FACULTÉ DES SCIENCES DHAR EL MAHRAZ
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH

Master WISD
2021-2022

Traitement Automatique du Langage Naturel

---

hULMonA حلمنا
Le modèle universel de langage en arabe

Rapport de Projet

Réalisé par :

Anas Belfathi

Hamza El Filali

Hossam Boudraa

Encadré par :

Pr.Hicham Elmoubtahij

ABRÉVIATIONS

Bert	Bidirectional Encoder Representations from Transformers
SOTA	State Of The Art
TL	Transfer Learning Of Schedule
ULM	Universal Language Model
ELMO	Embeddings from Language Models
MSA	Modern Standard Arabic
ASTD	Arabic Sentiment Tweets Dataset
HARD	The Hotel Arabic Reviews Dataset
ReLU	Rectified Linear Unit
MT	Machine Translation
ASGD	Asynchronous Stochastic Gradient Descent
LSTM	Long short-term memory
AWD-LSTM	ASGD Weight Dropped LSTM
AJGT	Arabic Jordanian General Tweets
LEV	Levantine Arabic
NER	Named Entity Recognition
SA	Sentiment Analysis

Abstract

RÉSUMÉ :État de l'art et perspectives

Dans le domaine du traitement automatique du langage nature, la majorité des recherches menées et des réalisations accomplies ont porté principalement sur la langue anglaise et les langues dominantes en Europe. La langue arabe compte encore parmi les langues sous dotées. Ce n'est que depuis une dizaine d'années que cette langue a commencé à susciter un intérêt accru au sein de la communauté TAL, notamment compte tenu de son utilisation de plus en plus importante sur le Web social. Dans ce travail, nous nous focalisons à soutenir le développement du premier modèle de langage universel en arabe (hULMonA -), en démontrant son utilisation pour les tâches de classification en arabe. Ensuite, nous proposons de fournir un état de l'art sur le traitement automatique de ce modèle en l'évaluant avec un autre ULM multilingue tel que BERT. Enfin nous présentons et nous discutons les résultats des expériences de dans la tâche de l'analyse des sentiments.

Abstract : State-of-the-art and perspectives

In the field of natural language processing, the majority of expected research and achievements have focused mainly on the English language and the dominant languages in Europe. The Arabic language is still one of the under-competent languages. It is only in the past decade that this language has started to generate increased interest within the NLP community, especially given its increasingly important use on the social web. In this work, we focus on supporting the development of the first universal language model in Arabic (hULMonA - our dream), demonstrating its use for classification tasks in Arabic. Then, we propose to provide a state of the art on the automatic processing of this model by evaluating it with another multilingual ULM such as BERT. Finally we present and discuss the results of experiments in the sentiment analysis task.

MOTS-CLÉS : hULMonA, traitement automatique du langue naturel,langue arabe

KEYWORDS : hULMonA, natural language processing, arabic language

Chapitre 1

Introduction Générale

1.1 Contexte

Les modèles de langue pré-entraînés sont désormais indispensables pour obtenir des résultats à l'état-de-l'art (SOTA) dans de nombreuses tâches du TALN. En particulier, les modèles de langage universels (ULM), tels que le BERT, ont été développé grâce à une énorme quantité de textes bruts disponibles, ils permettent d'extraire des représentations continues des mots, contextualisées au niveau de la phrase.

L'efficacité de ces représentations pour résoudre plusieurs tâches de TALN a été démontrée récemment en utilisant l'apprentissage par transfert (TL) ce dernier a montré des résultats prometteurs pour améliorer la précision pour l'anglais. En revanche la langue arabe qui demeure une langue complexe avec des ressources limitées, rend difficile la production de tâches de classification de texte précises telles que l'analyse des sentiments.

Notre travail porte comme objet d'étude le développement du premier modèle de langage universel en arabe (hULMonA -notre rêve), en expliquant son utilité et son tache de classificateur dédié pour la langue arabe. Ceci et en tenant compte du modèle BERT multilingue pré-formé également pour l'arabe, nous procédons à une étude comparative de ces deux ULM basée sur une analyse des sentiments appris sur un corpus arabe hétérogène et de taille importante.

D'une manière générale, nous évaluons le modèle hULMonA sur diverses tâches en arabe (classification de textes, paraphrase, inférence en langage naturel, analyse des sentiments, désambiguïsation automatique) et montrons que ce modèle permet d'obtenir de nouveaux résultats de pointe et surpasse souvent l'autre approche de Bert sur le référentiel d'évaluation dans l'analyse des sentiments sur un ensemble de test.

