**جامعة تشرين كلية الهندسة المعلوماتية قسم الذكاء الصنعي**

الجمهورية العربية السورية

وزارة التعليم العالي

جامعة تشرين \_ كلية الهندسة المعلوماتية

قسم الذكاء الصنعي

**تشكيل النص العربي**

**Diacritize Arabic Text**

مشروع تخرج

**إعداد الطلاب**

عمار معلا ليما ديب روان دريباتي

**إشراف**

د.فاتن الكردي

العام الدراسي 2023\_2024

**ملخص**

في هذا المشروع، تم تطوير مجموعة نماذج ذكاء صنعي في عملية الترجمة (من اللغة الإنجليزية إلى اللغة الفرنسية)، تم مقارنة أداء المعماريات المستخدمة مع وبدون تضمين الكلمات في المدخلات  
 ( (word embedding, أظهرت النتائج أن استخدام تضمين الكلمات في المدخلات (word embedding) يحسن أداء النماذج في الترجمة بشكل كبير، حيث تمكنت النماذج التي استخدمت التضمين من تحقيق معدل دقة أفضل بكثير مقارنة بالنماذج التي لم تستخدم التضمين, من المهم ملاحظة أن الكثير من نماذج الذكاء الصنعي لا تصل إلى مرحلة الانتاج قمنا بتوصيف فصل كامل لمراحل تطوير نموذج ذكاء صنعي من مرحلة التخطيط إلى الإنتاج.

**Abstract**

In this project, a set of artificial intelligence models were developed for the translation process (from English to French). The performance of the architectures used was compared with and without word embedding in the inputs. The results showed that using word embedding in the inputs improves the models' performance in translation. The models that used word embedding were able to achieve much higher accuracy rates compared to the models that did not use embedding. It is important to note that many artificial intelligence models do not reach the production stage. We have described a complete chapter on the stages of developing an artificial intelligence model from the planning to production stage.

**الفهرس**

[الفصل الأول: مقدمة 1](#_Toc142519395)

[**الدراسات المرجعية** 2](#_Toc142519396)

[**1-1-1 الأوراق المرجعية التي اعتمدت** 2](#_Toc142519397)

[**1-1-1 جدول التلخيص** 3](#_Toc142519398)

[**2-1البيئة والأدوات المستخدمة** 5](#_Toc142519399)

[**3-1 منهجية البحث** 6](#_Toc142519400)

[الفصل الثاني : الدراسة النظرية 7](#_Toc142519401)

[**2-1 الترجمة** 7](#_Toc142519402)

[**2-2 Word Embedding** 8](#_Toc142519403)

[**2-3 Recurrent Neural Network (RNN)** 10](#_Toc142519404)

[**2-3-1 الشبكات العصبونية المتكررة البسيطة RNN** 10](#_Toc142519405)

[**2-3-2 LSTM (الذاكرة طويلة –قصيرة الأمد )** 14](#_Toc142519406)

[**2-4 التوابع المستخدمة** 15](#_Toc142519407)

[**2-4-1 RELU** 15](#_Toc142519408)

[**2-4-2 sigmoid** 16](#_Toc142519409)

[**2-4-3 softmax** 17](#_Toc142519410)

[الفصل الثالث: الجزء العملي 18](#_Toc142519411)

[**3-1 التخطيط planning** 18](#_Toc142519412)

[**3-2 الداتا ومعالجتها Data and Processing** 20](#_Toc142519413)

[**3-3 بناء وتدريب النماذج Build and Train Models** 23](#_Toc142519414)

[**3-3-1 النتائج** 37](#_Toc142519415)

[**3-4 التعبئة packaging** 39](#_Toc142519416)

[**3-5 مرحلة المنتج النهائي Deployment** 47](#_Toc142519417)

[**3-6 مرحلة المراقبة Monitior** 52](#_Toc142519418)

[التوصيات 53](#_Toc142519419)

[المراجع Reference 54](#_Toc142519420)

**فهرس الأشكال**

الشكل 1-2-...........................................................................................9

الشكل 2-2 ..........................................................................................12

الشكل 2-3 ..........................................................................................14

الشكل 2-4...........................................................................................16

الشكل 2-5 .........................................................................................16

الشكل 2-6 ..........................................................................................17

الشكل 3-1...........................................................................................18

الشكل 3-2..........................................................................................22

الشكل 3-3...........................................................................................26

الشكل 3-4...........................................................................................28

الشكل 3-5 ...........................................................................................30

الشكل 3-6 ...........................................................................................32

الشكل 3-7 ..........................................................................................33

الشكل 3-8 ..........................................................................................36

الشكل 3-9 ..........................................................................................37

الشكل 3-10 .........................................................................................38

الشكل 3-11 .........................................................................................40

الشكل 3-12 .........................................................................................42

الشكل 3-13 .........................................................................................43

الشكل 3-14 .........................................................................................45

الشكل 3-15 .........................................................................................46

الشكل 3-16 .........................................................................................48

الشكل 3-17 .........................................................................................52

**فهرس الجداول :**

الجدول(1-1) .........................................................................................4

**قائمة المصطلحات:**

|  |  |
| --- | --- |
| **المختصر الموافق** | **المصطلح** |
| RNN | Recurrent Neural Networks |
| DNN | Deep Neural Networks |
| LSTM | Long Short-Term Memory |
| MLOps | Machine Learning Operations |
| API | **Application Programming Interface** |

# الفصل الأول: مقدمة

تُعتبر اللغة العربية من أكثر اللغات انتشارًا في العالم، إلا أنها تُعاني من صعوبة في الكتابة، خاصة للمبتدئين، بسبب عدم وجود حركات على الحروف العربية في معظم النصوص المكتوبة. يؤدي ذلك إلى صعوبة قراءة وفهم اللغة العربية، مما قد يُثبط المستخدمين عن تعلمها واستخدامها.

"Diacritize Arabic Text" هو مشروع يهدف إلى توفير أدوات سهلة الاستخدام لمعالجة النصوص العربية، بما في ذلك إضافة الحركات (الضمائر) إلى الكلمات العربية. تهدف هذه الخدمة إلى تبسيط كتابة اللغة العربية وتحسين قابلية قراءتها وفهمها، خاصة للمستخدمين الذين لا يتقنون اللغة العربية بشكل كامل.

التحديات التي تواجهه :

1. تعددية اللهجات: اللغة العربية تتميز بالتعددية اللهجات والأصوات المختلفة. وبالتالي، يمكن أن يحتاج مشروع تشكيل النصوص العربية بالحركات إلى معالجة هذه التنوعات اللهجوية لضمان الدقة والفعالية في تطبيق الحركات.
2. الكلمات غير المعروفة: يمكن أن يواجه المشروع صعوبة في تشكيل الكلمات غير المعروفة أو النادرة التي لم تتواجد في قاعدة البيانات. وهذا يتطلب تحديث مستمر لقاعدة البيانات وإضافة الكلمات الجديدة.
3. التحديات التقنية: يتطلب تشكيل النصوص العربية بالحركات استخدام تقنيات متطورة لمعالجة اللغة الطبيعية وتحليل النصوص. قد تواجه التقنيات التي تستخدم في المشروع بعض التحديات التقنية مثل التعامل مع الإشكاليات النحوية والصرفية المعقدة .

## **الدراسات المرجعية**

## **1-1-1 الأوراق المرجعية التي اعتمدت**

**Neural Arabic Text Diacritization: State of the Art Results and a Novel-**

**Approach for Machine Translation**

**المؤلفون :** Ali Fadel, Ibraheem Tuffaha, Bara' Al-Jawarneh, and Mahmoud Al-Ayyoub

**سنة النشر:** 2019

**ملخص:**

* تتناول هذه الورقة البحثية تطوير تقنية حديثة لتمثيل النصوص العربية باستخدام الشبكات العصبونية.
* تعتمد هذه التقنية على تمثيل كل كلمة بشعاع ثم إسقاطها في فضاء هيكلي ذي بنية محددة، بحيث تعكس العلاقات الهندسية بين متجهات الكلمات العلاقات الدلالية بينها.
* تم استخدام FFNN و RNN .
* تُعدّ هذه التقنية حاسمة لتطوير نماذج الترجمة الآلية باستخدام الشبكات العصبونية، حيث يتم استخدام تمثيل الكلمات (word embedding) لتمثيل الجمل بشكل رقمي، مما يسمح للشبكات العصبية بفهم معنى ومضمون الجمل والكلمات.
* يتناول قسم FFNN في البحث ثلاثة نماذج تعتمد على شبكات عصبية تغذية أمامية (FFNN) لمهمة تشكيل النص العربي. تُعنى هذه المهمة بوضع علامات التشكيل (الحركات) فوق الحروف العربية وهذه النماذج هي :
* Basic model يتكون من 17 طبقة خفية ويستخدم ReLU ويحوي على 1.5 مليون معلمة قابلة للتدريب .
* Hot-100 : يحول كل عدد صحيح مدخل إلى one-hot vector ب75 بعد , يتكون من 5 طبقات خفية مع dropout ويحتوي على 2 مليون معلمة قابلة للتدريب.
* Embedding : يتكون النموذج من 5 طبقات خفية ويحوي على 728 ألف معلمة قابلة للتدريب.
* يتكون نموذج RNN من :
* Embedding layer .
* BiCuDNNLSTM layers : لمعالجة التسلسلات الزمنية لمدخلات التضمين.
* Dropout layer :تطبّق بعد كل طبقة BiCuDNNLSTM لمنع overfitting .
* Fully-Connected Feed-Forward layers: تُستخدم لتحويل مخرجات طبقة BiCuDNNLSTM الأخيرة إلى مخرجات متوقعة.
* Softmax layer.
* النموذج ToD :

## نموذج تسلسل إلى تسلسل (seq2seq) من نوع مشفر ومفكك (Encoder-Decoder)

## يتكون النموذج من شبكة عصبونية من نوع مشفر ومفكك (Encoder-Decoder) لترجمة الجمل من اللغة العربية إلى اللغة الإنجليزية.

* **مكونات النموذج:**
* **المُشفّر (Encoder):**
  + طبقة CuDNNLSTM ثنائية الاتجاه (256 وحدة لكل اتجاه، بإجمالي 512 وحدة).
  + وظيفتها تحويل الجملة العربية المدخلة إلى تمثيل متجه.
* **المُفكّك (Decoder):**
  + طبقة CuDNNLSTM بحجم 512 وحدة.
  + وظيفتها تحويل تمثيل المتجه الذي تم إنشاؤه بواسطة المُشفّر إلى الجملة العربية المترجمة.
* **الانتباه الإضافي (Additive Attention):**
  + يتم تطبيقه على مخرجات المُشفّر.
  + يساعد المُفكّك على التركيز على الأجزاء المهمة في الجملة العربية الأصلية أثناء عملية الترجمة.
* **طبقة التضمينات:**
  + طبقة عشوائية الإنشاء لتمثيل الوحدات الفرعية (بعد تطبيق BPE) بحجم متجه 64.
  + عند التدريب بدون تشكيل، تُستخدم هذه الطبقة فقط.
  + عند التدريب مع التشكيل، تُستخدم طبقة إضافية بنفس التكوين لتمثيل التشكيل.
  + يتم دمج مخرجات الطبقتين عند التدريب مع التشكيل.

- **Arabic Text Diacritization Using Deep Neural Networks**

المؤلفون : Ali Fadel, Ibraheem Tuffaha, Bara’ Al-Jawarneh and Mahmoud Al-Ayyoub .

سنة النشر : 2019

1. هذه الورقة تعرض نظامًا قائمًا على الشبكات العصبية العميقة لتشكيل النصوص العربية بدون الاعتماد على الميزات والأدوات اللغوية التقليدية. تم تدريب النظام على نصوص مشكولة مستخرجة من مجموعة بيانات Arabic Treebank باستخدام أنواع مختلفة من الشبكات العصبية مثل Feed-Forward وLSTM و .Bidirectional LSTM
2. **النتائج**: النتائج التجريبية تظهر أن النموذج المقترح يحقق أداءً متفوقًا مقارنة بالأساليب التقليدية والأساليب المعتمدة على التعلم الآلي السابقة.
3. **التطبيقات**: يمكن استخدام النموذج في تطبيقات عديدة مثل تحسين محركات البحث، تعليم اللغة العربية، والتطبيقات الدينية.
4. **التحديات المستقبلية**: التحديات المستقبلية تشمل تحسين دقة التشكيل بشكل أكبر، معالجة النصوص غير الرسمية واللهجات العامية، وتوسيع النموذج ليشمل نصوصًا ذات طبيعة مختلفة.

