ubot")

        self.setMinimumSize(1000, 750)

        # Set window icon

        try:

            response = requests.get("https://iconape.com/wp-content/files/vf/193315/png/193315.png")

            icon\_pixmap = QPixmap()

            icon\_pixmap.loadFromData(response.content)

            self.setWindowIcon(QIcon(icon\_pixmap))

        except:

            pass

    def switch\_to\_main\_app(self):

        # Reinitialize the main page to update the user info

        self.main\_page = QWidget()

ينشئ صفحة جديدة من نوع QWidget.

يضبط اتجاه النصوص من اليمين لليسار (RTL) حتى تتناسب مع اللغة العربية.

        self.main\_page.setLayoutDirection(Qt.RightToLeft)

        self.stacked\_widget.removeWidget(self.stacked\_widget.widget(1))

الـ StackedWidget فيه عنصرين:

صفحة تسجيل الدخول (index 0)

الصفحة الرئيسية القديمة (index 1)

هنا يحذف العنصر الموجود في index 1 لأنه نسخة قديمة قبل تسجيل الدخول.

        self.stacked\_widget.addWidget(self.main\_page)

يضيف النسخة الجديدة للصفحة الرئيسية داخل الـ StackedWidget.

يستدعي init\_main\_ui() حتى يرسم تصميم الصفحة الجديدة ويضيف كل الأزرار والعناصر (بما فيها زر لوحة التحكم إذا كان المستخدم مسؤول).

        self.init\_main\_ui()

        self.stacked\_widget.setCurrentWidget(self.main\_page)

يغير الصفحة المعروضة حاليًا في الواجهة إلى الصفحة الجديدة.

        user\_type\_display = "مسؤول" if self.user\_type == "admin" else "طالب"

        self.setWindowTitle(f"نظام الإجابة الآلي لجامعة تبوك - {user\_type\_display}: {self.current\_user}")

        if self.user\_info\_label:

            self.user\_info\_label.setText(f"المستخدم: {self.current\_user} ({user\_type\_display})")

    def init\_main\_ui(self):

        main\_layout = QVBoxLayout(self.main\_page)

        main\_layout.setContentsMargins(20, 20, 20, 20)

        main\_layout.setSpacing(20)

        main\_layout.setAlignment(Qt.AlignCenter)

        # Header with logo and title

        header = QWidget()

        header.setLayoutDirection(Qt.RightToLeft)

        header\_layout = QHBoxLayout(header)

        header\_layout.setContentsMargins(0, 0, 0, 0)

        header\_layout.setAlignment(Qt.AlignCenter)

        # Logo

        logo\_label = QLabel()

        try:

            response = requests.get("https://iconape.com/wp-content/files/vf/193315/png/193315.png")

            logo\_pixmap = QPixmap()

            logo\_pixmap.loadFromData(response.content)

            logo\_pixmap = logo\_pixmap.scaled(70, 70, Qt.KeepAspectRatio, Qt.SmoothTransformation)

            logo\_label.setPixmap(logo\_pixmap)

        except:

            pass

        header\_layout.addWidget(logo\_label)

        # Title

        title = QLabel("Ubot")

        title.setFont(QFont('Arial', 30, QFont.Bold))

        title.setAlignment(Qt.AlignCenter)

        header\_layout.addWidget(title, alignment=Qt.AlignCenter)

        header\_layout.addStretch()

        # User info - store reference to update later

        user\_type\_display = "مسؤول" if self.user\_type == "admin" else "طالب"

        self.user\_info\_label = QLabel(f"المستخدم: {self.current\_user if self.current\_user else 'غير مسجل'} ({user\_type\_display})")

        self.user\_info\_label.setFont(QFont('Arial', 12))

        self.user\_info\_label.setAlignment(Qt.AlignCenter)

        header\_layout.addWidget(self.user\_info\_label)

        # Admin panel button (only for admin users)

        if self.user\_type == "admin":

            admin\_btn = QPushButton("لوحة التحكم")

            admin\_btn.setFont(QFont('Arial', 12))

            admin\_btn.clicked.connect(self.show\_admin\_panel)

            header\_layout.addWidget(admin\_btn)

        # Logout button

        logout\_btn = QPushButton("تسجيل الخروج")

        logout\_btn.setFont(QFont('Arial', 12))

        logout\_btn.clicked.connect(self.logout)

        header\_layout.addWidget(logout\_btn)

        main\_layout.addWidget(header)

        # Question input area

        question\_layout = QHBoxLayout()

        question\_layout.setAlignment(Qt.AlignCenter)

        question\_layout.setSpacing(15)

