المشكلة العامة : sentence classification

تصنيف النصوص هو واحد من المجالات الهامة في معالجة اللغة الطبيعية. تمت دراسة مشكلة التصنيف على نطاق واسع في استخراج البيانات ، التعلم الآلي ، وقاعدة البيانات ، و مجال استرجاع المعلومات مع التطبيقات في عدد من المجالات المتنوعة ، مثل التسويق المستهدف ، التشخيص الطبي ، تصفية مجموعة الأخبار ، تحديد موضوع مقالة إخبارية ، تحليل المشاعر. والوكلاء التخاطبية . ومن المعروف أنه من المستحيل تعريف أفضل مصنف نصي. في مجالات مثل الرؤية الحاسوبية Computer Vision، هناك إجماع قوي حول طريقة عامة لتصميم النماذج والشبكات العصبونية وغيرها من المنهجيات المعتمدة . و بخلاف ذلك ، لا يزال تصنيف النص يفتقر إلى هذه الطريقة العامة في مجالات كثيرة .

المشكلة المطروحة : تصنيف بيانات جمل إلى أكثر من class .

البيانات dataset :

- مجموعة من جمل على شكل أسئلة تطرح على قسم الموارد البشرية في المنظمات قمنا بتجميعها من المواقع الإلكترونية لهذه المنظمات من قسم الموارد البشرية المخصص على هذه المواقع . ليتم استخدامه لاحقا في نظام chatbot قادر على تحديد الغاية من جملة المستخدم ليتم الاستجابة لها بالطريقة المناسبة .
- تم تجميع 8124 سطر من البيانات (جمل).
- قمنا بتصنيف هذه الجمل إلى 7 classes تعبر عن الغاية من السؤال المطروح على قسم الموارد البشرية وهي كالتالي

1. **Greeting** : ويكود القصد من الجملة هو لمجرد الترحيب بالمستخدم .
   مثال :" Hi there"
2. **inqueryService** : يعبر عن ان المستخد يرغب بالاستفسار بشكل عام عن خدمة .
   مثال : " What is the difference between being paid stipend versus salary"
3. **dateService** : يعبر عن ان المستخد يرغب بمعرفة وقت او زمن بخصوص خدمة.
   مثال : "after I sign up, how long will it take to receive my member card?"
4. **beQuestion** : يعبر عن ان المستخدم يسأل عن خدمة جوابها yes or no .
   مثال : "I'm not an employee of the University can I still access the services"
5. **placeService** : يعبر عن ان المستخد يرغب بمعرفة مكان يتعلق بحصوله عن خدمة.
   مثال : "I'm interested in a Faculty position, where can I find those job postings?
6. **getService** : يعبر عن أن المستخدم يرغب بمعرفة تفاصيل الحصول على خدمة .
   مثال : " I don't have a resume. How can I apply?"

7. **QuantityService** : يعبر عن أن المستخدم يستفسر عن كمية من عددية حول خدمة .

مثال : " How many free staff tickets do I get? "

يظهر الجدول التالي حجم كل class(intent) في البيانات :

Greeting	inqueryService	dateService	beQuestion	placeService	getService	QuantityService
63	3252	628	2881	303	720	277

State of the art :

في السنوات الأخيرة ، أدى التقدم في تقنيات الشبكات الاجتماعية إلى اهتمام كبير بتصنيف المستندات النصية والجمل التي تحتوي على روابط أو معلومات تعريفية أخرى . تم القيام بالعديد من الدراسات حول خوارزميات التصنيف من خلال العديد من الأبحاث. ومن هذه الخوارزميات Decision Trees و Support Vector Machine و Naïve Bayes و Hidden Markov model و Neural Network Classifiers و غيرها .و منها من أثبت كفائته في تصنيف النصوص .

❖ **Baseline** :

يُعرف مصنف **Naïve Bayes** كمجموعة من المصنّفات الاحتمالية البسيطة القائم على فرضية عامة مفادها أن جميع السمات مستقلة عن بعضها البعض ، وفقًا للصنف المحدد . ولسهولة تطبيق هذا المصنف وسرعته فهو يعتبر خط الأساس في تصنيف النصوص. إلا انه بتطور البحث العملي في مجال تصنيف النصوص أصبحت SVM هي خط الأساس لفاعليتها في مجال تصنيف النصوص إلى أن هذه الخوارزمية تعاني من مشكلة البطء فتعقيدها الزمني كبير اضافة إلى مسألة استخراج السمات التي تلعب دورا هامة في دقة المصنف والتي تعد الخطوة الأصعب في مسألة التصنيف. تقوم خوارزمية **SVM** بتمثيل النص كمتجه حيث يمثل بعده عدد الكلمات الأساسية المختلفة (keywords) إذا كان حجم النص كبيرًا ، فإن الأبعاد كبيرة جدًا بالنسبة لفضاء الأبعاد الخاص بعملية التصنيف و الذي يسبب تكلفة حسابية عالية. ولكن يمكن تخفيض فضاء الأبعاد عن طرق استخراج وتقليل السمات.

❖ **نوع خوارزمية التصنيف المستخدمة** :

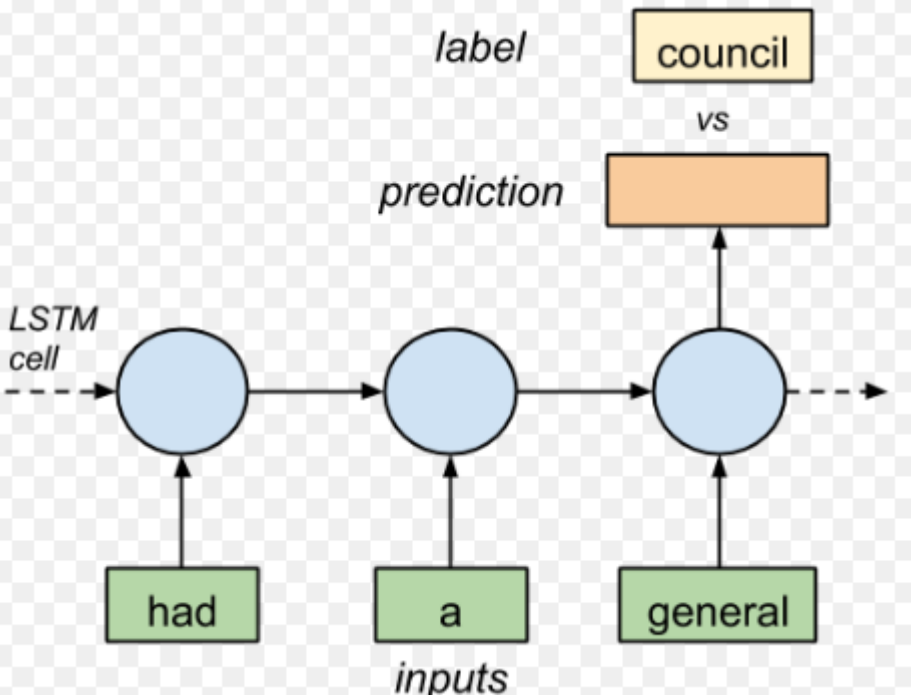
إن النموذج المستخدم هو NEURAL NETWORK حيث ان مثل هذه التقنيات لا تقدم أي افتراضات حول البيانات فهي تأخذ النص على اعتبار أنه سلسلة من الكلمات. في جميع الاساليب السابقة التي تطرقنا

لها كان الاعتماد على استخلاص السمات يدويا – والذي يشكل المشكلة الأكبر في مسألة تصنيف النصوص – يتم عن طريق نماذج مثل حقيبة الكلمات BOW و Unigrams و Bigrams ونماذج أخرى إلا إن هذه النماذج تتجاهل سياق النص و ترتيب الكلمات في النص الامر الذي يبقي هذه النماذج غير مرضية لإلتقاط دلالات النص . اما في مصنفات الشبكات العصبية يؤخذ تسلسل الكلمات بعين الاعتبار والتي قد يكون لها دلالات تؤثر على معنى النص بالاستفادة من تضمين الكلمات والذي يكون بمثابة نموذج مدرب مسبقاً word embedding والذي لديه القدرة على استخراج التمثيل النحوي والدلالي المفيد في عملية التصنيف . قد تكون لهذه المصنفات مشاكل قليلة مثل تفاوت البيانات التي قد يكون لها تاثير ع دقة المصنف إضافة إلى توفر حجم كبير بيانات التدريب.