باختصار، الورقة البحثية تقدم نموذجًا مبتكرًا يستخدم الشبكات العصبية العميقة لتشكيل النصوص العربية بدقة عالية، متفوقة على الأساليب التقليدية، وتفتح الطريق لمزيد من التحسينات والتطبيقات المستقبلية في هذا المجال.

## **1-1-1 جدول التلخيص**

|  |  |  |  |
| --- | --- | --- | --- |
| **مساؤى** | **محاسن** | **الأفكار الرئيسية** | **اسم الورقة البحثية** |
| **تعقيد النماذج المقترحة**  قلة البيانات المُدرجة  صعوبة تقييم جودة الترجمة الآلية العربية | -تحسين دقة تشكيل النص العربي.  -القدرة على التعامل مع مختلف أنواع النصوص العربية.  **-تحسين دقة الترجمة الآلية العربية.**  -سهولة التكامل مع نماذج الترجمة الآلية العصبية الحالية. | **-تقديم نماذج جديدة باستخدام الشبكات العصبونية العميقة:** تتضمن هذه النماذج تقنيات مثل LSTM، و convolutional layers، و attention mechanisms .  **-اقتراح نهج جديد للترجمة الآلية العربية باستخدام التنقيط (ToD)** **:**يعتمد هذا النهج على ترجمة النص العربي المُدرج بالكامل (مع الحركات) بدلاً من ترجمة النص غير المُدرج. | Efficient Estimation of Word Representations in Vector Space |
| الحاجة إلى بيانات كبيرة .  التعقيد الحسابي . | الدقة العالية مقارنة بالأساليب التقليدية .  عدم الاعتماد على الميزات اللغوية . | استخدام أنواع مختلفة من الشبكات العصبية العميقة مثل Feed-Forward وLong Short-Term Memory (LSTM) وBidirectional LSTM (B-LSTM) وstacked B-LSTM لتحقيق تشكيل دقيق للنصوص العربية.  تم تطوير نموذج متعدد الطبقات يعتمد على تعلّم الحروف والتراكيب اللغوية بدون الاعتماد على الميزات اللغوية التقليدية.  تم تدريب النموذج على نصوص مشكولة من مجموعة بيانات Arabic Treebank، مما يساعد في تحسين دقة التشكيل. | Arabic Text Diacritization Using Deep Neural Networks |

(الجدول 1-1)

## **2-1البيئة والأدوات المستخدمة**

في هذه الفقرة، سنشرح بإيجاز عن الأدوات والبيئة التي تم استخدامها في هذا البحث، والتي تشمل

Github Actions، Docker، Python، Jupyter Notebook، FastAPI، Colab، TensorFlow، Keras، NumPy، PyYAML، Pickle.

* Github Actions هو خدمة تقدمها Github لإدارة عمليات التطوير الخاصة بمشاريع البرمجة. باستخدام Github Actions يمكنك إنشاء نظام للتحكم بعمليات التطوير والإختبار والنشر بصورة آلية، بحيث يمكنك تحسين الإنتاجية وتوفير الوقت اللازم لإنجاز المهام. يتم إعداد Github Actions بإستخدام ملفات YAML التي تحتوي على سلسلة من الخطوات التي يجب تنفيذها.
* Docker هو برنامج يساعد على إنشاء وتشغيل بيئات البرمجة في أي نظام تشغيل، بما في ذلك Linux وWindows وMacOS. يساعد Docker في توفير بيئة متكاملة للتطوير والتشغيل، ويسمح بإنشاء حاويات (containers) تحتوي على جميع المكتبات والأدوات والإعدادات اللازمة لتشغيل تطبيقك.
* Python هي لغة برمجة شائعة الاستخدام ومفتوحة المصدر، وتستخدم في العديد من المجالات بما في ذلك الذكاء الاصطناعي وعلوم البيانات وتطوير الويب.
* Jupyter Notebook هو برنامج تفاعلي للبرمجة يسمح للمستخدمين بكتابة الشفرات وعرض النتائج في واجهة مستخدم بديهية. يتم استخدام Jupyter Notebook عادة للتعلم العملي والتجارب السريعة في علوم البيانات والذكاء الاصطناعي.
* Colab هو بيئة تفاعلية تقدمها Google، ويتم استخدامها عادة لتطوير النماذج الذكية والتعلم العميق وعلوم البيانات. يتميز Colab بواجهة مستخدم بديهية وخدمات السحابة التي يتم توفيرها من قبل Google.
* TensorFlow هو إطار عمل للتعلم الآلي والذكاء الاصطناعي، ويستخدم عادة لبناء الشبكات العصبية وتدريبها.
* Keras هو إطار عمل للتعلم الآلي يُستخدم كواجهة عالية المستوى لتطبيقات TensorFlow. يعتبر Keras سهل الاستخدام ويتميز بالمرونة والسرعة في بناء الشبكات العصبية وتدريبها.
* NumPy هو مكتبة لغة بايثون مفتوحة المصدر تُستخدم للعمليات الرياضية والعلمية، وتتضمن مجموعة من الأدوات والوظائف المفيدة لتحليل ومعالجة البيانات الكبيرة. يتميز NumPy بالسرعة والكفاءة والتعامل مع البيانات في صيغة المصفوفات.
* yaml هو مكتبة لغة بايثون مفتوحة المصدر يتم استخدامها للتعامل مع صيغة YAML يستخدم YAML عادة لتخزين البيانات المنظمة بشكل هرمي، ويمكن استخدام yaml لتحويل البيانات بين صيغ YAML وصيغ أخرى.
* Pickle هو مكتبة لغة بايثون تُستخدم للتسلسل والإلغاء التسلسل للبيانات. يستخدم Pickle عادة لحفظ واستعادة البيانات في صيغة ملف، ويتميز بالسهولة والكفاءة في التعامل مع البيانات.
* Neptune.ai هو إطار عمل يستخدم لتتبع ومراقبة نماذج الذكاء الصنعي في مرحلة التعلم.
* FastAPI هو إطار عمل يستخدم لبناء Restful Api لتحويل النماذج إلى micro service.

## **3-1 منهجية البحث**

Dataset: جمل مشكلة

معالجة الداتا

استخدام تقنية التضمين   
Charater Embedding

استخدام الشبكة العصبونية

BI-LSTM

تشكيل الجملة العربية

# الفصل الثاني : الدراسة النظرية

## **2-1 تشكيل النص العربي :**

اللغة العربية، بجمالها وتعقيدها، تعدّ إحدى أعظم كنوز الحضارة الإنسانية. تمتاز هذه اللغة بقدرتها الفريدة على التعبير الدقيق والغني، ويعود الفضل في ذلك جزئيًا إلى نظام الحركات التشكيلية الذي يضفي على النصوص وضوحًا ودقة في النطق والمعنى. في عصرنا الرقمي، أصبحت الحاجة ماسة إلى تقنيات حديثة تستطيع تشكيل النصوص العربية تلقائيًا وبكفاءة عالية، لضمان المحافظة على جمالية اللغة ودقتها.

التشكيل ليس مجرد إضافة زخرفية؛ بل هو عنصر أساسي لفهم النصوص العربية بشكل صحيح. الكلمات العربية غالبًا ما تكون متعددة المعاني، ويعتمد المعنى الدقيق على الحركات التشكيلية. على سبيل المثال، كلمة "عَلِمَ" تعني "عرف" بينما "عُلِمَ" تعني "أُخبر". من دون هذه الحركات، يصبح الفهم مشوشًا وغير دقيق، مما يبرز الحاجة إلى أنظمة آلية قادرة على تشكيل النصوص بدقة عالية.

### تطبيقات تشكيل النصوص

تطبيقات التشكيل الآلي للنصوص العربية متعددة وتشمل:

1. **التعليم**: يمكن استخدام هذه التقنيات في تطوير أدوات تعليمية تساعد الطلاب على تعلم القراءة بشكل صحيح.
2. **التطبيقات الدينية**: تشكيل النصوص الدينية مثل القرآن الكريم يساعد في قراءته وتلاوته بشكل صحيح.
3. **تحسين محركات البحث**: يمكن لتحسين دقة التشكيل أن يعزز من قدرات محركات البحث في معالجة وفهم النصوص العربية بشكل أفضل.

مشروع تشكيل النصوص العربية باستخدام الذكاء الاصطناعي ليس مجرد خطوة تقنية، بل هو رحلة نحو الحفاظ على تراث لغوي غني وتعزيز استخدامه في العصر الرقمي. من خلال تطوير أنظمة ذكية قادرة على تشكيل النصوص بدقة عالية، نضمن استمرار جمال اللغة العربية ودقتها، مما يساهم في إثراء المحتوى العربي وتعزيز التواصل الفعال بين الأجيال الحالية والقادمة.

## **2-2 Word Embedding**

عند القيام ببناء نماذج تعلم الآلة مهمّتها فهم وتفسير اللغات الطبيعيّة المتدفّقة مثل لغات البشر من غير الممكن لهذه النّماذج التّعامل مع البيانات النصيّة بشكل مباشر، فهي ليست ذكيّة بما فيه الكفاية لبدء معالجة النّص في شكلها الأصليّ، نماذج تعلّم الآلة تعتمد بشكل رئيسيّ على مبادئ الإحصاء والرّياضيات والتّحسين، ولا تفهم سوى لغة الأرقام الأمر الذي يستدعي إجراء معالجة مسبقة للبيانات النصيّة وتحويلها إلى تمثيل عدديّ مقابل لها، إنّ تحويل البيانات النصيّة  إلى بيانات عدديّة مقابلة لها سيمكّن حتماً خوارزميّة التعلم الآليّ من فهمها والتعامل معها؛ حيث تعمل خوارزميّات التضمين Embedding على إيجاد تمثيل عدديّ مقابل للكلمات النصيّة بشكل فعّال ,سنستخدم في هذا المشروع word embedding .

في هذه الطّريقة يتمّ أوّلاً تجميع الكلمات التي نرغب في إيجاد ترميز لها، ومن ثمّ اختيار بُعد الشعاع للكلمة[1]، هنا البُعد يرمز إلى عدد الميّزات features، هذه الميّزات ستصف الكلمة (سنوضح هذا لاحقاً)، في هذه الحالة نستطيع التحكّم ببُعد شعاع الكلمة (عدد الميّزات) ونستطيع جعله أقلّ بكثير من العدد الإجماليّ للمفردات، قيمة كلّ ميّزة في الشعاع تكون عدداً حقيقيّاً ضمن المجال [1,-1]، وكلّما كانت القيمة قريبة من الواحد كلّما كانت الميّزة تعبر عن الكلمة والعكس في حال اقتربت من الصّفر. على سبيل المثال بإمكاننا اختيار 300 ميّزة في حال كان لدينا قاموس يحتوي على  10000  مفردة.

مثال توضيحيّ :

بفرض لدينا المفردات التالية

{ولد , أميرة , ملكة , ملك}

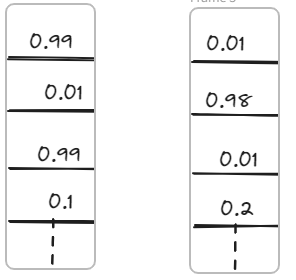
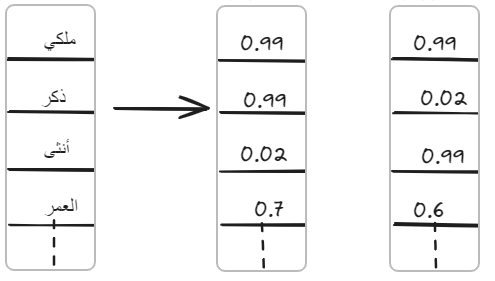
والميّزات المستخدمة لهذه المفردات هي:

(العمر, أنثى, ذكر, ملكي)

لدينا الشكل التالي يعبر عن شعاع كل كلمة [3]:

ملك ملكة أميرة ولد

(الشكل 2-1 : يوضح مميزات كل كلمة )



نلاحظ من الشكل السابق أن الميّزة "ذكر" تخص الكلمتين "ملك" و "ولد" فقط لذلك تكون قيمتها عند هاتين الكلمتين قريبة من الواحد , بينما هذه الميّزة لا تخصّ الكلمتين "ملكة" و "أميرة" لذلك تكون قيمتها عند هاتين الكلمتين قريبة من الصفر .