1.2 Arrière plan TALN

L'évolution du Word Embeddings

Le modèle word2vec développé par (Mikolov et al. en 2013 a été l'un des pionniers modèles de la représentation significative des mots. Dès lors, la recherche a commencé à s'orienter vers des variations de word2vec tel que GloVe en 2014 et fastText en 2017. Alors que des résultats satisfaisants ont été obtenus avec ces premiers modèles, ils manquaient encore d'informations contextualisées, ce qui a été résolu par ELMO en 2018. De plus, la performance sur différentes tâches s'est amélioré sensiblement, conduisant à des structures plus grandes qui avaient des représentations supérieures des mots et des phrase.

Plusieurs modèles de compréhension du langage ont été développés tels que ULMFit (Howard et Ruder, 2018) et BERT (traité ici), qui offraient de meilleures performances en explorant différentes méthodes de préformation(pretraining), architectures des modèles modifié et un large corpus de formation plus importants.

Représentations non contextuelles pour l'arabe

Suite au succès de word2vec(Mikolov et al., 2013), pour l'anglais,le même exploit était visé par des chercheurs en TALN pour créer des incorporations spécifiques à la langue pour la langue arabe. word2vec arabe a été tenté pour la première fois par (Soliman et al., 2017), puis suivi d'un modèle Fasttext (Bojanowski et al., 2017) formé sur les données de Wikipédia et affichant de meilleures performances que word2vec.

Un autre chercheur en TALN (Erdmann et al., 2018) a présenté des techniques de formation des plongements sur deses variations de mots multidialectaux dialectales en arabe, tandis que (Abu Farha et Magdy, 2019 ; Abdul-Mageed et al., 2018) ont fourni des intégrations de mots arabes formées sur environ 250 millions de tweets.

Représentations contextualisées pour l'arabe

Google a publié pour les langues autres que l'anglais un modèle BERT multilingue (Devlin et al., 2018) prenant en charge plus de 100 langues avec des performances solides pour la plupart des langues. Cependant, le BERT monolingue de pré-formation pour les non-anglophones les langues se sont avérées offrir de meilleures performances que BERT multilingue comme BERT pour l'italien (Polignano et al en 2019) et d'autres accessibles au publicBERT (Martin et al., 2019 ; de Vries et al., 2019).

En particulier, plusieurs modèles de représentations contextualisées ont été conçus spécifiquement pour l'arabe, tels que hULMonA (ElJundi et al., 2019), qui utilise la structure ULMfit, dont les résultats de test de performance seront comparés aux celle du Bert dans le chapitre 3.

Chapitre 2

Méthodologie

Dans ce chapitre , nous décrivons comment ils ont construit Hulmona et comment ils ont ensuite réglé Hulmona et BERT ULM multilingue pour les tâches de classification en arabe. Le modèle complet consiste en la combinaison d'un modèle ULM préformé et de couches supplémentaires spécifiques aux tâches pour les tâches souhaitées. Une fois qu'un modèle ULM est développé, le processus d'apprentissage se limite à apprendre les paramètres des couches supplémentaires. Ce processus d'apprentissage par transfert est appelé fine-tuning avec ULM et c'est le principal avantage de l'utilisation des ULM.

Le modèle complet consiste en la combinaison d'un modèle ULM préformé et de couches supplémentaires spécifiques aux tâches pour les tâches souhaitées. Une fois qu'un modèle ULM est développé, le processus d'apprentissage se limite à apprendre les paramètres des couches supplémentaires. Ce processus d'apprentissage par transfert est appelé fine-tuning avec ULM et c'est le principal avantage de l'utilisation des ULM.

FIGURE 1 Étapes d'apprentissage du modèle hULMonA

2.1 DataSet

DataSet hULMonA

La dataset utilisée dans la phase d'entrainement du modèle universel hULMona a été construite par l'extraction des articles a partir de Wikipédia de 2019 . Les images, les liens, les balises HTML sont supprimées avec un outil de nettoyage online Ces articles sont constitué par 108M mots, et 4M mots unique. L'ensemble de données de référence est résumé dans le tableau 1 avec des statistiques sur leurs contenu :

- Hotel Arabic Reviews Dataset (HARD) : un ensemble de données collectée d'avis d'hôtels écrit en dialecte arabe standard moderne et arabe classé en positif et négatif.
- ASTD :un ensemble de données sur les tweets de sentiment pour l'arabe (ASTD) écrit en Arabe standard moderne et en dialecte égyptien.
- ArSenTD-Lev : un corpus de 4 000 tweets collectés dans les pays levantins (Palestine, Jordanie,Syrie et Liban) et annoté en particulier pour l'analyse des sentiment.