أشهر الشبكات العصبونية المستخدمة في تصنيف النصوص هي الشبكات العصبية المتكررة Recurrent NN والشبكات العصبية التلافيفية Convolution NN .

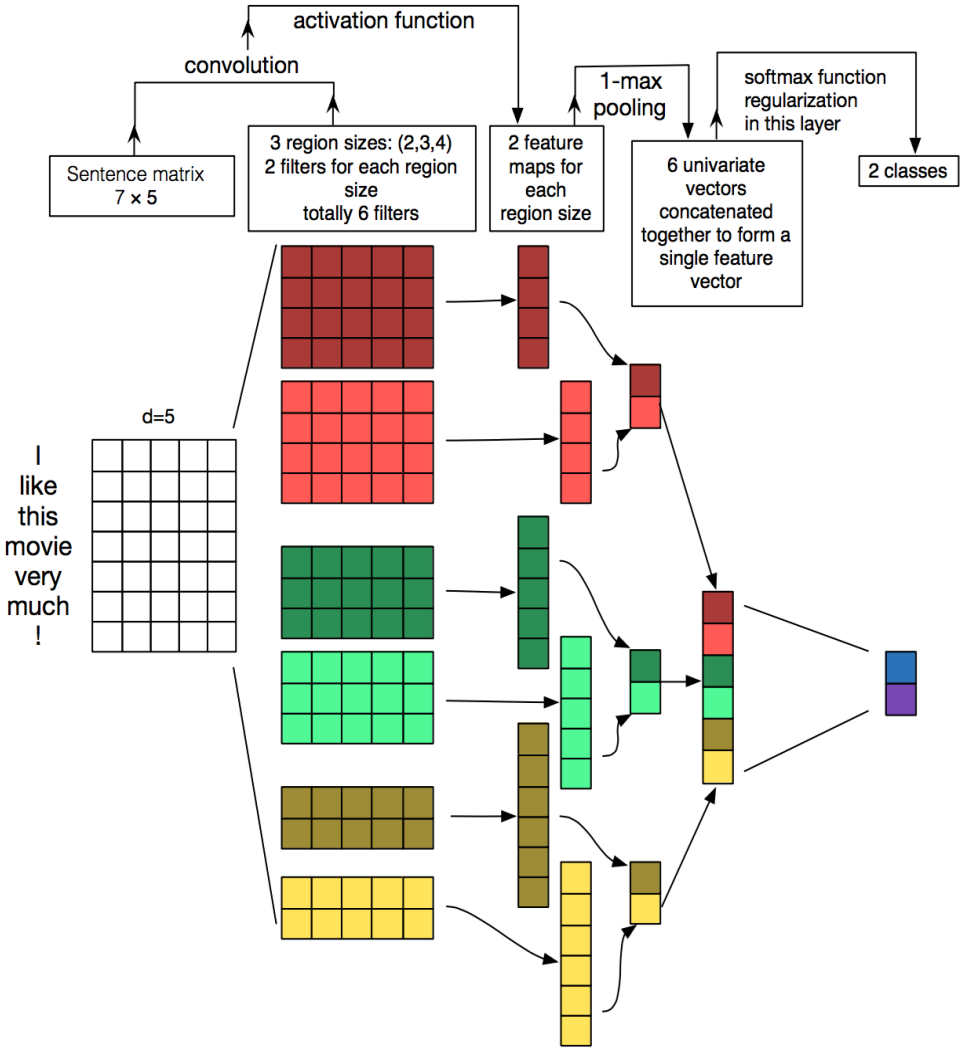
- **الشبكات العصبية المتكررة RNN** :

التي أثبتت فعاليتها حيث لديها القدرة على استخراج الدلالة من الجملة باستخدام بنية شجرية. يعتمد أدائها بشكل كبير على تطور الشجرة النصية. حيث تقوم بفحص كلمات النص كلمة كلمة وتتحفظ بالدلالات للنص جميعها في طبقة مخفية الأمر الذي يجعل تعقيدها الزمني كبير بأحسن الأحوال (On2) حيث n هو طول النص. إذا كانت الجملة أو النص طويلة جداً ، فإن هذه التقنية تستغرق وقتا طويلا جداً. بالإضافة إلى ذلك ، قد يكون من الصعب جدًا تطوير علاقة بين جملتين بواسطة بنية شجرية. وبالتالي ، NN Recursive غير مناسبة لصياغة جمل طويلة أو نص طويل. ميزة RNN هي قدرتها على التقاط المعلومات السياقية بشكل أفضل. ومع ذلك ، فإن RNN هو نموذج متحيز ، حيث تكون الكلمات اللاحقة أكثر هيمنة من الكلمات السابقة. وبالتالي ، يمكن أن تقلل الفعالية عندما يتم استخدامها لالتقاط الدلالات في نص كامل ، لأن المكونات الأساسية يمكن أن تظهر في أي مكان في النص وليس في النهاية وجميع الكلمات لها نفس الاحتمالية لظهورها في تسلسل الكلمات. أشهرها هي LSTM الشكل (1) و BiLstm الشكل (2)



الشكل (1)

- **الشبكات العصبية التلافيفية (Convolution NN)** : تم تصميمها لتأخذ بعين الاعتبار البنية المكانية في بيانات الصورة الي الذي يقدم معلومات مهمة عن توضع واتجاه الغراض في الصورة .حيث يمكن استخدامه لفكرة التسلسل مثل تسلسل الكلمات أحاية البعد . الطريقة نفسها التي تجعل نموذج CNN أكثر فائدة لتعلم التعرف على الأغراض في الصور يمكن أن يساعد على تعلم نمط في جمل من الكلمات ، لهذا النموذج القدرة على استخراج الدلالة من النصوص بطريقة منهجية أكثر بالمقارنة مع الشبكات العصبية المتكررة مع تعقيد زمني O(n) حيث وجدت لتحل مشكلة التحيز الموجودة في شبكات ال RNN . وذلك باستخدام طبقة max-pooling .على الرغم من أن الأبحاث السابقة على CNN أظهرت أن تقنية النواة Kernal تعتبر غير مرنة إضافة إلا أنه من الصعب جداً العثور على حجم Kernal مناسب ; فلو كان حجم Kernal كبيراً فسيكون سبب في زيادة عدد المعاملات وبالتالي يصبح من الصعب تدريبها في حين لو كان حجم Kernal صغير قد يكون لذلك تأثير على دقة التصنيف نتيجة فقدانه لمعلومات مهمة .الشكل (3)

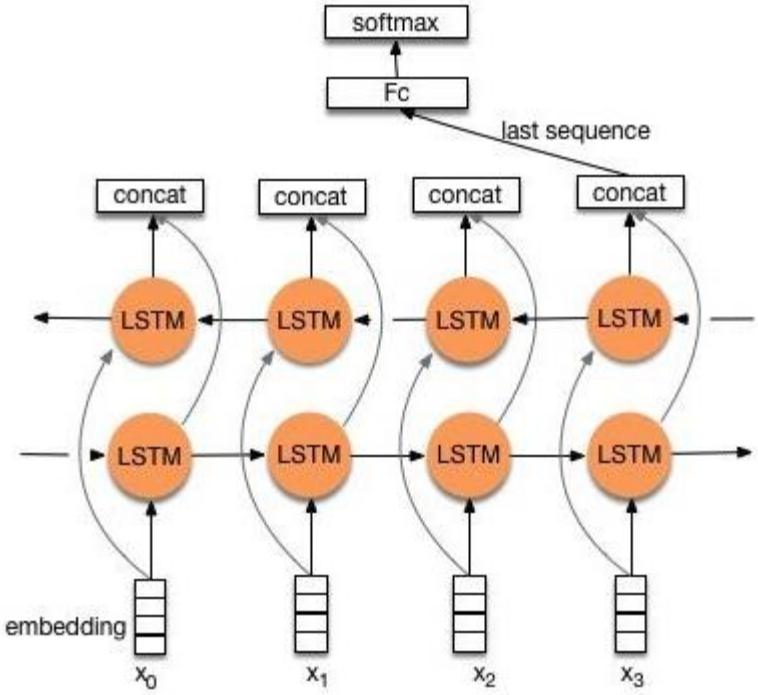


الشكل (3)