باستخدام هذه الطريقة استطعنا تمثيل الكلمات التي تتشابه في المعنى بأشعّة قريبة جدآ من بعضها : أي أنّ المسافة الإقليدية بينهما تكون صغيرة وهذا مفيد جدآ في حالات كثيرة منها حالة تعميم الشبكة العصبونية .

فعلى سبيل المثال في حال أردنا إجراء تحليل مشاعر للجمل النصية وتصنيفها إلى ثلاثة أصناف هي إيجابي positive وسلبي negative وحيادي Neutral . في حال كان لدينا الجملتين التاليتين :

1. بعد استخدام الجهاز لمدة شهر أدركت أنه جهاز جميل .
2. بعد استخدام الجهاز لمدة شهر أدركت أنه جهاز رائع .

نذكر أن الكلمتين "رائع" و "جميل" متشابهتان جدآ بالمعنى ويتم استخدامهما في سياق متشابه .

أثناء تدريب الشبكة العصبونية كانت الجملة الأولى موجودة في مجموعة بيانات التدريب Training set وتم تدريب الشبكة العصبونية على أن يكون خرج هذه الجملة إيجابيآ .

الجملة الثانية موجودة في مجموعة بيانات الاختبار Test dataset ولم تتدرب عليه الشبكة أثناء عملية التدريب , في حال أدخلنا الجملة الثانية إلى الشبكة العصبونية لمعرفة خرج هذه الجملة مع العلم أن كلمة "رائع" غير موجودة أبدآ في بيانات التدريب . وباستخدام أسلوب التضمين سيتمّ تمثيل كلمة "رائع" بشعاع قريب جدآ من "جميل" وبالتالي ستتمكن الشبكة العصبونية من إعطاء خرج إيجابي .

فنستنتج ميزة كبرى لدى تضمين الكلمات word embedding وهي أنها قادرة على إجراء تعميم بشكل كبير جدآ للشبكة العصبونية .

## **2-3 Recurrent Neural Network (RNN)**

عندما نتحدث عن أي مشكلة في NLP مع الشبكات العصبية , فإن أول أمر يتبادر للأذهان هو الشبكات العصبية المتكررة , فهي الأكثر فعالية لأنها تعالج التسلسلات النصية , وما يلي هي أنواع هذه الشبكات :

### **2-3-1 الشبكات العصبونية المتكررة البسيطة RNN**

نوع من الشبكات العصبونية الاصطناعية التي تعتمد على البيانات التسلسلية، وتستخدم في تطبيقات التعلم العميق التي تتعامل مع البيانات المتتابعة والسلاسل الزمنية مثل ترجمة اللغات ومعالجة اللغة الطبيعية والتعرف التلقائي على الكلام المنطوق ومن أهم الأمثلة على استخدامها البحث الصوتي وخدمة الترجمة من جوجل[1].

تعتبر الشبكات العصبونية المتكررة قديمة نسبياً، حيث ظهرت لأول مرة في ثمانينيات القرن الماضي. لكن لم تتم الاستفادة من إمكانياتها بالشكل الأمثل إلا فى السنوات الأخيرة تزامناً مع توفر البيانات الضخمة والقدرات الحاسوبية العالية.

وتتمتع هذه الشبكات بميزة الذاكرة" التي تسمح لها بأخذ معلومات من بيانات الدخل السابقة للتأثير على بيانات الدخل والمخرجات الحالية. فعلى عكس الشبكات العصبونية العميقة الأخرى التي تفترض أن المدخلات مستقلة عن المخرجات، يعتمد الخرج في الشبكات المتكررة على العناصر السابقة في تسلسل البيانات .

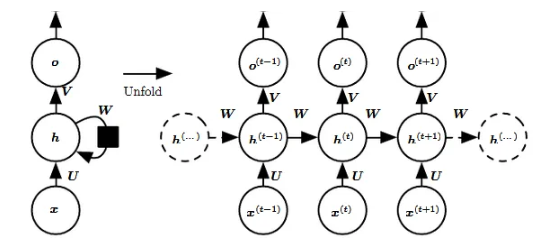
كيف تعمل الشبكات العصبونية المتكررة ؟

تختلف الشبكات العصبونية المتكررة عن شبكات التغذية إلى الأمام (Feed-Forward)، فشبكات التغذية إلى الأمام تأخذ كمية ثابتة من بيانات الدخل دفعة واحدة وتعطي كمية ثابتة من المخرجات في كل مرة . على الجانب الآخر لا تأخذ الشبكات العصبونية المتكررة جميع بيانات الدخل معاً، إنما تأخذ عنصراً واحداً من تلك البيانات في كل مرة بشكل متسلسل , ثم تجري عليه مجموعة من العمليات الحسابية المتسلسلة قبل إنتاج الخرج الموافق[2]، ويعرف هذا الخرج باسم الحالة المخفية ( Hidden State)يتم بعد ذلك دمج الخرج الناتج عن المرحلة الحالية مع عنصر الدخل التالي لإنتاج خرج جديد وتستمر العملية حتى انتهاء تسلسل بيانات الدخل، ويمكن أيضاً برمجة النموذج للتوقف عند مرحلة معينة .

إذا في حالات مثل عندما يكون مطلوبا التنبؤ بالكلمة التالية من الجملة، تكون الكلمات السابقة مطلوبة وبالتالي هناك حاجة لتذكر الكلمات السابقة. وهنا تكمن أهمية شبكات RNN ، والتي حلت هذه المشكلة عن طريق استخدام مانسميه "Hidden state ، فهي الميزة الرئيسية والأكثر أهمية لـ RNN ، والتي تتذكر بعض معلومات التسلسل . أي أنها تعالج التسلسلات "sequences كالنصوص و السلاسل الزمنية (رغم أنها ليست ناجحة جداً مع السلاسل كما أشار فرانسوا كوليت ) [3].

إن ما يميزها كما قلنا هو استخدام ذاكرة تتذكر جميع المعلومات حول ما تم حسابه في المراحل السابقة، وتستخدم نفس الأوزان لكل إدخال لأنه يؤدي نفس المهمة على جميع المدخلات أو الطبقات المخفية لإنتاج المخرجات , على عكس الشبكات العصبية الأخرى. وأيضاً إحدى المشكلات المتعلقة بالشبكات العصبية التقليدية وكذلك شبكات (CNN) هي أنها تعمل فقط بأحجام محددة مسبقاً، فهي تأخذ مدخلات ذات حجم ثابت وتنتج مخرجات ذات حجم ثابت. أما RNNS تتيح لنا تسلسلات متغيرة الطول كمدخلات ومخرجات ولهذا فهي الخيار رقم 1 للتعامل مع ومعالجة اللغة الطبيعية NLP .

إن الشيئ الذي يعطي هذه الشبكات الخاصية التكرارية هو أنها تستخدم نفس الأوزان لكل خطوة .

والشكل التالي يوضح شكل الشبكة RNN :

الشكل 2-2 : (شبكة RNN)

يُظهر الجانب الأيسر من الرسم البياني أعلاه تدوينًا لـ RNN وعلى الجانب الأيمن يتم فتح RNN (أو فتحه) في شبكة كاملة. من خلال إلغاء التمرير ، نعني أننا نكتب الشبكة للتسلسل الكامل. على سبيل المثال ، إذا كان التسلسل الذي نهتم به عبارة عن جملة من 3 كلمات ، فسيتم فتح الشبكة في شبكة عصبية ثلاثية الطبقات .

الإدخال: يتم أخذ x (t) كمدخل للشبكة في الخطوة الزمنية t. على سبيل المثال ، x1 ، يمكن أن يكون one-hot vector ، الموافق لكلمة في الجملة .

الحالة المخفية: h (t) تمثل حالة مخفية في الوقت t وتعمل بمثابة "ذاكرة" للشبكة. h (t) يُحسب بناءً على الإدخال الحالي والحالة المخفية لخطوة الوقت السابقة: h (t) = f (U x (t) + W h (t − 1)). [2]تعتبر الدالة f تحويلاً غير خطي مثل tanh و ReLU (سنتحدث عنها لاحقآ ) .

الأوزان: يحتوي RNN على hidden connections (مدخلات للوصلات المخفية )بواسطة مصفوفة الوزن U ، و hidden-to-hidden recurrent connections (الوصلات المتكررة المخفية إلى المخفية المعلمة ) بواسطة مصفوفة الوزن W ، و hidden-to-output connections (الوصلات المخفية إلى المخرجات المحددة ) بواسطة مصفوفة الوزن V وجميع هذه الأوزان (U ، V، W) تتم مشاركتها عبر الوقت.

الخرج : O(t) .

من المفترض أن تحمل RNN المعلومات حتى وقت طويل ومع ذلك فإنه من الصعب نشر كل هذه المعلومات عندما تكون الخطوة الزمنية طويلة جدآ , وعندما تحوي الشبكة على عدد كبير من الطبقات العميقة تصبح غير قابلة للتدريب , تسمى هذه المشكلة (اختفاء مشكلة التدرج vanishing gradient problem ) : أي أن الشبكة العصبية تقوم بتحديث الأوزان باستخدام خوارزمية النسب للتدرج , تنمو التدرجات أصغر عندما تتقدم الشبكة إلى الطبقات السفلية , إذا بقى التدرج ثابت يعني أنه لا يوجد مساحة للتحسين , هذا يؤثر على خرج الشبكة , ومع ذلك إذا كان الفرق في التدرج صغير جدآ (على سبيل المثال تتغير الأوزان قليلآ ) فان تتمكن الشبكة من تعلم أي شيء .

للتغلب على هذه المشكلة تم تحسين RNN إلى LSTM حيث توفر للشبكة معلومات سابقة ذات صلة إلى وقت أكثر حداثة .

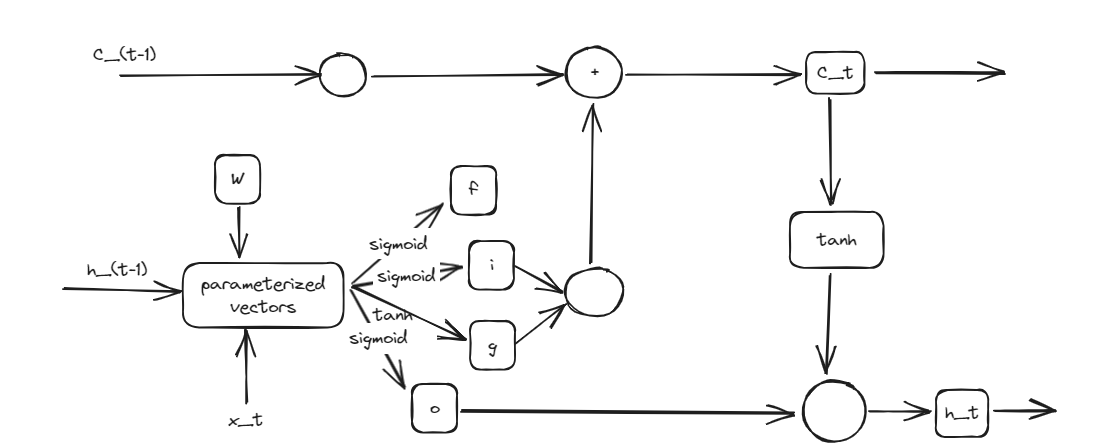
سنتعرف على LSTM:

### **2-3-2 LSTM (الذاكرة طويلة –قصيرة الأمد )**

إنها مجموعة متنوعة من الشبكات العصبية المتكررة (RNNs) , يحتوي LSTM على اتصالات تغذية مرتدة ، أي أنه قادر على معالجة تسلسل البيانات بالكامل ، بصرف النظر عن نقاط البيانات الفردية مثل الصور. يجد هذا التطبيق في التعرف على الكلام ، والترجمة الآلية ، وما إلى ذلك. LSTM هو نوع خاص من RNN ، والذي يظهر أداءً متميزًا في مجموعة متنوعة من المشكلات.

يتم عقد الدور المركزي لنموذج LSTM بواسطة خلية ذاكرة تُعرف باسم "حالة الخلية" التي تحافظ على حالتها بمرور الوقت. حالة الخلية هي الخط الأفقي الذي يمر عبر الجزء العلوي من الرسم التخطيطي أدناه. يمكن تصورها كحزام ناقل تتدفق من خلاله المعلومات دون تغيير.