Dataset	Resource	samples	classes	MSA || Dialect
HARD Hotel reviews	93,700	2	MSA	Gulf (Elnagar et al., 2018) (www.booking.com)
ASTD Twitter	10,000	4	MSA	Egyptian (Nabil et al., 2015)
ASTD-B Twitter	1,600	2	MSA	Egyptian (Nabil et al., 2015)
ArSenTD-Lev Twitter	4,000	5	Levantine	Dialect(Baly et al., 2018)

TABLE 1 Statistiques des Datasets

DataSet Adaptée

Nous utilisons comme solution adaptée aux porblèmes rencontrés ,qui seront detaillés dans ce qui va suivre, une Dataset de classification des appréciations des clients des restaurants pour l'arabe , l'appréciation peut être négative, positive ou neutre. Nous avons trouvé cette dataset a partir d'un site des dataset arabe Masader(https ://arbml.github.io/). Cette dataset contient un certain nombre des appréciations bien classifiées et distribuées environ 10970 entrées . Mais nous utilisons une version plus petite qui contient 2642 entrées à cause des limites de l'editeur colab en usage de la RAM (12 GB) ,ce qui est insuffisant pour l'étape de fine-tunning .

Dataset	Resource	samples	classes	MSA || Dialect
Restaurant reviews V1	10970	3	MSA	Gulf (Masader(https ://arbml.github.io/))
Restaurant reviews V2	2642	3	MSA	Gulf (Masader(https ://arbml.github.io/))

TABLE 2 Statistiques des Datasets adaptées

2.2 Modèle Hulmona

Choix du modèle de base

La tâche source idéale était considérée comme la modélisation du langage et était considérée comme l'analogue d'ImageNet pour les tâches NLP. C'est pour la raison suivante : « Il capture de nombreuses facettes du langage pertinentes pour les tâches en aval, telles que les dépendances à long terme, les relations hiérarchiques et le sentiment. Contrairement à des tâches telles que la traduction automatique (MT) et l'implication, il fournit des données en quantités quasi illimitées pour la plupart des domaines et des langues. est un composant de diverses autres tâches de la NLP. Généralement, un bon modèle de langage (LM) comme l'AWD-LSTM, est choisi comme modèle de base. On s'attend généralement à ce que plus le modèle de base est bon, meilleures seront les performances du modèle final sur diverses tâches NLP après un réglage fin.

Domaine-Génerale LM pré-trainer

La pré-entrainement est à faire sur un large corpus de langage qui capte efficacement les principales propriétés et aspects du langage. Ce serait quelque chose comme le corpus Image-Net, mais, pour le langage. Cette étape ne doit être effectuée qu'une seule fois. Le modèle pré-entraîné résultant peut être réutilisé pour les étapes suivantes pour toutes les tâches. La pré-entrainement est effectuée de sorte que le modèle comprenne déjà les propriétés générales du langage et doit être légèrement modifié pour s'adapter à la tâche spécifique. En fait, il a été constaté que la pré-entrainement était particulièrement utile pour les petits ensembles de données et les ensembles de données de taille moyenne.

Réglage fin du LM de la tâche cible

Cette étape est effectuée pour que le modèle s'adapte au modèle de la tâche cible spécifique. Lorsqu'un modèle pré-entraîné est utilisé, la convergence à ce stade est alors plus rapide. Dans cette étape, un réglage fin discriminatif et des taux d'apprentissage triangulaires inclinés sont utilisés pour affiner le modèle de langage.

Ajustement précis discriminant

« Étant donné que différentes couches capturent différents types d'informations, elles doivent être affinées à des degrés divers. » Ainsi, pour chaque couche, un taux d'apprentissage différent est utilisé. L'apprentissage de la dernière couche, est fixe. Ensuite, $\eta^l - 1 = \eta^l/2.6)$ est utilisé pour obtenir le reste des taux d'apprentissage.