- **Bidirectional LSTM :**

الشبكة العصبية ثنائية الاتجاه وهي شكل من أشكال الشبكات العصبية المتكررة يمكنها تحسين الأداء مقارنة بال LSTM التقليدية في مسألة تصنيف النصوص . في المشاكل التي تتوفر فيها جميع الخطوات الزمنية لتسلسل الدخل ، تقوم LSTM ثنائية الاتجاه بتدريب اثنين من LSTMs بدلاً من واحدة على تسلسل الدخل ، الشكل (4) . الأول على تسلسل الدخل كما هو ، والثاني على نسخة معكوسة من تسلسل الدخل . يمكن أن يوفر ذلك سياقًا إضافيًا للشبكة ويؤدي إلى تعلم أسرع وحلاً أفضل للمشكلة. إن فكرة هذا النموذج تنطوي على تكرار أول طبقة متكررة في الشبكة بحيث يكون هناك الآن طبقتان جنبًا إلى

جنب ، ثم توفير تسلسل الدخل كدخل للطبقة الأولى و نسخة معكوسة من تسلسل الدخل إلى الثانية . يمكن لهذه البنية أن تضيف ما هو أكثر من مجرد عكس الدخل فيما يتعلق بالسياق العام للنص . وهذه البنية تعمل بشكل أفضل في تصنيف الجمل منه من ال documents .



الشكل(2)

-

❖ **Word embedimgs [1]:**

هي نوع من تمثيل الكلمات التي تسمح للكلمات ذات المعنى المتشابه أن يكون لها تمثيل متشابه. يتم تمثيل الكلمات كل كلمة على حدى كأشعة ذات قيمة حقيقية في فضاء أشعة محدد مسبقًا. يتم تمثيل كل كلمة إلى بشعاع واحد ويتم تعلم قيم الشعاع بطريقة تشبه الشبكة العصبية . قمنا باستخدام تمثيل Golve المدرب على بليون tokens من ويكيبيديا .توضع الكلمات غير الموجودة في مجموعة الكلمات المدربة مسبقًا هذه من خلال أخذ عينات عشوائية من التوزيع المنتظم ضمن المجال [−0.1,0.1] .يتم تعديل القيم من هذا التمثيل أثناء التدريب لتحسين الأداء . وتتمثل الفائدة الأساسية لهذا التمثيل في إمكانية تعلم عالية (مساحة وزمن أقل) ، الامر الذي يسمح لembeddings أكثر (بأبعاد كبيرة) من التعلم من مجموعة نصية أكبر بكثير (مليارات الكلمات).

لعملية تصنيف البيانات تم اشتخدام أنوع عامة من المصنفات مشهورة في مسألة ال text classification وكنا قد تحدثنا عنها في ال state of the art وعن بنية كل منها وهي كالتالي :

1. Naïve Bayes
2. SVM
3. CNN
4. Bidierctional NN(BiLSTM)
5. CNN-LSTM

وتمت مقارنة النتائج في النهاية بين أداء كل من هذه المصنفات على البيانات الخاصة بنا .

أولا : Naïve Bayes classifier :

خطوات العمل وبناء النموذج والتي تمت بمساعدة مكتبة Sicikt-learn :

1) **preprocessing** :

- تقسيم ال **dataset** : تم تقسيم البيانات إلى 0.67% للتدريب و 0.33 % للاختبار .
- إزالة الاختصارات من الداتا مثل (NYU , SMMTY,.....etc) .
- عملية تقطيع لكلمات الجمل tokenizing .
- تحويل الكلمات في الجمل إلى lower case .
- اعادة الكلمات إلى جذوعها (stemming)
- عملية إزالة ال stopwords .

2) **Feature extraction** : تم الاعتماد على نوعين من السمات وهي :

- bag of words :وتمثل مجموعة الكلمات الأساسية في الداتا .
- TF-IDF : (Term Frequency times inverse document frequency): وتمثل عدد مرات تكرار كل مفردة في الجمل مع امكانية تقليل ظهور الكلمات الشائعة مث (etc... ,a, the).

3) **Fitting** :

تم تدريب المصنف في حالتين :

الاولى : التدرب على الكلمات ولكن بدون القيام بإزالة ال stopwords و القيام ب stemming .

الثانية .التدرب على الكلمات مع وجود ال stopwords و stemming .

وتم حساب الدقة والخطأ في كل من الحالتين , الشكل (4)

```
]: bayes_clf.fit(X_train, y_train)
predicted = bayes_clf.predict(X_test)
print('Naive Bayes  %f' %(np.mean(predicted == y_test)))
print('Naive Bayes error : {:4.2f} '.format(1 - metrics.accuracy_score(y_test, predicted)))

bayes_clf_stop_stem.fit(X_train, y_train)
predicted_stop_stem = bayes_clf_stop_stem.predict(X_test)
print('Naive Bayes with remove stopwords and stemming %f' %(np.mean(predicted_stop_stem == y_test)))
print('Naive Bayes error with stem and stop : {:4.2f} '.format(1 - metrics.accuracy_score(y_test, predicted_stop_stem)))
```

```
Naive Bayes  0.665423
Naive Bayes error : 0.33
Naive Bayes with remove stopwords and stemming 0.535621
Naive Bayes error with stem and stop : 0.46
```

الشكل(4)

4) **Model selection** :

باستخدام ال grid search للحصول على افضل model يعطي أفضل دقة على بيانات الاختبار بأفضل المعاملات.

المعاملات المستخدمة في عملية البحث هي :

I. vect__ngram_range': [(1, 1), (1, 2')] : ويقصد بها استخدام n–gram و bi–gram .

II. tfidf__use_idf': (True, False) : وتعني استخدام tf–idf بدون أو مع ال idf .

Grid search

```
: parameters_NB = {'vect__ngram_range': [(1, 1), (1, 2)], 'tfidf__use_idf': (True, False)}
gs_clf_NB = GridSearchCV(bayes_clf, parameters_NB, n_jobs=-1)
gs_clf_NB = gs_clf_NB.fit(X_train, y_train)
print(gs_clf_NB.best_score_)
print(gs_clf_NB.best_params_ )
```

```
0.6880396839977954
{'tfidf__use_idf': False, 'vect__ngram_range': (1, 2)}
```

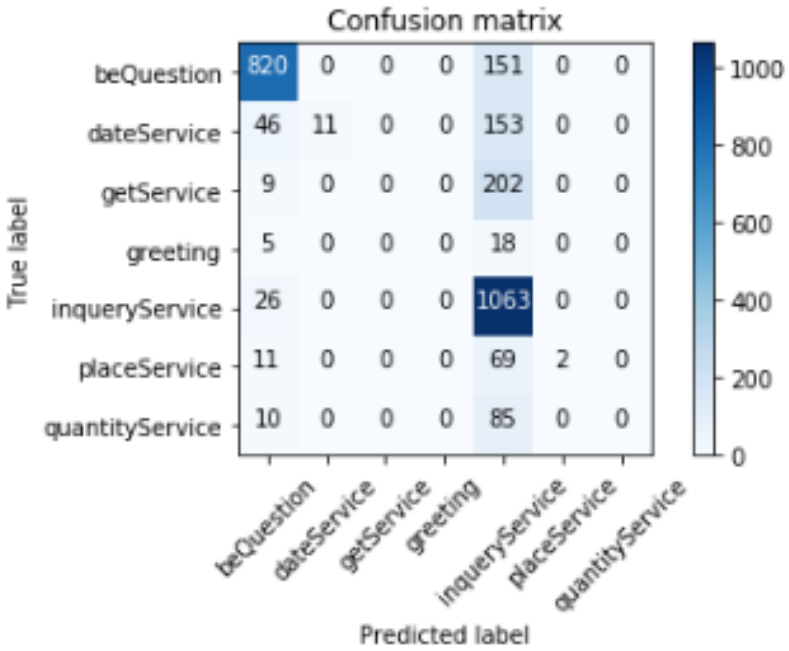
باستخدام المعاملات التي تعطي أفضل دقة تم تدريب النموذج فتم الحصول على دقة : 70.8% , وخطأ : 20%. كما هو موضح بالشكل (5)

```
from sklearn.linear_model import SGDClassifier
bayes_clf = Pipeline([('vect_', CountVectorizer(ngram_range=(1,2))),
    ('tfidf', TfidfTransformer(use_idf=False)),
    ('mnb', MultinomialNB(fit_prior=True))

])

bayes_clf.fit(X_train, y_train)
predicted = bayes_clf.predict(X_test)
print('Naive Bayes  %f' %(np.mean(predicted == y_test)))
print('Naive Bayes error : {:4.2f} '.format(1 - metrics.accuracy_score(y_test, predicted)))
```

```
Naive Bayes  0.707199
Naive Bayes error : 0.29
```