يمكن إضافة المعلومات إلى حالة الخلية أو إزالتها منها في LSTM ويتم تنظيمها بواسطة البوابات[1]. تسمح هذه البوابات اختياريًا للمعلومات بالتدفق داخل وخارج الخلية. يحتوي على pointwise multiplication operation و sigmoid neural net layer تساعد الآلية.

تعطي sigmoid layer أرقامًا بين صفر وواحد ، حيث يعني الصفر "لا يجب السماح بمرور أي شيء" ، ويعني الواحد "أنه يجب السماح لكل شيء بالمرور".

(الشكل 2-3 : شبكة LSTM )

## **2-4 التوابع المستخدمة**

## **2-4-1 RELU**

يتم استخدام RELU كوظيفة تنشيط افتراضية , وهي وظيفة التنشيط الأكثر استخدامآ في الشبكات العصبية , خاصة في شبكات CNN .

تعيد الدالة 0 إذا تلقّت أي إدخال سالب , وتعيد القيمة الموجبة نفسها التي تتلقّاها[2] .

بالتالي فإنه يعطي ناتجآ يتراوح بين 0 و لانهاية .

أحد سلبيات تابع RELU أنه يقوم بتحويل جميع القيم السالبة إلى صفر، اي انه في حال وجود قيم في التدريب مختلفة سالبة ينظر إليها التابع بشكل واحد وهو الصفر, مما قد يتسبب بخسارة معلومات هامة لازمة لتدريب الشبكة العصبية.

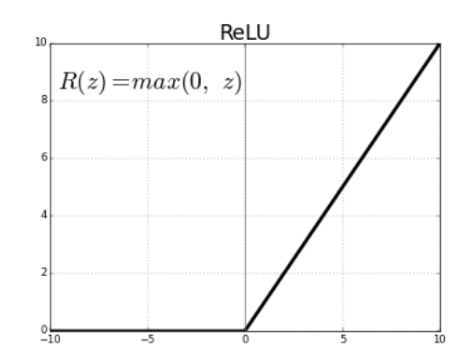
Def RLU (x):

If (x>0) :

return x

else:

return 0



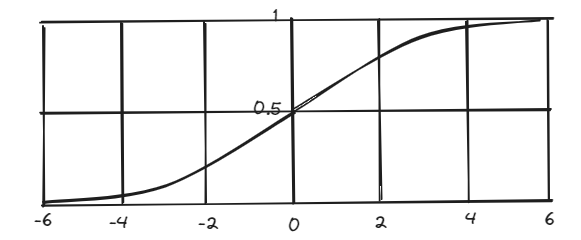
(الشكل 2-4 : تابع RELU )

### **2-4-2 sigmoid**

تعمل الدالة sigmoid مع مدخلات من قيم حقيقية , وتعطي قيم مستمرة بين 0 و 1[2] .

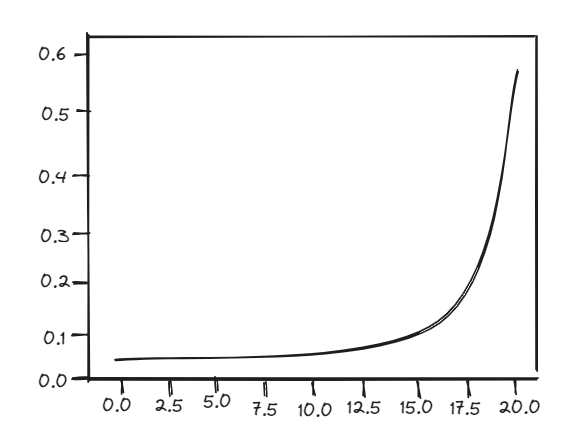
Def sig(x):

return 1/(1+np.exp(-x))

****

(الشكل 2-5 : تابع sigmoid)

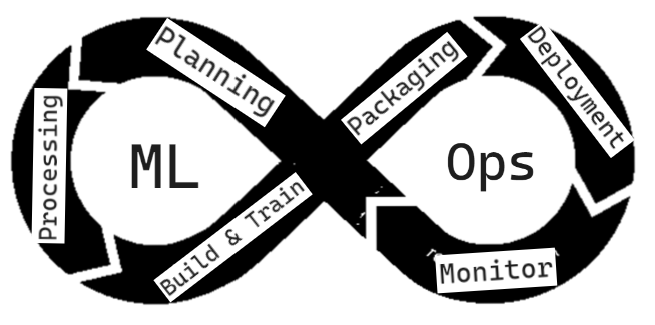
### **2-4-3 softmax**

تستخدم وظيفة softmax كوظيفة تنشيط في طبقة الخرج لنماذج الشبكة العصبية التي تتنبأ بتوزيع احتمالي متعدد الحدود . أي هي دالة رياضية تقوم بتحويل متجه الأرقام إلى متجه للاحتمالات , حيث تتناسب احتمالات كل قيمة مع المقياس النسبي لكل قيمة في المتجه [2].

(الشكل 2-6 : تابع softmax )

# الفصل الثالث: الجزء العملي

تحدثنا في فقرة منهجية البحث عن عملية التشكيل و التي تبدأ بعملية المعالجة وتنتهي بمرحلة الحصول على model قادر على تشكيل الجملة العربية ولكن في هذا الفصل سوف نتحدث عن مفهوم أشمل وهو MLOps cycle و هي باختصار دورة حياة نموذج التعلم الآلي من مرحلة التخطيط plan إلى مرحلة الانتاج deployment والذي يوضحه الشكل التالي



)الشكل3-1)

## **3-1 التخطيط planning**

في هذه المرحلة تم العمل على :

* تقسيم أعضاء الفريق إلى عدة فرق:

1. فريق البحث (Research Team) المسؤول عن وضع معمارية النماذج و التحكم بمعاملات التدريب وتدريب النماذج ومراقبة تدريب النماذج.
2. فريق (MLOps Team) المسؤول عن التعامل مع جميع الفرق وتأمين متطلباتهم من توفير أدوات لجعل مرحلة التدريب automation و مراقبة النماذج monitor والعمل على إيصال النماذج إلى مرحلة production.
3. فريق الاختبار (Testing Team) المسؤول عن اختبار النماذج والتأكد من كفاءتها.

* تقسيم بيئة العمل إلى بيئتين:

1. بيئة التطوير التدريب (Dev/Train)، وهنا سيتم تقسيم هذه المرحلة إلى بيئتين:

بيئة التدريب train وهي تتم على سيرفر التدريب الذي يحوي GPU وعلى اعتبار لا يوجد سيرفر مستقل لدينا, تم اعتبار سيرفر colab هو السيرفر الخاص بعملية التدريب.

بيئة التطوير dev وهي تتم على سيرفر مستقل (أجهزة الحاسب الخاص بنا) حيث يتم تحويل الأوزان الناتجة عن مرحلة التدريب إلى (Web Service).

1. بيئة الانتاج (Prod) حيث يتم فيها نشر (Web Service) على سيرفر production واختبارها وتحسينها.

* التركيز على معمارية المشروع وفقا لأفضل الممارسات في السوق الحالية, واخترنا التقسيم التالي تبعنا لاحتياجات المشروع:

1. ملف fleet\_dataset.sh: وهو linux script يستخدم لبناء ملفين من الداتا الاصلية لعملية التدريب والتقييم.
2. dictionaries.py: يحتوي على الترميزات المختلفة بالاحرف العربية وعلامات الترقيم والتشكيل.
3. ملف preprocess.py: يحتوي على التوابع التي تقوم بمعالجة الداتا.
4. ملف generator.py: يقوم ببناء batches بعد استخدام كافة عمليات المعالجة على الداتا وذلك لتصبح عملية التدريب أكثر كفاءه
5. ملف models.py يحتوي على معمارية النموذج المستتخدم وأيضا يمكن تحقيق أي معمارية للشبكة في هذا الملف.
6. ملف train.yaml: يحتوي على إعدادات الخاصة بعملية التدريب (عقل المشروع) مثل نوع معمارية النموذج ومكان وجود الداتا الخاصة بالتشكيل ومعدل التعلم وعدد العينات وbatch size و عدد epochs ونوع الجهاز الذي سيتم التدريب عليه GPU or CPU وعدد العصبونات في الطبقات الخاصة بالنموذج المستخدم.
7. ملف train.py: هو الملف الذي تتم فيه عملية التدريب بعد تحديد الإعدادات في train.yaml من معالجة الداتا إلى توفير أوزان خاصة بعملية المعالجة وأوزان خاصة بعملية التدريب.
8. ملف :app.py يحتوى واجهة بسيطة لتجريب النماذج الجديدة باستخدام streamlit.
9. ملف telegram.py: يحتوي على تابع نستخدمه في ارسال message الى تلغرام
10. ملف :requirements.txt يحوى على python packages الخاصة بالمشروع مع رقم الأصدار لكل packages.
11. مجلد :data يحوي على الداتا المستخدمة في المشروع والتي تضم ملف التدريب والمعالجة
12. مجلد :outputs يحتوي على الأوزان الخاصة والتدريب
13. مجلد papers\_research: يحتوي على الأوراق البحثية التي تم اعتمادها في المشروع

إلى هنا نكون قد انتهينا من مرحلة التخطيط (planning) وسوف ننتقل إلى المرحلة الثانية الخاصة بتعريف الداتا ومعالجتها Data and Processing)).

## **3-2 الداتا ومعالجتها Data and Processing**

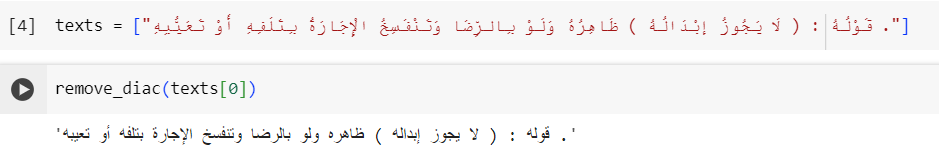
في هذه المرحلة تم العمل على :

* التعرف على الداتا المستخدمة في عملية التدريب والتي توجد في المجلد data حيث يوجد 3 ملفات ملف يحوي كامل الداتا وهو dataset.txt ويحتوي على 53 ألف جملة مشكلة وملف خاص بعملية التدريب train.txt واخر خاص بعملية التقييم valid.txt يتم التحكم بعدد الجمل في كل من الملفين باستخدام سكربت fleet\_dataset.sh.
* في ملف dictionaries.py الخاص بعملية ترميز الاحرف وعلامات الترقيم والتشكيل تم تعريف:

1. arabic\_characters: متحول يحتوي جميع أحرف اللغة العربية.
2. diacritics\_list: قائمة تحتوي علامات التشكيل فتحة، ضمة، كسرة، سكون، شدة، علامات التنوين.
3. classes: قاموس المفاتيح فيه تمثل حركات التشكيل + الفراغ + رمز خاص بالحشو <PAD> + رمز خاص ببداية الجملة <SOS> + رمز خاص بنهاية الجملة <EOS> + رمز خاص بحركة غير معروفة <N/A>، في حين تضم قيم القاموس ارقام خاصة بكل رمز لأن نماذج الذكاء الصنعي تستطيع التعامل مع الارقام ولا تستطيع فهم الرمز المجرد ويستخدم هذا القاموس في عملية المعالجة والتدريب.
4. قاموس revers\_classes: عكس القاموس السابق المفاتيح تضم الارقام و القيم تضم الحركات التي تحدثنا عنها في القاموس classes ويستخدم هذا القاموس في عملية تحويل الجملة العربية إلى جملة مشكلة حيث أن خرج النموذج ارقام تمثل حركات التشكيل والفراغ والرموز UNK , PAD, SOS, EOS و يتم تحويل هذه الارقام الى مقابلاتها ثم يتم دمجها مع الجملة العربية وتصبح جملة مشكلة.
5. قاموس symbol: قاموس مفاتيحة تمثل الاحرف والارقام و علامات الترقيم وبعض الرموز التي تظهر في النص العربي مثل الاقواس الكبيرة والصغيرة ورمز البداية والنهاية والحشو والغير معرف وتمثل قيم القاموس ارقام مقابله لهذه الرموز.