        # Chat area

        self.scroll\_area = QScrollArea()

        self.scroll\_area.setWidgetResizable(True)#  تعيين قابلية تغيير حجم المحتوى

        self.chat\_container = QWidget()

        self.chat\_layout = QVBoxLayout()

        self.chat\_layout.setAlignment(Qt.AlignTop)

        self.chat\_container.setLayout(self.chat\_layout)

        self.scroll\_area.setWidget(self.chat\_container)

        main\_layout.addWidget(self.scroll\_area)

        # Status bar

        self.status\_bar = QStatusBar()

        self.status\_bar.setLayoutDirection(Qt.RightToLeft)

        self.setStatusBar(self.status\_bar)

        self.status\_bar.showMessage("جاهز", 3000)

        # Lower layout for question input and ask button

        self.lower\_layout = QHBoxLayout() # تخطيط الجزء السفلي

        self.lower\_layout.setAlignment(Qt.AlignCenter)

        self.lower\_layout.setSpacing(2)

        # Ask button

        self.ask\_button = QPushButton("➤")

        self.ask\_button.setFont(QFont('Arial', 14))

        self.ask\_button.setFixedSize(55, 55)

        with open("style/AskButton.qss", "r", encoding="utf-8") as f:

            ask\_button\_style = f.read()

        self.ask\_button.setStyleSheet(ask\_button\_style)

        self.ask\_button.setLayoutDirection(Qt.RightToLeft)

        self.ask\_button.clicked.connect(self.ask\_question)

        self.lower\_layout.addWidget(self.ask\_button, alignment=Qt.AlignRight)

        # Question input field

        self.question\_input = QLineEdit()

        self.question\_input.setPlaceholderText("اكتب سؤالك هنا...")

        self.question\_input.setFont(QFont('Arial', 14))

        self.question\_input.setMinimumHeight(50)

        self.question\_input.setMinimumWidth(500)

        with open("style/QuestionInput.qss", "r", encoding="utf-8") as f:

            question\_input\_style = f.read()

        self.question\_input.setStyleSheet(question\_input\_style)

        self.question\_input.setLayoutDirection(Qt.LeftToRight)

        self.question\_input.returnPressed.connect(self.ask\_question)

        question\_layout.addWidget(self.question\_input)

        self.lower\_layout.addLayout(question\_layout)

        main\_layout.addLayout(self.lower\_layout)

    def show\_admin\_panel(self):

        if self.user\_type == "admin":

            admin\_panel = AdminPanel(self.user\_db, self.db\_manager, self.qa\_engine, self)

            admin\_panel.exec\_()

    def logout(self):

        self.current\_user = None

        self.user\_type = None

        self.stacked\_widget.setCurrentWidget(self.login\_page)

    def ask\_question(self):

        question = self.question\_input.text().strip()

        if not question:

            QMessageBox.information(self, "معلومة", "الرجاء إدخال سؤال")

            return

        try:

            self.status\_bar.showMessage("جارٍ معالجة السؤال...")

            QApplication.processEvents()  # Update UI immediately

            self.add\_chat\_message(question, sender='user') #عرض سؤال المستخدم

            answer = self.qa\_engine.ask\_question(question)

            self.add\_chat\_message(answer, sender='bot')  # عرض إجابة النظام

            ##self.answer\_display.setPlainText(answer)

            self.question\_input.clear()

            self.status\_bar.showMessage("تمت معالجة السؤال", 3000)

        except Exception as e:

            QMessageBox.critical(self, "خطأ", "حدث خطأ أثناء معالجة السؤال")

            self.status\_bar.showMessage("خطأ في معالجة السؤال", 3000)

    def add\_chat\_message(self,text,sender='user'):

            message = QLabel(text)

            message.setWordWrap(True)

            with open("style/Messageuser.qss", "r", encoding="utf-8") as f:

                messageuser\_style = f.read()

            with open("style/MessageBot.qss", "r", encoding="utf-8") as f:

                messageBot\_style = f.read()

            if sender == 'user':

                message.setStyleSheet(messageuser\_style)

                self.chat\_layout.addWidget(message, alignment=Qt.AlignRight)

            else:

                message.setStyleSheet( messageBot\_style)

                self.chat\_layout.addWidget(message, alignment=Qt.AlignLeft)

            message\_container = QWidget()

            layout = QHBoxLayout()

            layout.setContentsMargins(10, 5, 10, 5)

            if sender == 'user':

                layout.addStretch()

                layout.addWidget(message)

            else:

                layout.addWidget(message)

                layout.addStretch()

            message\_container.setLayout(layout)

            self.chat\_layout.addWidget(message\_container)

            QTimer.singleShot(100, lambda: self.scroll\_area.verticalScrollBar().setValue(self.scroll\_area.verticalScrollBar().maximum()))