5) **Evaluation Model** :

باستخدام ال confusion matrix في الشكل (6)



الشكل(6)

SVM :

خطوات العمل وبناء النموذج :

1) **preprocessing** و ال feature selection هي ذاتها الخطوات في مصنف Naïve Bayes.
2) **Fitting** : ايضاً تم التدريب في حالتين (مع وبدون remove stopwords, stemming). وكانت النتائج كما في الشكل التالي:

```
svm = svm_clf.fit(X_train, y_train)
predicted = svm_clf.predict(X_test)
print('SVM correct prediction: {:4.2f}'.format(np.mean(predicted == y_test)))



#with remove stopword and stemming
svm_stem_stop = svm_clf_stop_stem.fit(X_train, y_train)
predicted_ss = svm_stem_stop.predict(X_test)
print('SVM correct prediction with (stemming ,stopwords) : {:4.2f}'.format(np.mean(predicted_ss == y_test)))
```

```
SVM correct prediction: 0.86
SVM correct prediction with (stemming ,stopwords) : 0.57
```

3) **Model selection** :

تم اختيار نفس المعاملات التي تم استخدامها في مصنف Naïve Bayes باستخدتم grid search وكانت النتائج كما في الشكل (7):

Grid search

```
parameters_svm = {'vect__ngram_range': [(1, 1), (1, 2)], 'tfidf__use_idf': (True, False), 'clf__alpha': (1e-2, 1e-3)}
gs_clf_svm = GridSearchCV(svm_clf, parameters_svm, n_jobs=-1)
gs_clf_svm = gs_clf_svm.fit(X_train, y_train)
print(gs_clf_svm.best_score_)
print(gs_clf_svm.best_params_ )
```

```
0.8623920632004409
{'clf__alpha': 0.001, 'vect__ngram_range': (1, 2), 'tfidf__use_idf': False}
```

الشكل(7)

باستخدام المعاملات التي تعطي أفضل دقة تم تدريب النموذج تم الحصول على دقة : 87% , وخطأ : 13%. كما هو موضح بالشكل التالي:

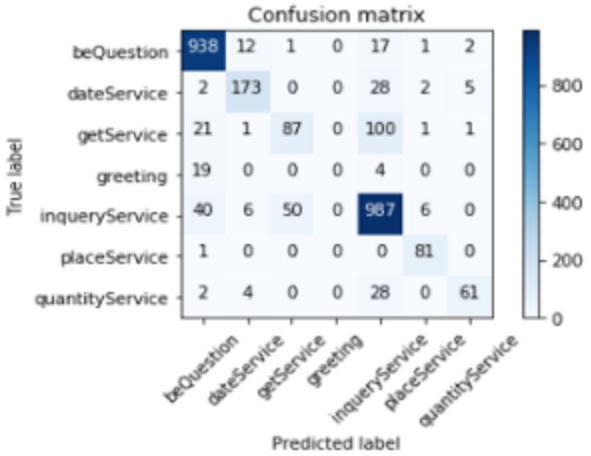
```
svm_clf = Pipeline([('vect_', CountVectorizer(ngram_range=(1,2))),
    ('tfidf', TfidfTransformer(use_idf=False)),
    ('clf', SGDClassifier(loss='hinge', penalty='l2', alpha=1e-3, max_iter=20, random_state=random_state,shuffle=True)),
])

svm = svm_clf.fit(X_train, y_train)
predicted = svm_clf.predict(X_test)
print('SVM accuracy: {:4.2f}'.format(np.mean(predicted == y_test)))
# error
print('SVM error : {:4.2f} '.format(1 - metrics.accuracy_score(y_test, predicted)))
```

```
SVM accuracy: 0.87
SVM error : 0.13
```

4) Evaluation Model :

باستخدام ال confusion matrix وال classification_report في الشكل (8):

	precision	recall	f1-score	support
beQuestion	0.92	0.97	0.94	971
dateService	0.88	0.82	0.85	210
getService	0.63	0.41	0.50	211
greeting	0.00	0.00	0.00	23
inqueryService	0.85	0.91	0.88	1089
placeService	0.89	0.99	0.94	82
quantityService	0.88	0.64	0.74	95
avg / total	0.85	0.87	0.86	2681



الشكل(8)

ثالثاً : Neural Network :

بداية وقبل بناء نموذج التدريب الخاص بال Neural Network قمنا بما يلي :

a. تقسيم ال **dataset** : تم تقسيم البيانات إلى 0.67% للتدريب و 0.33 % للاختبار .

b. **preprocessing** : القيام بعمليات معالجة مسبقة حيث أن ال neural network تتعامل من قيم حقيقية للدخل ويتطلب ذلك ما يلي :

- tokenizing : تم تقطيع جميع الجمل المراد تدريبها إلى كلمات .
- تمثيل كل كلمة شعاع.

- Padding :ويقصد به توحيد أبعاد أشعة الكلمات ,وفقا لاطول جملة في البيانات من خلال إضافة أصفار إلى نهاية الشعاع.
- عملية تحويل الكلمات إلى حالة lower case وازالة الاختصارت مثل (NYU, DAMAS,..etc).
- تمثيل كل class بشعاع binary بعده 7 (عدد ال classes) بالشكل التالي one hot encoded :[0,1,0,0,0,0,0] حيث يأخذ القيمة 1 مقابل ال class المراد تمثيله وذلك وفقا لترتيب قاموس ال classes .
- Embedding matrix : حيث نحتاج إلى إنشاء مصفوفة واحدة لل embedding لكل كلمة في بيانات التدريب. يمكننا القيام بذلك عن طريق تعداد كل الكلمات الفريدة في الداتا عن طريق تابع () Tokenizer.word_index الموجود في keras والقيام بعملية توزين لهذه الكلمات من GloVe embedding الذي تم قرائته .

c. بناء النموذج:

- تم الاستعانة بمكتبة keras المتخصصة في بناء نماذج deep learning لبناء النموذج الخاص بنا .
- توفر keras طبقة ال embedding سهولة في استخدام نماذج تمثيل للكلمات مثل golve بحيث تكون كل كلمة هي one-hot encoded. يتم تحديد حجم فضاء الشعاع الممثل لكل كلمة كجزء من النموذج ، بأبعاد مثل 50 أو 100 أو 300. يتم تهيئة الأشعة بأعداد عشوائية صغيرة. يتم استخدام طبقة التضمين في الواجهة الأمامية للشبكة العصبية وتناسبها بطريقة supervisor باستخدام خوارزمية **Backpropagation**.
- يكون لطبقة ال **embedding** الشكل التالي :

Embedding(max_no_un_word, embedding dim, weights=[embedding_matrix], max_length)

حيث :

- max_no_un_word, : وتمثل عدد الكلمات المميزة في بيانات التدريب.
- embedding dim : بعد نموذج التضمين المختار من تمثيل glove .
- weights : مصفوفة اوزان كلمات الدخل للشبكة والتي تم تهيئتها بأشعة لكل كلمة من glove .
- max_length : عدد كلمات اطول جملة في بيانات التدريب.