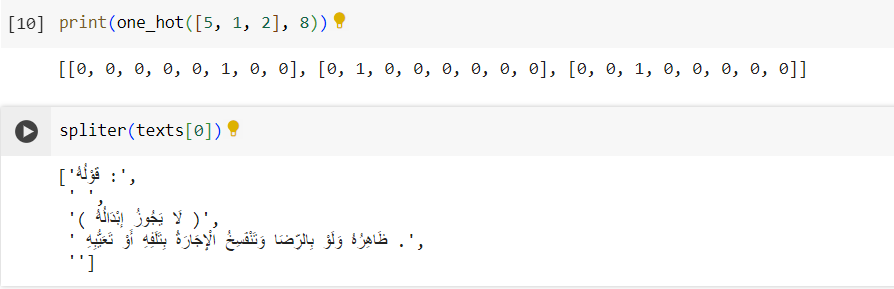
* في ملف preprocess.py الخاص بمعالجة الداتا تم تعريف التالي:

1. التابع read\_data يأخذ هذا التابع مسار الداتا كبارمتر ثم يقوم بقراءة كل سطر في كل من ملف train.txt, valid.txt وتخزين الجمل في متحول train, valid على الترتيب ثم يقوم بطباعة عدد الاسطر الخاصة بالتدريب والتقييم ويقوم بإعادة train, valid
2. التابع remove\_diac يقوم بحذف حركات التشكيل عن الجملة المشكلة اعتماد على متحول diacritics\_list الذي يضم حركات التشكيل (ضمة، سكون، فتحة، ..) كما في الشكل(كذا)



الشكل(كذا)

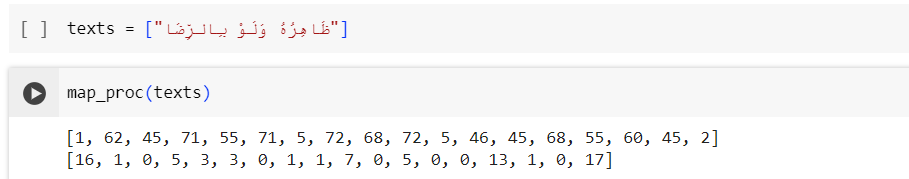
1. التابع one\_hot: يقوم هذا التابع بتحويل كل رقم في المصفوفة الى قائمة تحوي 0 و1 بطول ثابت حيث تكون هذه القائمة جميع قيمها اصفار في حين دليل الرقم يكون قيمته 1 كما في الشكل(كذا)
2. التابع spliter يقوم بتقسيم الجملة الى عدة جمل اعتمادة على علامات الترقيم والاقواس والفواصل بأشكالها كما في الشكل(كذا)



الشكل(كذا)

1. التابع split\_data: يقوم بالمرور على عدة جمل ثم تطبيق التابع spliter للحصول على عدة جمل ثم يقوم بأخذ الجمل التي عدد محارفها اكبر من 0 واصغر من 500 طبعا يمكن القيام بتغيير هذه القيمة الى قيمة اكبر في حال امتلاك موارد اكبر من ram, cpu
2. التابع map\_proc يقوم بإنشاء دخل وخرج النموذج من الجملة المشكلة حيث يكون input هو الاحرف العربية والارقام وعلامات الترقيم والاقواس في حين output هو حركات التشكيل مع اخذ بعين الاعتبار الرموز الخاصة التي تضاف الى الاثنين معنا مثل بداية ونهاية الجملة والحشو (UNK , PAD, SOS, EOS) حيث بشكل اساسي يقوم هذا التابع بعدة خطوات للوصول الى هذا الشكل يقوم بتعريف قائمتين للدخل والخرج ثم يقوم بالمرور على الجمل المشكلة سطر سطر وفي كل سطر يقوم بتعيين قائمة x من اجل الدخل وقائمة y من أجل الخرج ويقوم بوضع رمز بداية الجملة <SOS> في كل منهما **كقيمة** اعتمادا على القاموسين classes, symbol ثم يتحقق في البداية اذا كان المحرف موجود في diacritics\_list يقوم بتجاهله واذا كان رمز(حرف لغة عربية، ارقام، علامات الترقيم، اقواس) يقوم بإضافة ترميزه الرقمي إلى x وبعد ذلك يتحقق اذا لم يكن هذا الرمز حرف لغة عربية فيقوم بإضافة الفراغ كقيمة الى y لأنه يهمنا أن نقوم بأخذ احرف اللغة العربية وحركاتها في حين الارقام وعلامات الترقيم والاقواس لا يوجد عليها حركات في حال كان هذا الرمز هو حرف لغة عربية نقوم بأخذ الحركة الخاصة به ومن الممكن أن لا يكون على الحرف حركة فيكون فراغ لكن في حال وجود حركة يتم أخذها، يوجد مناقشة حالتين هنا هي أن يكون هناك حركة أو حركتين بعد الحرف اذا كان يوجد حركة يتم أخذها مباشرةً وفي حال وجودة حركة اخرى يتم التحقق في البداية هل هذه الحركة هي فتحه + فتحه مثلا في حال ذلك لا يتم اخذها لانه لا يوجد في اللغة العربية حالة فتحه + فتحه أو ضمه + ضمه وفي حال كانت الحركتين احد حالات classes مثل الشدة + الكسرة يتم أخذها وبعد ذلك يتم إضافة الترميز **الرقمي** إلى y بعدها يتم التحقق من تطابق طول x مع y في حال عدم التطابق يتم تجاهل هذه الجمله  
   في حال التطابق يتم إضافة **قيمة** رمز النهاية <EOS> الى كل منها ثم يتم ترميز قيم y الى اصفار ووحدات بحيث يكون طول كل شعاع بقيمة عدد classes وبعد ذلك يتم اضافة x إلى المصفوفة التي تحوي كامل الدخل وy الى المصفوفة التي تحوي كامل الخرج، حيث

كما في الشكل(كذا) عند طباعة جملة واحدة في المصفوفة **الاولى** من الخرج التي تمثل الدخل يوجد 1 في بدايتها دلالة على بداية الجمله ثم ارقام تمثل الرموز (احرف عربية والارقام والاقواس) ثم في نهاية المصفوفة 2 دلالة على نهاية الجملة وكذلك في المصفوفة **الثانية** التيتمثل الخرج يوجد 16 كبداية ثم ارقام من 0 الى 17 تمثل حركات التشكيل ثم الرقم 17 في نهاية المصفوفة الذي يمثل نهاية الحركات، تم طباعة المصفوفة بدون تطبيق one\_hot لتكون الفكرة أوضح

**الشكل(كذا)**

* في ملف generator.py الخاص ببناء batchs بعد تطبيق عمليات المعالجة عليها تم تعريف التالي:

1. تم تعريف custom class من الصف Sequence في الحالة الطبيعية لا يتم استخدام هذا الصف عند اجراء عملية التدريب في keras لأنه يكون لدينا دخل وخرج واضح ولكن في مشكلتنا يوجد لدينا فقط جملة مشكلة ونريد استخراج منها دخل وخرج وذلك بالنسبة لجميع الجمل المشكلة وكذلك تقسميها الى batchs لإجراء عملية التدريب بكفاءة لذلك قمنا بوراثة الصف Sequence تحت الاسم Generator وقمنا بتمرير عدد الاسطر وحجم batch وعرفنا التابع len ضمن هذا الصف لحساب عدد batchs وتم تعريف التابع getitem الذي يساعدنا في الحصول على كل batch بحسب index الخاص به يقوم في البداية بأخذ الجمل الخاصة ب batch ما وليكن batch رقم 2 مثلا يتم أخذ ارقام الاسطر من (2 \* حجم (batch إلى ((2 + 1) \* حجم (batch ثم يتم تطبيق تابع map\_proc الذي تحدثنا عنه في ملف preprocess.py الذي يقوم بإنشاء الدخل والخرج ثم يتم حساب اطول دخل وخرج في كامل الbatch وذلك ليتم إضافة الحشو بحيث يكون جميع العينات لها نفس الطول حيث يتم المرور على الدخل وإضافة قيمة الحشو <PAD> بحسب المتبقي من السلسلة للوصول للطول الاعظم، كذلك الحال في الخرج يتم إضافة قيمة الحشو بحسب المتبقي من السلسلة للوصول للطول الاعظم مع تطبيق one\_hot لكي يناسب الخرج y ثم يتم إعادة X\_batch, Y\_batch الذي تمثل كل منهما دخل وخرج كامل batch.
2. التابع full\_process هو مسار لجميع العمليات السابقة في ملفي preprocess.py, generator.py يقوم بأخذ مصفوفتين تمثل الاولى train الجمل المشكلة الخاصة بعملية التدريب وتمثل الثانية valid الجمل الخاصة بعملية التقييم ويوجد ايضا batch\_size الذي يمثل حجم batch وكذلك shuffle متحول بولياني يأخذ القيمة 1 لجعل الداتا عشوائية وصفر لإبقائها على حالها، يقوم التابع بتقسيم داتا التدريب والتقييم لإجراء عمليات تقسيم على الجمل الطويلة والاحتفاظ بها ثم يقوم بجعل الداتا عشوائية في حال تحقيق shuffle القيمة 1 بعد ذلك يتم تطبيق Generator على التدريب والتقييم لتوليد جميع batchs التي ستستخدم في عملية التدريب والتقييم.

إلى هنا نكون قد انتهينا من مرحلة الداتا ومعالجتها Data and Processing)) وننتقل إلى المرحلة التالية التي تخص بناء معمارية النموذج وإجراء عملية التدريب والتقييم.

## **3-3 بناء وتدريب النماذج Build and Train Models**

في هذه المرحلة تم العمل على :

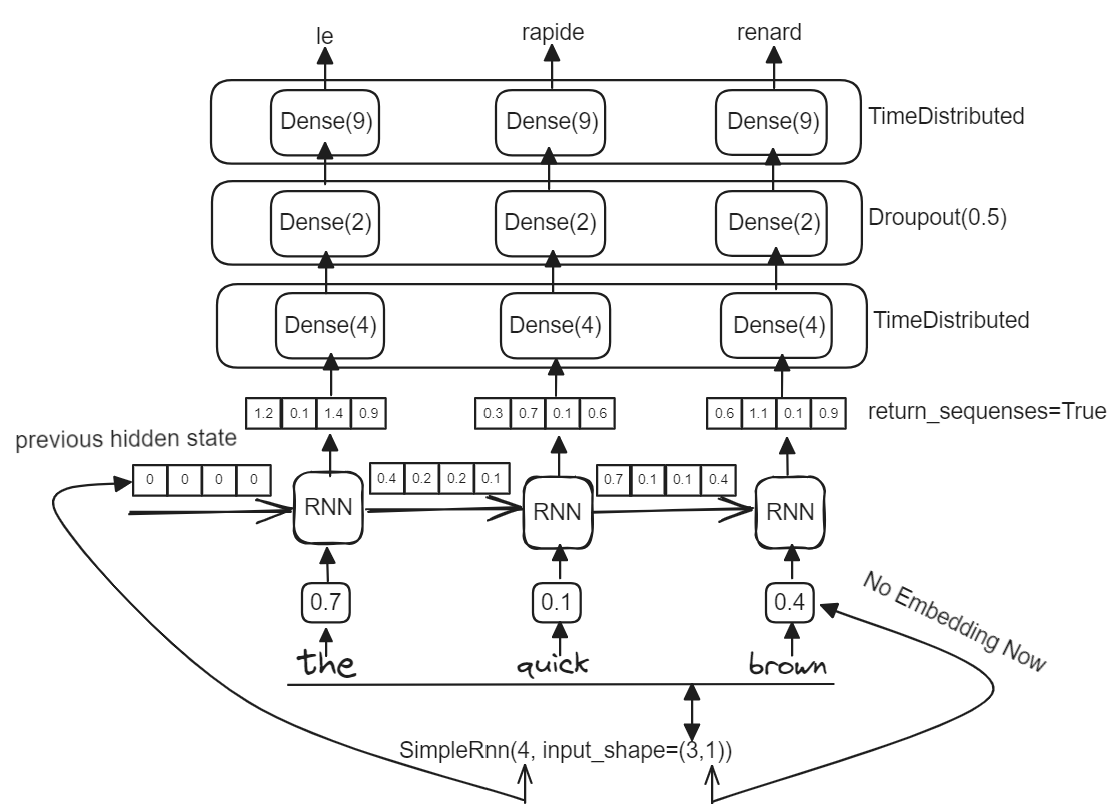
1. في ملف train.yaml كما تحدثنا في مرحلة التخطيط يحوي على إعدادات الخاصة بعملية التدريب (عقل المشروع) والذي يستطيع الباحث أو المطور التحكم بكامل عملية التدريب من خلاله حيث تم تعريف المتحولات التالية :
2. data\_url: هو متحول string يمثل مسار الملف الذي يحوي ملفي train.txt و valid.txt الخاصين بعملية التدريب والتقييم
3. type\_model: هو متحول string يحوي على اسم معمارية الشبكة التي ستقوم عملية  
   التدريب عليها حيث يتم في كل عملية تدريب تغيير قيمة هذا الاسم لتفريق النماذج عن بعضها حيث نقوم باستخدام اسم مختلف في حال زيادة عدد عينات التدريب
4. learning\_rate: هو متحول float يمثل معدل التعلم يفضل أن تكون قيمته بين 0 و 1.
5. batch\_size: هو متحول integer يمثل عدد العينات التي يتم إرسالها في كل دفعة خلال عملية التدريب يفضل أن يكون صغير لكي لتجنب استخدام الذاكرة بشكل كبير.
6. shuffle: متحول بولياني يأخذ القيمة 1 لجعل الداتا عشوائية و 0 لإبقائها على حالها
7. epochs: هو متحول integer يمثل عدد مرات تدريب النموذج على مجموعة العينات.
8. type\_device: هو متحول string لتحديد على ماذا تتم عملية التدريب سواء على CPU أو GPU وذلك بحسب التوافرية لدى الباحث أو المطور ويتم استخدام نفس الاسماء لتحديد نوع الجهاز.
9. n\_neurons\_embedding: هو متحول integer يمثل طول شعاع الرمز الذي نريد الحصول عليه في طبقة (Character Embedding).
10. n\_neuros\_lstm: هو متحول integer يمثل عدد الحالات المخفية state hidden التي تعطيها كل طبقة (LSTM) للطبقة التالية .
11. n\_neuros\_ timedistributed: هو متحول integer يمثل عدد العصبونات في طبقة (Time Distributed).
12. في ملف models.py كما تحدثنا في مرحلة التخطيط أن هذا الملف سوف يستخدم لتحقيق أي معمارية للشبكة, لذلك تم العمل على:

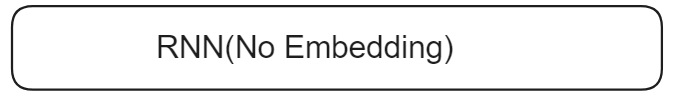
* بناء تابع motor يأخذ هذا التابع المتحولات الخاصة ببناء معمارية الشبكة مثل**:**

1. n\_neurons\_embedding: هو متحول integer يمثل طول شعاع الرمز الذي نريد الحصول عليه في طبقة (Character Embedding).
2. n\_neuros\_lstm: هو متحول integer يمثل عدد الحالات المخفية state hidden التي تعطيها كل طبقة (LSTM) للطبقة التالية .
3. n\_neuros\_ timedistributed: هو متحول integer يمثل عدد العصبونات في طبقة (Time Distributed).
4. learning\_rate: هو متحول float يمثل معدل التعلم يفضل أن تكون قيمته بين 0 و 1.

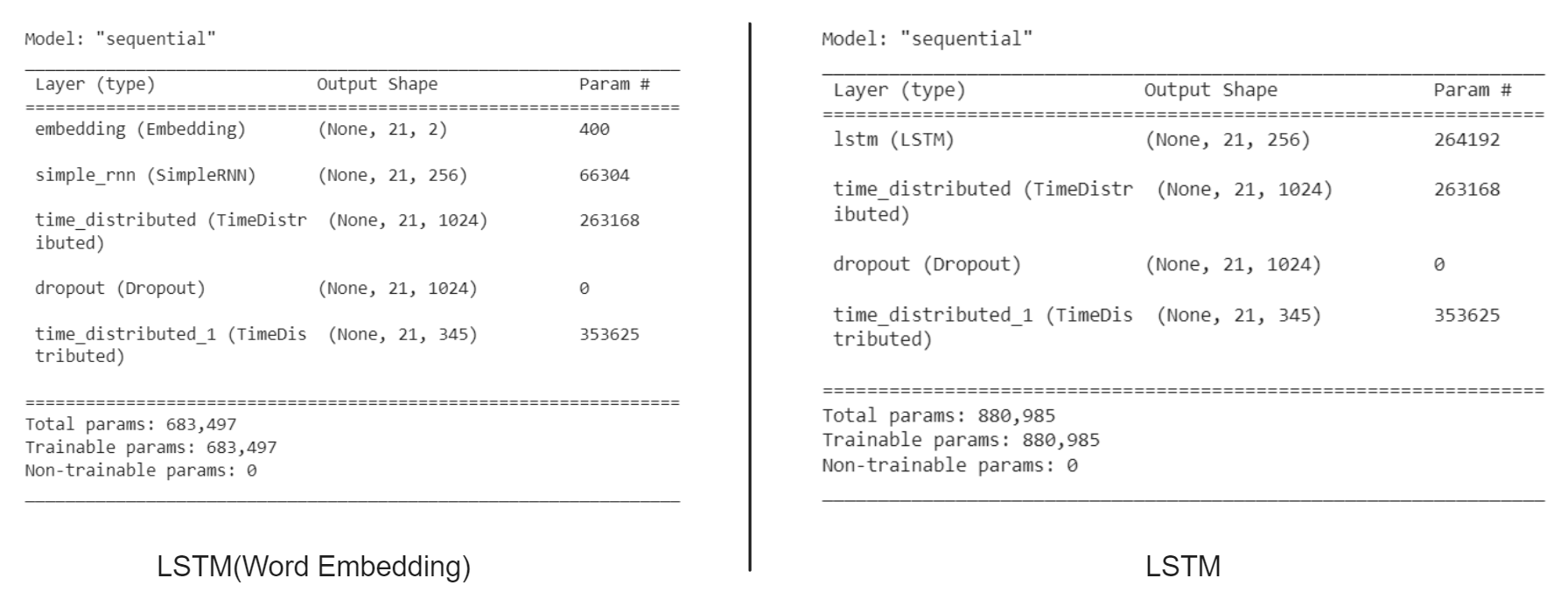
* في التابع السابق تم تحقيق معمارية الشبكة بالاعتماد على تكديس مجموعة من طبقات LSTM وفق الاتي:

1. طبقة Input
2. طبقة Embedding تقوم بتمثيل كل حرف في مجموعة الرموز التي تأتي في طبقة الدخل مثلما تحدثنا سابقا (حروف، ارقام، علامات ترقيم ..) ويكون لكل حرف طول شعاع يتم تحديده من خلال n\_neurons\_embedding تفيد هذه الطبقة في تحديد العلاقة الدلالية بين هذه الرموز.
3. طبقة LSTM ثنائية الاتجاه حيث تأخذ n\_neuros\_lstm الذي يمثل عدد الحالات المخفية بين الطبقات و input\_shape ثنائية الجزء الأول يضم عدد الكلمات في الجملة ومثل أيضا time steps عدد العصبونات الموجود في طبقة lstm والتي سوف يهتم كل عصبون فيها بكلمة واحدة من الجملة والجزء الأخر من الثنائية يمثل طول شعاع الكلمة و تم وضع return\_sequences على القيمة true لكي يقوم كل عصبون بإرجاع hidden states الخاص به, الشكل (3-5) يوضح آلية الطبقة.
4. طبقة TimeDistributed تكمن فائدة هذه الطبقة في أنها لو لم تكن موجوده كان سيتم فقط معالجة hidden state لأخر عصبون وهذا سيئ لأننا نريد تحديد كلمة الخرج باللغة الفرنسية بناء على الكلمة الانكليزية الحالية التي وصلنا لها وليس بناء على كامل الجملة (بكلمات أخرى لو كانت المهمة تصنيف الجمل إلى خبر كاذب وحقيقي أي 0 أو 1 كان باستطاعتنا استخدام Dense على أخرhidden state خارج من الطبقة الأخيرة لأن المشكلة هي Many-To-One ولكن في حالتنا باستخدام TimeDistributed طبقنا Dense على كل hidden state لأن المشكلةMany-To-Many) ) هذه الطبقة تأخذ n\_neurons\_ timedistributed كعدد عصبونات لكل dense, الشكل (3-3) يوضح آلية الطبقة.
5. طبقة Dropout مع قيمة 0.5لمنع حصول over fitting في الشبكة, الشكل (3-3) يوضح آلية الطبقة.
6. طبقة TimeDistributed ولكن نقوم بأعطاءه dict\_fr\_size (عدد كلمات اللغة الفرنسية في الداتا) كعدد عصبونات لكل dense مع تابع التفعيل softmax لإعطاء مصفوفة احتمالات للكلمات في كل dense وبعدها يتم اختيار الكلمة اعتماد على التابع ids\_to\_words المعرف مسبقا في مرحلة المعالجة والذي يقوم بتحويل كل مصفوفة احتمالات بعد اختيار أكبرها باستخدام argmax إلى الكلمة الفرنسية المقابلة للرقم اعتماد على القاموس , الشكل (3-3) يوضح آلية الطبقة.



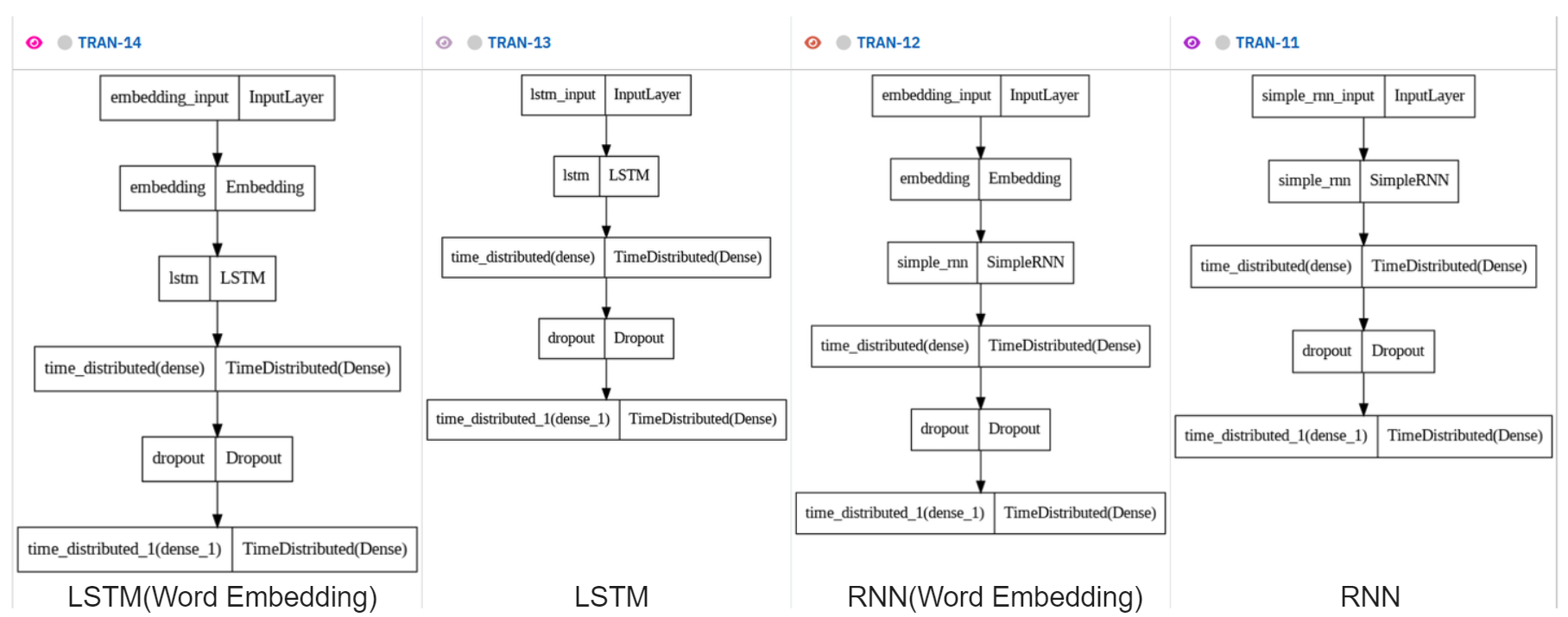


)الشكل 3-3)

)الشكل3-7)