تم استخدام اكثر من نوع من الشبكات العصبونية (CNN , BLSTM , C-LSTM).

CNN : تم بناء النموذج وفقا لورقة البحث [2] الي قام بها Chunting وآخرون وفقا للشكل (3) :

عند اظهار الطبقات التي ستتعلم سيكون تسلسل الطبقات بالشكل (9) :

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 81, 100)           465500
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 79, 250)           75250
_________________________________________________________________
global_max_pooling1d_3 (Glob (None, 250)               0
_________________________________________________________________
dense_5 (Dense)              (None, 120)               30120
_________________________________________________________________
dense_6 (Dense)              (None, 7)                 847
=================================================================
Total params: 571,717
Trainable params: 106,217
Non-trainable params: 465,500
_________________________________________________________________
None
Train on 5444 samples, validate on 2680 samples
Epoch 1/12
5444/5444 [==============================] - 15s 3ms/step - loss: 0.7731 - acc: 0.7384 - val_loss: 0.4638 - val_acc: 0.8362
Epoch 2/12
5444/5444 [==============================] - 14s 3ms/step - loss: 0.3661 - acc: 0.8749 - val_loss: 0.4965 - val_acc: 0.8157
```

الشكل(9)

وفقا لل hyperprameters التالية كما هي موضحة بالجدول(1) :

حيث :

- neurons_hidden : تمثل عدد خلايا الطبقة المخفية dense() مع تابع التنشيط ‘relu”
- nb_filters :وتمثل عدد الفلاتر المستخدمة في الشبكة في طبقة ال convolution .
- 13ernel_sz : وتمثل حجم ال kernel المستخدم .
- Dropout_rate : تمثل قيمة L2 regulaization
- Epochs : تمثل عدد عصور التدريب.
- Batch_size : والذي يمثل عدد ال iteration في كل epoch (عدد امثلة التدريب في كل عصر من عصور التدريب).

hyperparameter	Hyperprameter value
neurons_hidden	120
nb_filters	250
13ernel_sz	3
Batch_size	20
Dropout_rate	None
Epochs	12

الجدول(1)

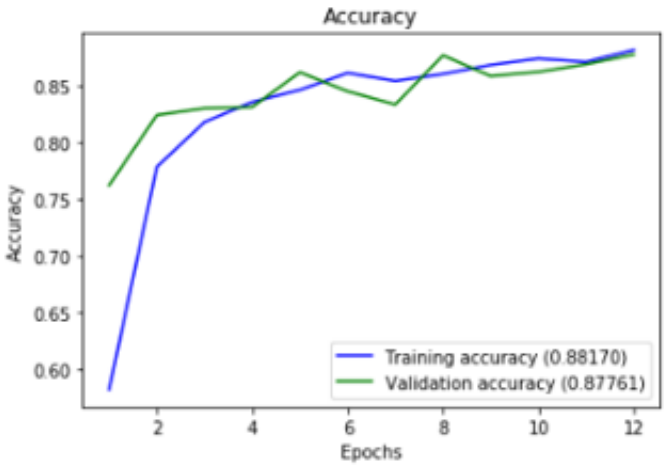
تظهر نتائج التدريب مايلي :

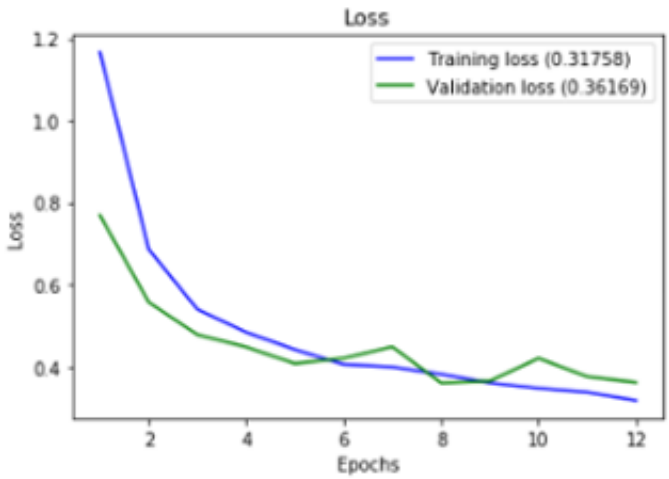
دقة : و 86.08 % ونسبة خطا :0.52% .

يظهر الرسم البياني حالة overfitting . لذلك مع استخدام نفس hyperprarmeters القديمة مع اضافة قيمة Dropout_rate(0.5) :

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 81, 100)           465500
_________________________________________________________________
dropout_4 (Dropout)          (None, 81, 100)           0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 79, 250)           75250
_________________________________________________________________
global_max_pooling1d_4 (Glob (None, 250)               0
_________________________________________________________________
dense_7 (Dense)              (None, 120)               30120
_________________________________________________________________
dropout_5 (Dropout)          (None, 120)               0
_________________________________________________________________
dense_8 (Dense)              (None, 7)                 847
=================================================================
Total params: 571,717
Trainable params: 106,217
Non-trainable params: 465,500
_________________________________________________________________
None
Train on 5444 samples, validate on 2680 samples
Epoch 1/12
5444/5444 [==============================] - 18s 3ms/step - loss: 1.1670 - acc: 0.5825 - val_loss: 0.7692 - val_acc: 0.7623
Epoch 2/12
5444/5444 [==============================] - 17s 3ms/step - loss: 0.6860 - acc: 0.7787 - val_loss: 0.5572 - val_acc: 0.8243
```

يظهر ذلك تحسن أفضل عن النموذج السابق بدقة 87.7% وخطأ 0.36%.





مع تعديل علىhyperprarmeters مع Dropout_rate1 قبل طبقة ال convolution و Dropout_rate2 بعد الطبقة المخفية Dense كما في الشكل () , كما في الجدول (2) :

hyperparameter	Hyperprameter value
neurons_hidden	80
nb_filters	250
kernal_sz	3
Batch_size	20
Dropout_rate1	0.3
Dropout_rate2	0.5
Epochs	12

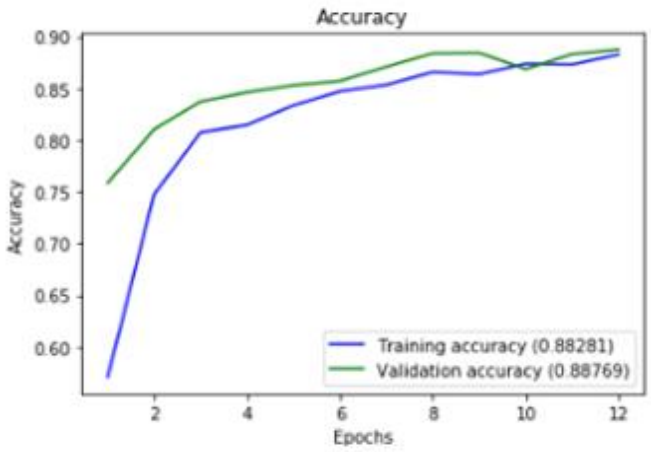
الجدول(2)

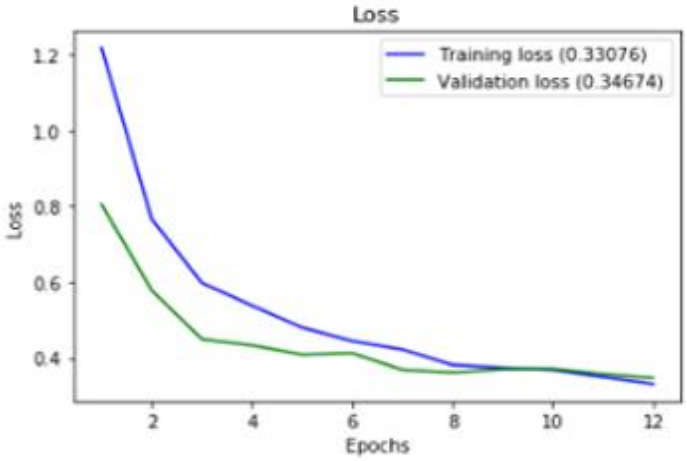
```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 81, 100)           465500
_________________________________________________________________
dropout_22 (Dropout)         (None, 81, 100)           0
_________________________________________________________________
conv1d_13 (Conv1D)           (None, 79, 250)           75250
_________________________________________________________________
global_max_pooling1d_3 (Glob (None, 250)               0
_________________________________________________________________
dense_19 (Dense)             (None, 80)                20080
_________________________________________________________________
dropout_23 (Dropout)         (None, 80)                0
_________________________________________________________________
activation_2 (Activation)    (None, 80)                0
_________________________________________________________________
dense_20 (Dense)             (None, 7)                 567
=================================================================
Total params: 561,397
Trainable params: 95,897
Non-trainable params: 465,500
```