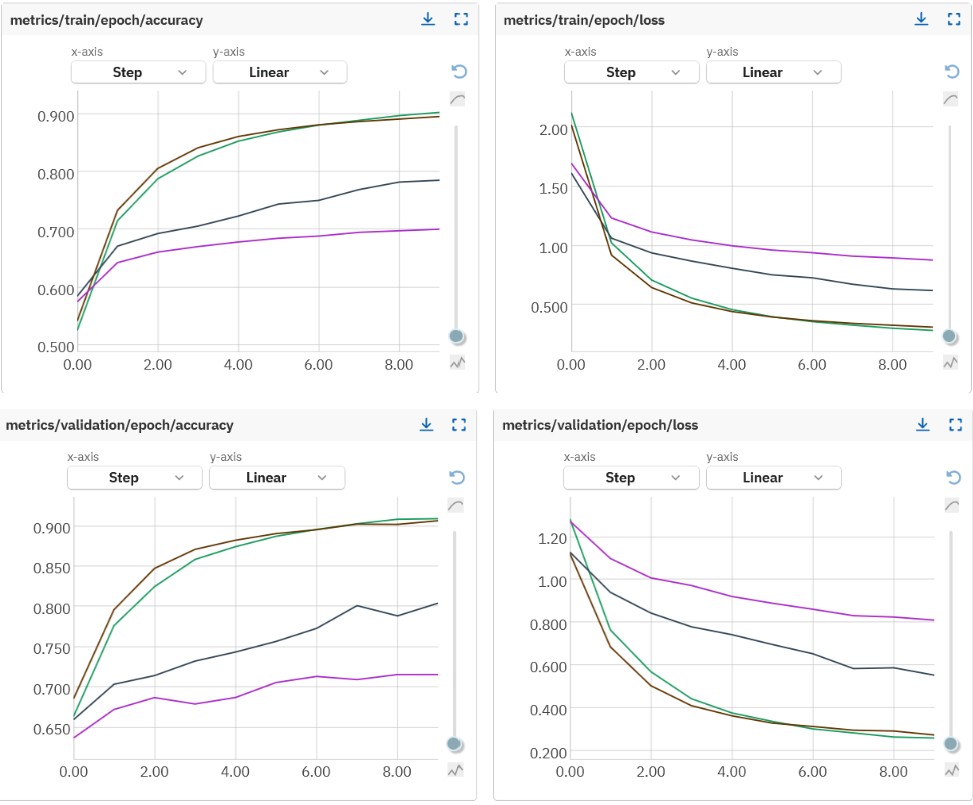
* في التابع السابق تم استخدامcompile لتعريف توابع loss, metrics, optimizer ويقوم بعدها التابع motor بإرجاع model بعد كامل التجهيز, وبذلك نكون قد انتهينا من الحديث عن ملف models.py.

1. في ملف train.py كما تحدثنا في مرحلة التخطيط أن هذا الملف سوف يستخدم في عملية التدريب, لذلك تم بناء التابع train الذي يأخذ متحول config\_path وهو مسار الملف train.yaml الذي يضم إعدادات عملية التدريب, بعد تعريف الإعدادات في ملف train.yaml وتعريف المعمارية في ملف models.py وتوابع المعالجة في ملف preprocess.py, يقوم التابع train بالتالي
2. قراءة جميع المتحولات في ملف train.yaml باستخدام المكتبة yamlوإسنادها إلى متحولات مطابقه لها بالاسم منعا من الضياع.
3. بدء مرحلة معالجة الداتا من خلال التوابع التي عرفناها في الملف preprocess.py, وذلك بقراءة داتا التدريب والتقييم من خلال ملفي train.txt , valid.txt باستخدام read\_data واستخدام التابع full\_process لتجهيز الدخل والخرج من خلال الجمل المشكلة وجعلها على شكل دفعات لتسهيل عملية التدريب
4. الحصول على معمارية الشبكة باستخدام التابع motor عبر type\_model كما ذكرنا وكذلك تجهيز طبقات المعمارية المختارة باستخدام متحولات Structural Models
5. أما الآن سوف نتعرف على النوع الأول من monitoring في مشروعنا, حيث كان يجب عنه الحديث في مرحلة خاصة تحت مسمى **Monitior**, ولكن منعا لتشت تناسق الأفكار الخاصة بالمشروع, يستخدم هذا النوع لجعل حياة Research أسهل وتقع عاتق توفير هذه المهمة على فريق MLOps, فهو يوفر طريقة لمراقبة عدة نماذج نقوم بتدريبها على التفرغ في مرحلة ومراقبة مخططات الخطأ و الكفاءة وكذلك مراقبة حالة CPU وGPU و Memory وبالتالي هذا يساعدنا على إطلاق الحكم على استخدام النموذج الأفضل بين النماذج للقيام بعملية تطويره لاحقا, لذلك سوف نقوم أداة Neptune.ai مع keras, قمنا باستخدام التابع init\_run في neptune لتجهيز المشروع باستخدام project=" ammar.mlops/arabic-loneha" ويوجد متحول أخر name يأخذ قيمة type\_model الحالية تكمن فائدة هذا المتحول في تسمية كل عملية تدريب، هذا التنظيم يساعد في التتبع الجيد لعمل كل model ضمن المشروع الواحد والمتحول الاخير api\_token يمكن اعتباره بمثابة باسورد تعطي سماحية لمستخدمها من الولوج إلى موقع neptune وتخزين نتائج تدريب النماذج عليه, بعد ذلك تم ربط neptune مع keras تعريف callbacks حيث يقوم هذا المتحول بكتابة كل عملية التدريب من قيم الخطأ و الكفاءة ورسم مخططات في موقع neptune.
6. كما ذكرنا سابقا في ملف telegram.py تم تعريف تابع لارسال الرسائل الى قناة لنا على التلغرام بحيث يتابع كامل الفريق عملية التدريب من خلال رابط neptune يتم ارساله يحتوي عملية التدريب والمراقبة على النموذج الحالي مع توصيف بسيط لعدد عينات التدريب والتقييم ومعدل التعلم وحجم batch
7. بعد ذلك يتم استخدام GPU أو CPU عبر المتحول type\_device, وبداية عملية التدريب عبر التابع fit المعرف في keras والذي يأخذ متحولات تم شرحها بالتفصيل سابقا خلال المراحل السابقة مثل train\_generator ,valid\_generator التي تمثل دفعات التقييم والتدريب واخير callbacks الذي عرفناه مسبقا لتحصل عملية الربط بين neptune وkeras
8. تم استخدام التابع save في keras لحفظ model وتم استخدام type\_model ليختلف اسم النموذج المدرب وبالتالي يسهل التعرف عليه واستخدامه.
9. تم كتابة train.py على شكل script وهذا ما يناسب بيئة التدريب عند العمل على سيرفر مستقل أو colab بحيث يكون هذا الملف قابل للعمل في أي مكان يوضح فيه ويتم تنفيذه بالشكل التالي وتسمح لنا المكتبة argparse بإعطاءه config\_path كمتحول خارجي والتنفيذ يكون على الشكل التالي:  
   
10. بعد التنفيذ دخلنا إلى موقع neptune.ai وقمنا بمراقبة عمل النماذج على التوازي وكذلك مراقبة مخططات الخطأ والكفاءة وحصلنا على النتائج التالية, في البداية نستعرض مخططات المعماريات على التوازي ويوضحها الشكل(3-8)



)الشكل3-8)

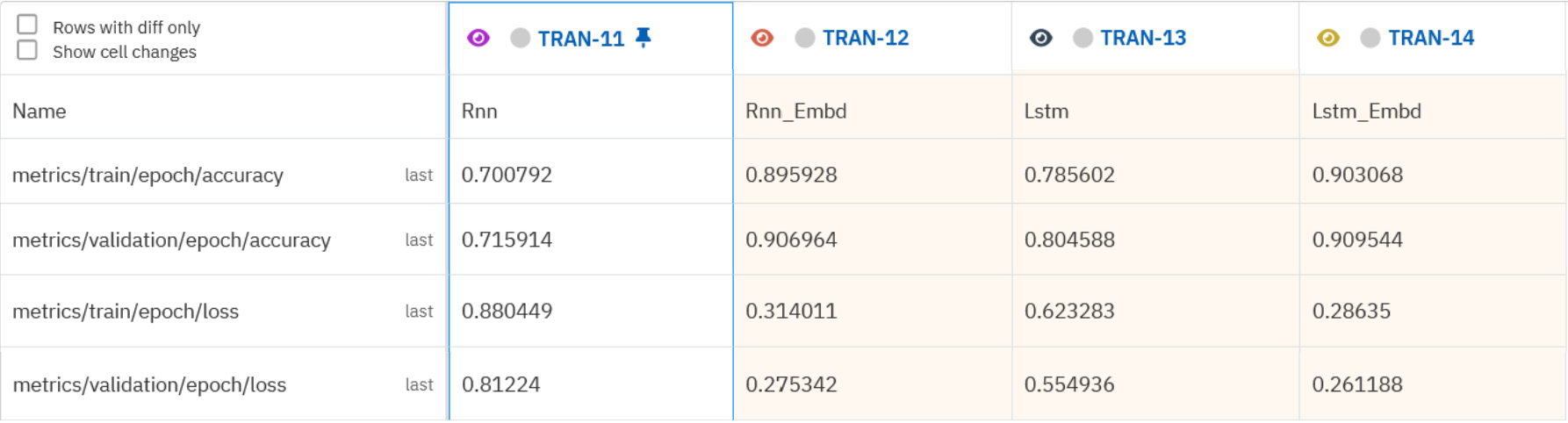
وكذلك مخططات loss و accuracy في كل epochs بالنسبة للأربع معماريات حيث نلاحظ أن بالنسبة لمخطط accuracy في الشكل (كذا) على اليسار سواء train أو validation أن الخطين البيانين في أعلى كل مخطط يحققان accuracy أعلى خلال زيادة قيمة epochs فيما الخطين البيانين في أسفل كل مخطط يحققان accuracy أقل وهذا يدل على أن النماذج تتعلم أفضل سواء LSTM أو RNN عند استخدام طبقة Word Embedding)) وكذلك مخطط loss في الشكل(3-9) على اليسار سواء train أو validation تعطي نفس الحقيقة حيث الخطين في أسفل كل مخطط يحققان loss أقل عند تقدم epochs وبالتالي طبقة Word Embedding)) حققت فرق ملموس وواضح بين المعماريات التي تستخدم هذه الطبقة والمعماريات التي لا تستخدمها



)الشكل3-9)

### **3-3-1 النتائج**

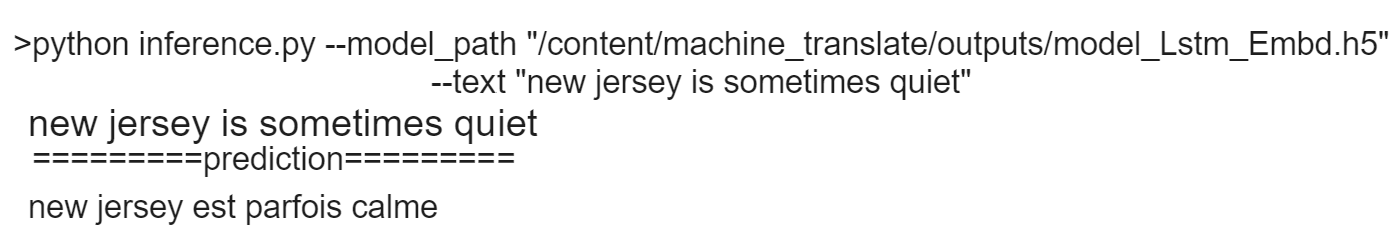
من خلال عملية الربط بين neptune وkeras نستطيع الحصول على جدول لنتائج المقارنة بين الأربع معماريات الشكل(3-10), لن نستعرض كامل الجدول سوف نستعرض ما يهمنا من عملية التدريب في الأربع معماريات وأي معمارية أفضل بالنسبة للنتائج الحالية , حيث على الترتيب الجدول على مستوى الأعمدة كل عمود له اسم يمثل رقم فريد للمعمارية المستخدمة, على مستوى الاسطر في السطر الأول اسم النموذج المدرب وبعدها قيمة الكفاءة بالنسبة ل train وvalidation وقيمة الخطأ بالنسبة train وvalidation نلاحظ نفس الحقيقة التي وجدناها في المخططات نجدها في الجدول الشكل(كذا), المعماريات التي تستخدم Word Embedding)) أي العمود ذات الاسم tran12 و tran14أفضل في عملية التعلم وبالتالي سوف تكون أفضل في عملية التحول الكامل



)الشكل3-10)

بعد ذلك يتم اختبار النموذج بعد عملية التدريب ويتم ذلك عن طريق الملف inference.py

حيث يقوم هذا الملف بتحويل جملة من اللغة الانكليزية إلى الفرنسية بعد اختيار احد النماذج الاربعة التي تم تدريبها وإعطاءه رابط النموذج وكذلك تمرير الجملة المراد ترجمتها, فيقوم بتحميل الاوزان الموجودة في مجلد ,outputs طبعا مكتبة argparse هي التي تسمح لنا بهذا النوع من العمليات, بعد تنفيذ الملف نلاحظ الخرج الذي يمثل الجملة المراد ترجمتها والترجمة المتوقعة.