الشكل()

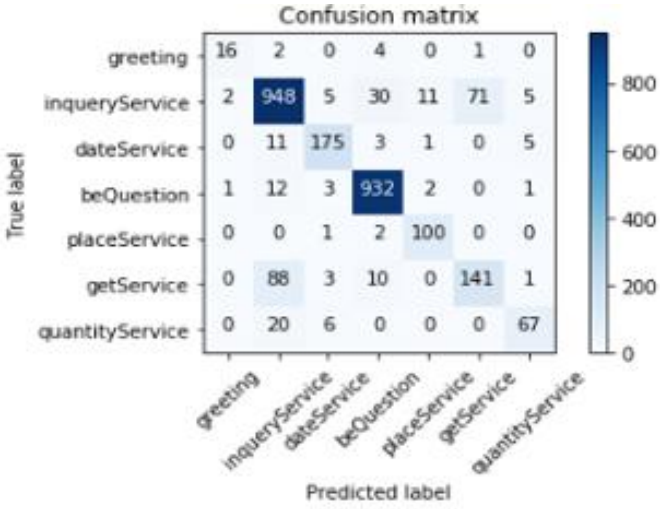
حصلنا على النتائج التالية :

دقة : 88.7% , وخطأ: 0.34 .






```
Accuracy score: 0.8876865671641792

Classification Report
              precision    recall  f1-score   support

           0       0.842     0.696     0.762        23
           1       0.877     0.884     0.881      1072
           2       0.907     0.897     0.902       195
           3       0.950     0.980     0.965       951
           4       0.877     0.971     0.922       103
           5       0.662     0.580     0.618       243
           6       0.848     0.720     0.779        93

avg / total       0.884     0.888     0.885      2680
```



- **Bidirectional LSTM NN** :

وهو نموذج بسيط يستخدم احد انواع شبكات RNN وهو عبارة عن تركيب خلايا LSTM forword و LSTM backword ويظهر هذا النموذج أداءا فعالا مع الجمل مقارنة بال documents : يمثل

الشكل تسلسل الطبقات في الشبكة المستخدمة للتدريب الشكل(10) مع ال hyperparameter المبينة في الجدول (3) :

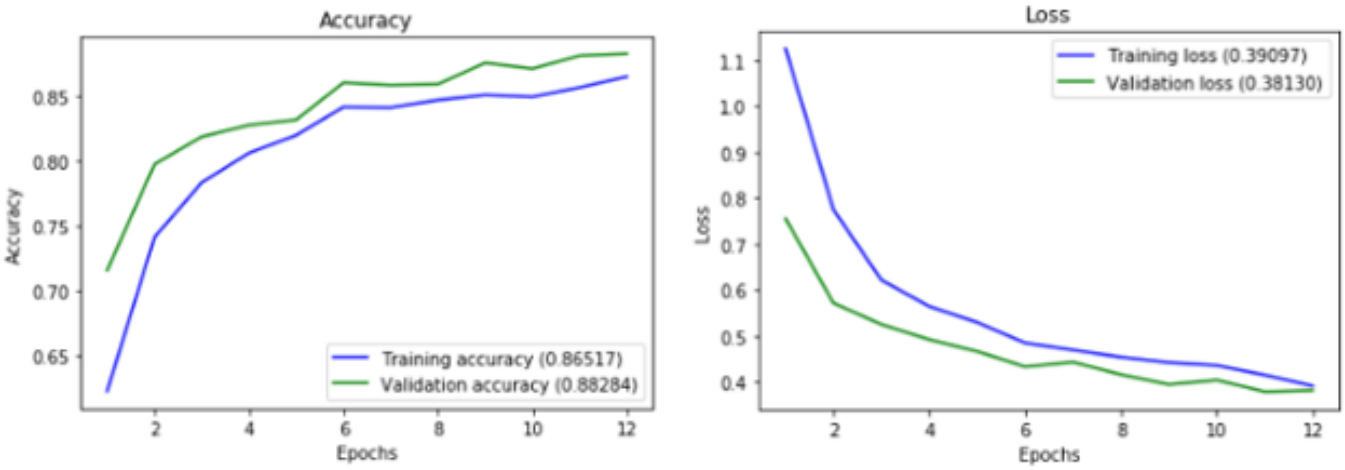
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 81, 100)           465500
_________________________________________________________________
dropout_1 (Dropout)          (None, 81, 100)           0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 81, 120)           77280
_________________________________________________________________
bidirectional_2 (Bidirection (None, 120)               86880
_________________________________________________________________
dropout_2 (Dropout)          (None, 120)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 847
=================================================================
Total params: 630,507
Trainable params: 165,007
Non-trainable params: 465,500
_________________________________________________________________
None
Train on 5444 samples, validate on 2680 samples
Epoch 1/12
5444/5444 [==============================] - 91s 17ms/step - loss: 1.1257 - acc: 0.6220 - val_loss: 0.7555 - val_acc: 0.7153
Epoch 2/12
5444/5444 [==============================] - 97s 18ms/step - loss: 0.7752 - acc: 0.7410 - val_loss: 0.5714 - val_acc: 0.7974
```

الشكل(10)

hyperparameter	Hyperprameter value
neurons_hidden	60
Batch_size	20
Epochs	12
Dropout_rate	None

الجدول(3)

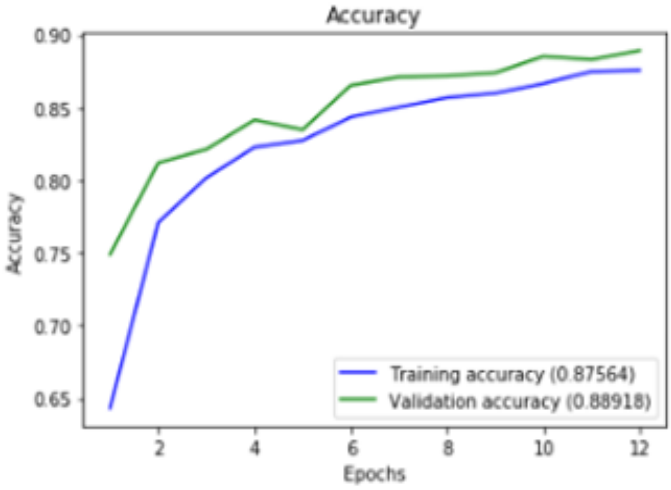
يظهر أداء النموذج حالة overfitting لحل هذه المشكلة نضيف Dropout بقيمة 0.6 فنجد ان أداء النموذج في تحسن (دقة 88.2% و خطأ 0.38) كما في الشكل التالي :

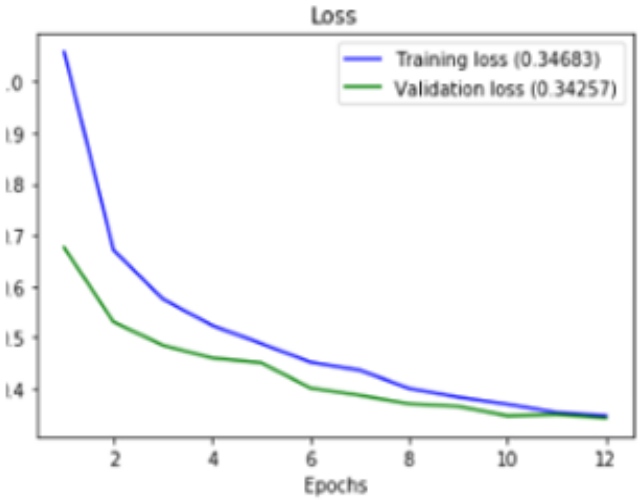


بعمل tunning لل hyperparameter كما في الجدول (4) : حصلنا على النتائج (دقة : 88.9% وخطأ 0.34)

hyperparameter	Hyperprameter value
neurons_hidden	100
Batch_size	20
Epochs	12
Dropout_rate	0.6

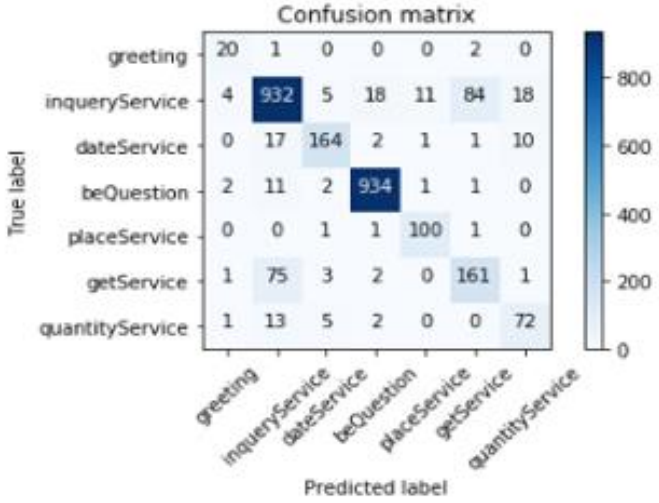
الجدول(4)





تقييم الاداء من خلال confusion matrix :

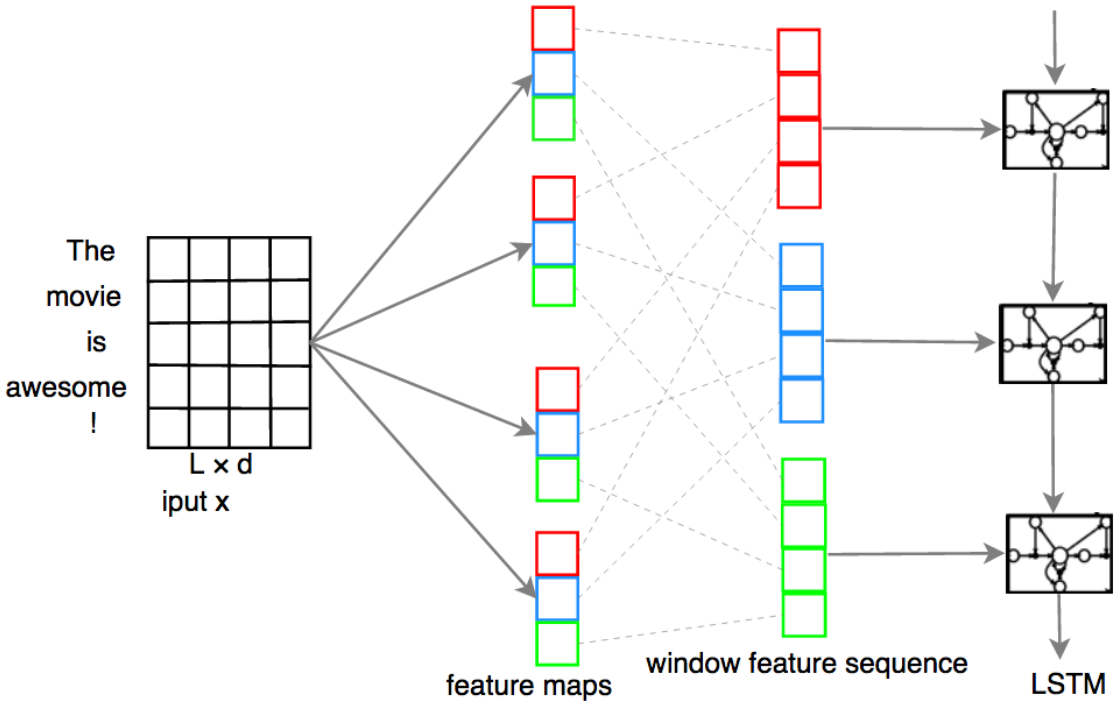
```
Accuracy score: 0.889179104477612

Classification Report
              precision    recall  f1-score   support

           0       0.714     0.870     0.784        23
           1       0.888     0.869     0.879      1072
           2       0.911     0.841     0.875       195
           3       0.974     0.982     0.978       951
           4       0.885     0.971     0.926       103
           5       0.644     0.663     0.653       243
           6       0.713     0.774     0.742        93

avg / total       0.891     0.889     0.890      2680
```



- **CNN–LSTM(C–LSTM)** : تم اقتراح هذه البنية من قبل Zhou[3] 2015 وآخرون

لاستفادة من البنيتين CNN التي تقوم بالتقاط سمات محلية و ال RNN لالتقاط دللالات الكلمات في الجمل الشكل(12) .تم اختبار النموذج المقترح على مسألة تصنيف أسئلة إلى 6 أصناف تتضمن location, human, entity, abbreviation, description and numeric وحققت دقة تصل الى 94% .وهي مسألة شبيهة بمسألتنا .

ويكون تسلسل الطبقات وفقا للنموذج المقترح هو بالشكل (11):



الشكل (12)

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 81, 100)           465500
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 80, 800)           160800
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 1, 800)            0
_________________________________________________________________
lstm_7 (LSTM)                (None, 500)               2602000
_________________________________________________________________
dropout_11 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_11 (Dense)             (None, 7)                 3507
=================================================================
Total params: 3,231,807
Trainable params: 2,766,307
Non-trainable params: 465,500
_________________________________________________________________
None
Train on 5444 samples, validate on 2680 samples
Epoch 1/12
5444/5444 [==============================] - 52s 10ms/step - loss: 0.7062 - acc: 0.7605 - val_loss: 0.4122 - val_acc: 0.8627
Epoch 2/12
5444/5444 [==============================] - 50s 9ms/step - loss: 0.3352 - acc: 0.8797 - val_loss: 0.4261 - val_acc: 0.8537
```

الشكل (11)

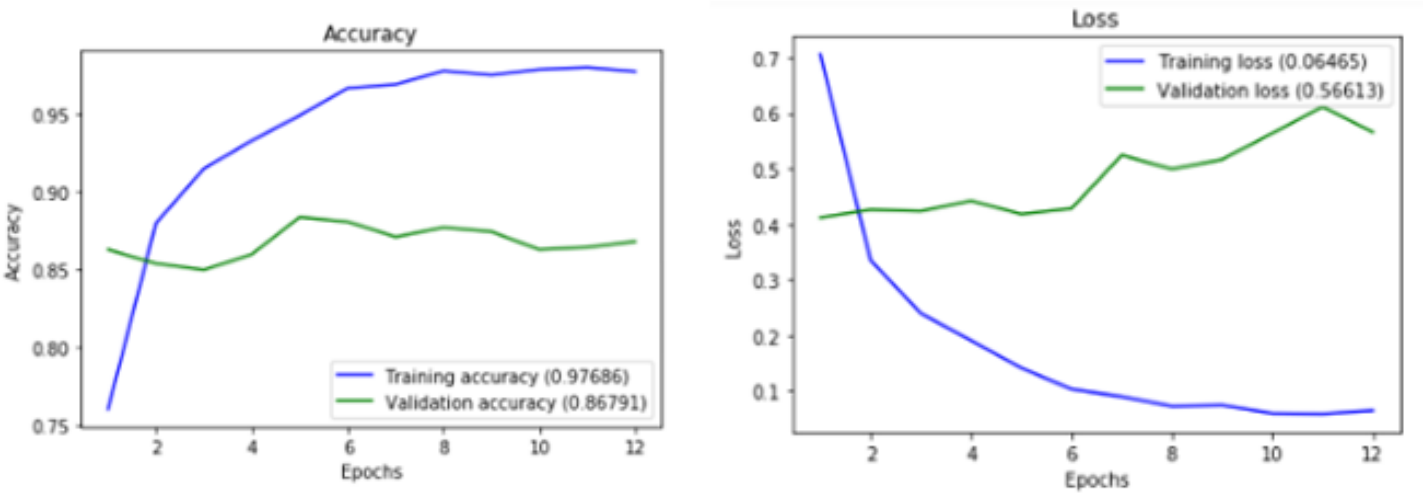
تم الاعتماد هذا النموذج على عدد من huperpararmeters وفقا للجدول التالي :