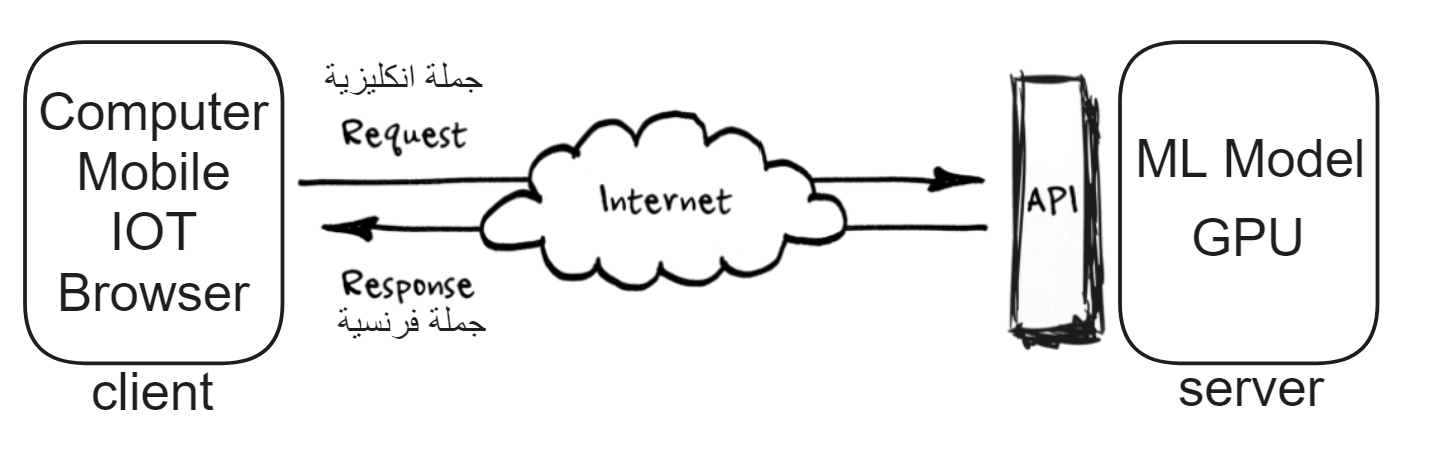
إلى هنا نكون قد انتهينا من مرحلة بناء وتدريب النماذج Build and Train Models وسوف ننتقل إلى المرحلة الرابعة الخاصة بمرحلة التعبئة (packaging)

## **3-4 التعبئة packaging**

في هذه المرحلة يتم تجميع جميع المكونات والتبعيات اللازمة لتشغيل تطبيق أو نموذج ذكاء صناعي في حزمة مستقلة. وفي حالتنا سوف نقوم باستخدام fastapi وذلك لتعريف API تقوم بتحميل نموذج وتنفيذ الترجمة واستخدام Docker لتعريف البيئة المعزولة التي ستحتوي على جميع التبعيات و الأدوات اللازمة لتشغيل النموذج, وبالتالي لدينا مرحلتين هما:

1. **مرحلة تحويل ML Model إلى API:**

ما هوAPI؟ هو مجموعة من البرمجيات والقواعد والبروتوكولات التي تسمح للتطبيقات بالتواصل والتفاعل مع بعضها البعض, يعتبر API نقطة اتصال بين تطبيقين مختلفين أو بين تطبيق وخدمة خارجية. يستخدم التطبيق المصدر (client) API للوصول إلى وظائف أو بيانات تقدمها التطبيق المستهدف (server), الشكل(3-11) يشرح الآلية حيث يتم إرسال الجملة الانكليزية بصيغة json مثلا عبر الأنترنت إلى API حيث يقوم بمعالجة الطلب على server ومن ثم يقوم API بإعادة الجملة الفرنسية بصيغة json أيضا عبر الأنترنت إلى client من الممكن أن يكون client أي جهاز يملك الحد الأدنى لوصف ألة تعمل بالكهرباء وتملك معالج وذاكرة ويمكنها الاتصال بالانترنت مثل computer, mobile, sensor..etc.

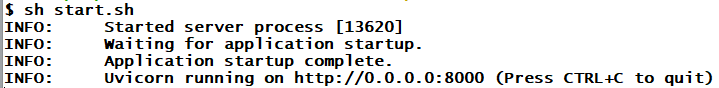


)الشكل 3-11)

بعد شرح API ننتقل الأن إلى تقديم شرح عن دمج ml model مع fastapi وهو كما ذكرنا سابقا إطار عمل لبناء API ويتم ذلك في مجلد fastapi والذي يحوي:

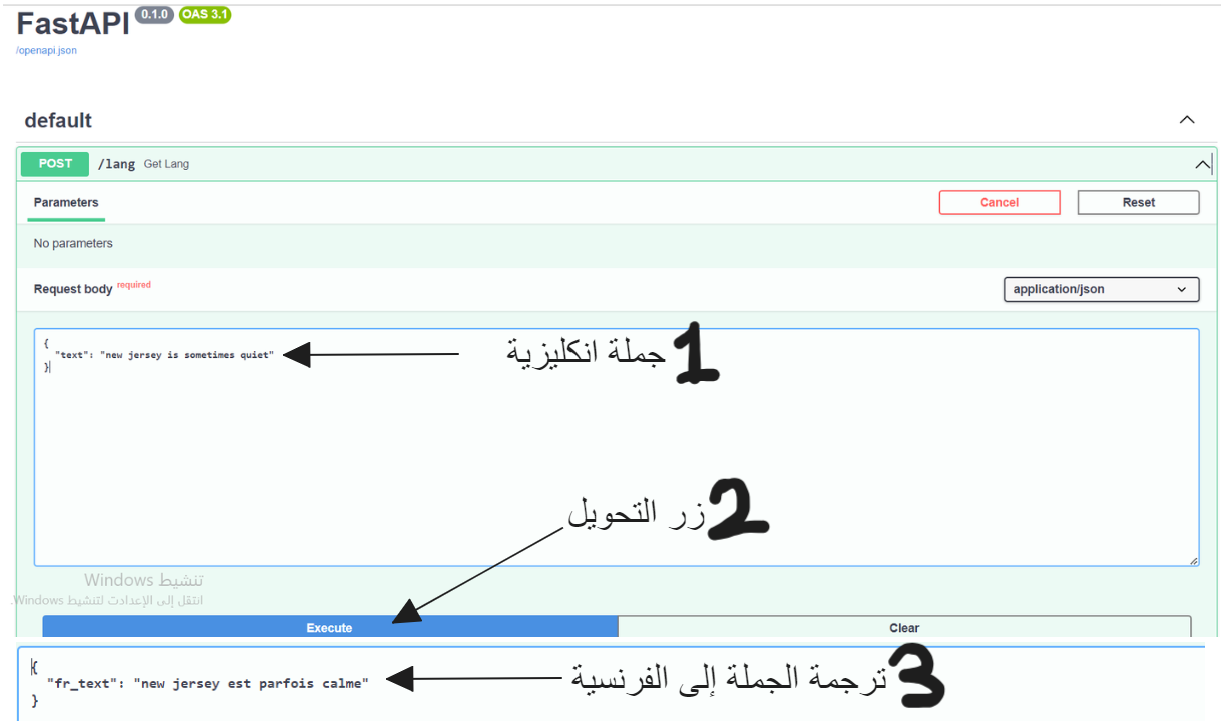
1. المجلد الأول www يملك ملف init\_fastapi.py يقوم بأخذ object من FastAPI الذي يمثل تطبيق منفصل وذلك لتعريف المسارات والوظائف المرتبطة بهذا التطبيق حيث في تطبيقنا لدينا وظيفة واحدة وهي تحويل الجملة من الانكليزية إلى الفرنسية.
2. المجلد الثاني routes في هذا المجلد يتم تعريف المسارات لاستقبال الطلبات ((requestsواستخدام التوابع المناسبة لمعالجة الطلب على server , وفي ملف lang.py قمنا ببناء التابع lang حيث تستقبل english\_text مرسل من قبل client وتقوم باستخدام التابع en\_to\_fr لمعالجة الطلب والذي يتم في المجلد الثالث handlers ثم تقوم بإرجاع الجملة الفرنسية إلى client.
3. المجلد الثالث handlers وهو المجلد الذي يقوم بمعالجة الطلب على server, حيث في ملف lang.py ضمن المجلد نقوم بتحميل الأوزان الخاصة بالمعالجة مثل tokenizer\_en.pkl, tokenizer\_fr.pkl والأوزان الخاصة بمرحلة التدريب طبعا تم اختيار النموذج الأكبر دقة وهو LSTM(Word Embedding) وبعد ذلك تم تعريف التابع en\_to\_fr الذي يقوم باستخدام tokenizer ثم padding ثم predict المعرفة في keras ثم تحويل الأرقام كلمات باللغة الفرنسية ثم إعادة الناتج.
4. ملف server.py والذي يتم فيه تعريف IP address + Port Number لأنشاء عملية الاتصال عندما يريد client التخاطب مع FastAPI server.
5. ملف start.sh لبدء عملية تشغيل .server

بعد تشغيل السيرفر باستخدام الملف الاخير نلاحظ الخرج التالي وهو يعني جاهزية السيرفر لاستقبال طلبات الترجمة على الرابط التالي للاستخدام المحلي http://127.0.0.1:8000



نستطيع إرسال طلب عبر استخدام command line ولكن احد اسباب اختيار FastAPI هو أنه يقوم بتوفير Swagger UI تريحنا من هذه العملية وتوفر لنا واجهة مستخدم سهلة للتعامل معها ويتم ذلك عن استخدام نفس الرابط مع إضافة كلمة docs بالشكل http://127.0.0.1:8000/docs

فيظهر لنا هذه الواجهة مع كامل عملية التحويل الشكل(3-12)



)الشكل 3-12)

إلى هنا نكون قد انتهينا من مرحلة التعبئة (packaging) وسوف ننتقل إلى المرحلة الخامسة الخاصة بمرحلة deployment

# التوصيات

بالنسبة لمراحل العمليات من التخطيط إلى الانتاج تم التغاضي عن مرحلة مهمة في mlops cycle وهي مرحلة Test مثل اختبار المعماريات قبل عملية التدريب و واختبارات على النماذج الناتجة وكذلك اختبارات على web service قبل الأطلاق إلى مرحلة production.

أما بالنسبة لتسريع سير العمليات وتقليل الوقت في انتاج نماذج دقيقة في عملية الترجمة بشكل اسرع هو امتلاك موارد افضل مثل توفر 2 او 4 GPU على الاقل وذلك يساعد في عملية اختبار 4 المعماريات في وقت واحد واختيار المعمارية الافضل لنقوم بضبط hyperparmaters على عكس google colab الذي يمكننا من تشغيل GPU في وقت واحد ولا يعتبر سيرفر مستقل للدخول والخروج.

أما بالنسبة للمعماريات المستخدمة فلا تعتبر هي الافضل يوجد معماريات افضل في عملية الترجمة مثل نماذج الانتباه والمحولات.

# المراجع Reference

1. https://arxiv.org/pdf/1301.3781.pdf Tomas Mikolov, Kai Chen , Greg Corrado Jeffrey Dean, (September 2013) Efficient Estimation of Word Representations in Vector Space.
2. https://arxiv.org/pdf/1409.3215.pdf Ilya Sutskever , Oriol Vinyals ,Quoc V. Le,

(December 2014) Sequence to Sequence Learning with Neural Networks.

1. https://arxiv.org/pdf/1409.0473.pdf Dzmitry Bahdanau, KyungHyun Cho Yoshua Bengio∗, (may 2016) Neural Machine Translation by Jointly Learning to Align and Translate.