- Regularization : إضافة dropout الى خرج طبقة ال LSTM..

hyperparameter	Hyperprameter value
hidden cells of LSTM	800
no. of filtrers	800
N-gram (kernel size)	2
Batch_size	20
Epochs	12
Dropout_rate	0.6

وكانت النتائج هي :

دقة : 86,7% ونسبة الخطأ: 0.56

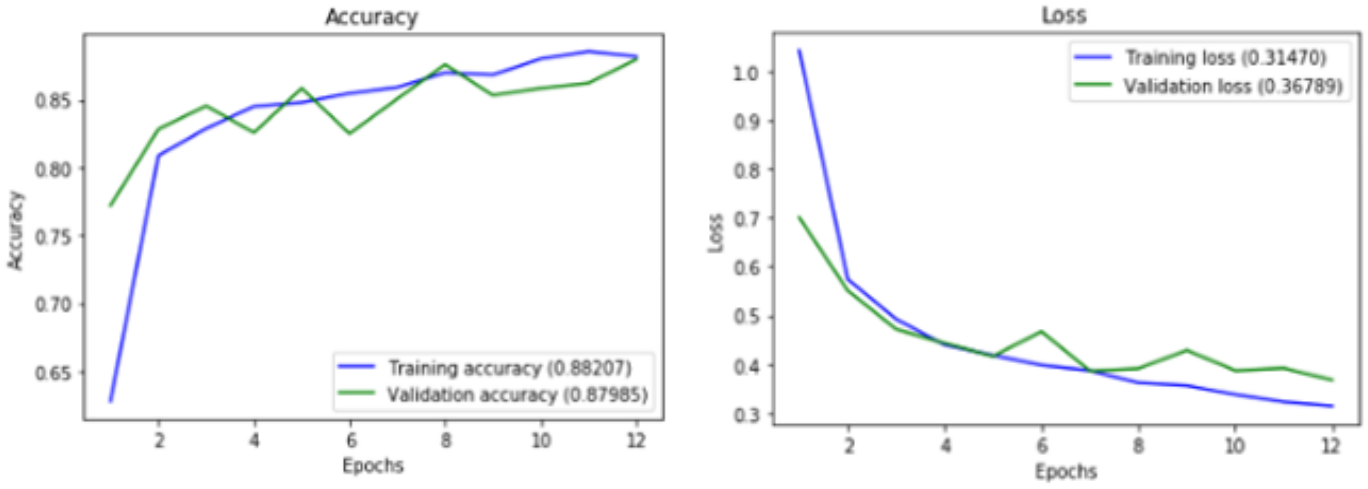


نلاحظ انها حالة overfitting لحل هكذا مشكلة نضيف dropout ايضا قبل طبقة ال convolution الشكل(13)

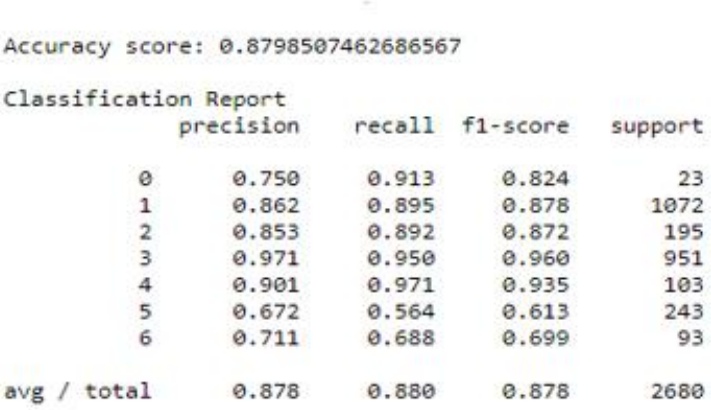
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 81, 100)           465500
_________________________________________________________________
dropout_12 (Dropout)         (None, 81, 100)           0
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 80, 800)           160800
_________________________________________________________________
max_pooling1d_8 (MaxPooling1 (None, 1, 800)            0
_________________________________________________________________
lstm_8 (LSTM)                (None, 800)               5123200
_________________________________________________________________
dropout_13 (Dropout)         (None, 800)               0
_________________________________________________________________
dense_12 (Dense)             (None, 7)                 5607
=================================================================
Total params: 5,755,107
Trainable params: 5,289,607
Non-trainable params: 465,500
_________________________________________________________________
None
Train on 5444 samples, validate on 2680 samples
Epoch 1/12
5444/5444 [==============================] - 88s 16ms/step - loss: 1.0422 - acc: 0.6277 - val_loss: 0.7002 - val_acc: 0.7720
Epoch 2/12
5444/5444 [==============================] - 94s 17ms/step - loss: 0.5741 - acc: 0.8088 - val_loss: 0.5507 - val_acc: 0.8284
```

الشكل(13)

,فنلاحظ تحسن في الأداء وحصلنا على دقة 87.9% ونسبة خطأ 0.36% .الشكل(14):



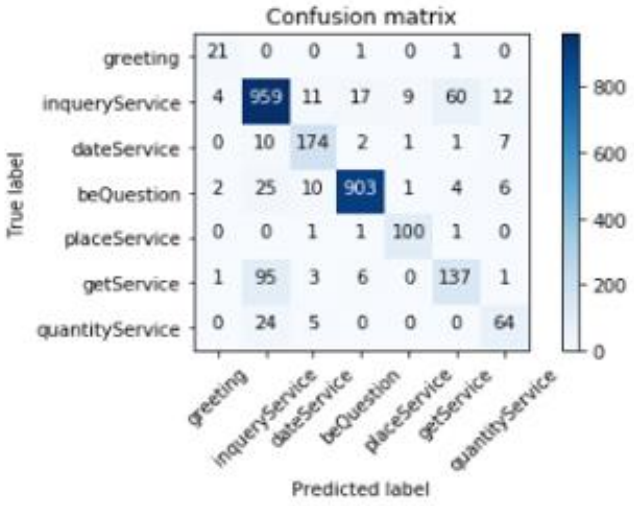
الشكل(14)



Accuracy score: 0.8798507462686567

Classification Report

	precision	recall	f1-score	support
0	0.750	0.913	0.824	23
1	0.862	0.895	0.878	1072
2	0.853	0.892	0.872	195
3	0.971	0.950	0.960	951
4	0.901	0.971	0.935	103
5	0.672	0.564	0.613	243
6	0.711	0.688	0.699	93
avg / total	0.878	0.880	0.878	2680



نتائج واستنتاجات :

تمت مقارنة أداء جميع نماذج المصنفات المستخدمة وفقاً للجدول التالي :

	Accuracy	Loss	Precision	Recall	F1
Nive byes	**70.2**	**0.50**	**0.68**	**0.71**	**0.63**
SVM	**87.1**	**0.10**	**0.85**	**0.87**	**0.86**
CNN	**88.7**	**0.34**	**0.88**	**0.88**	**0.88**
BiLSTM	**88.9**	**0.34**	**0.89**	**0.88**	**0.89**
CNN-LSTM	**87.9**	**0.36**	**0.87**	**0.88**	**0.87**

إن مسألة تصنيف النصوص او document classification هي مسألة **Precision-oriented**.لذلك يتم الاعتماد عليها غالبا في تقييم أداء المصنف إضافة إلى confusion matrix .

المصادر:

[1]: https://arxiv.org/ftp/arxiv/papers/1711/1711.08609.pdf

[2] : http://www.aclweb.org/anthology/D14-1181

[3]: https://arxiv.org/pdf/1511.08630.pdf