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta)$$

FIGURE 3 mise à jour des paramètres du fine-tuning

Ajustement du classificateur de tâche cible

Enfin, pour affiner le classificateur, nous augmentons le modèle de langage pré-entraîné avec deux blocs linéaires supplémentaires." Chaque bloc a les éléments suivants :

1. batch normalization
2. dropout
3. activation ReLU pour la couche intermédiaire
4. activation softmax pour la couche de sortie pour prédire les classes

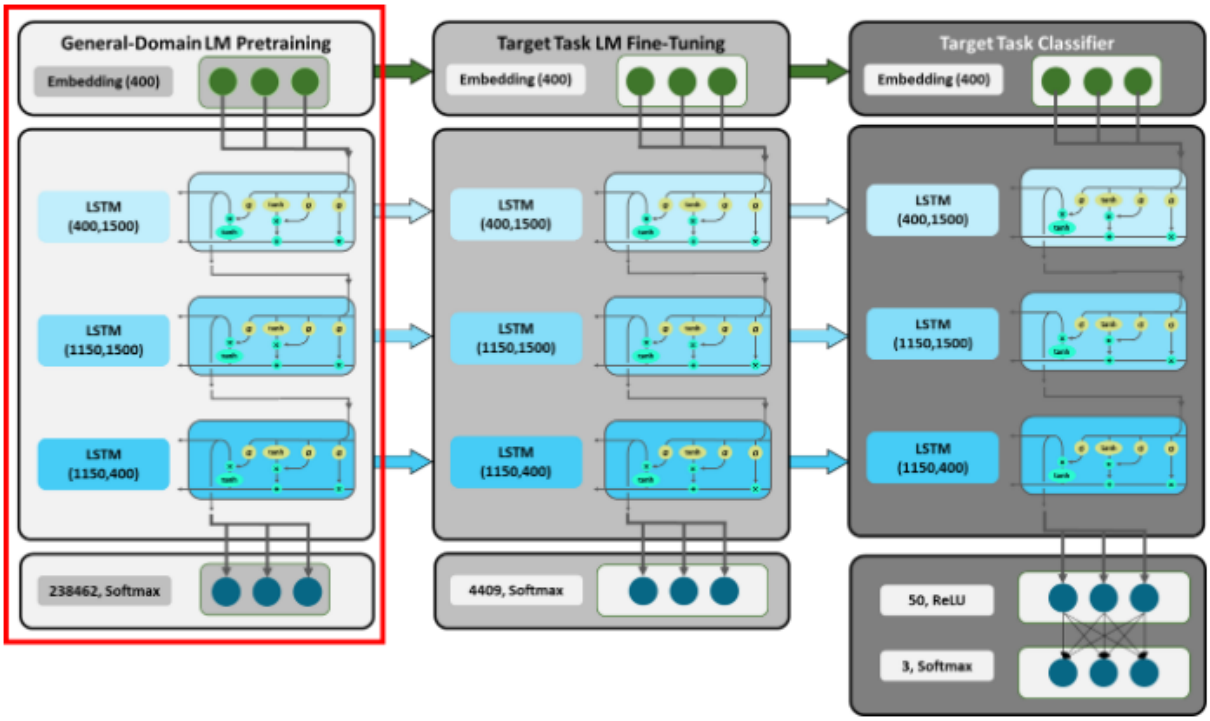


FIGURE 4 Classificateur de tâche cible

2.3 Multi-language BERT ULM pour les tâches en arabes

Comment fonctionne BERT ?

BERT utilise Transformer, un mécanisme d'attention qui apprend les relations contextuelles entre les mots (ou sous-mots) dans un texte. Dans sa forme vanille, Transformer comprend deux mécanismes distincts - un encodeur qui lit l'entrée de texte et un décodeur qui produit une prédiction pour la tâche. Puisque l'objectif de BERT est de générer un modèle de langage, seul le mécanisme d'encodeur est nécessaire. Contrairement aux modèles directionnels, qui lisent le texte saisi de manière séquentielle (de gauche à droite ou de droite à gauche), l'encodeur Transformer lit la séquence entière de mots en une seule fois. Par conséquent, il est considéré comme bidirectionnel, bien qu'il soit plus exact de dire qu'il est non directionnel. Cette caractéristique permet au modèle d'apprendre le contexte d'un mot en fonction de l'ensemble de son environnement (gauche et droite du mot).

Lors de l'entrainement de modèles de langage, il est difficile de définir un objectif de prédiction. De nombreux modèles prédisent le mot suivant dans une séquence Par exemple : عاد الطفل من .....

Une approche directionnelle qui limite intrinsèquement l'apprentissage du contexte. Pour surmonter ce défi, BERT utilise une stratégies de l'entrainement .

Prédiction de la prochaine phrase

Dans le processus de l'entrainement de BERT, le modèle reçoit des paires de phrases en entrée et apprend à prédire si la deuxième phrase de la paire est la phrase suivante dans le document d'origine. Pendant l'apprentissage, 50 % des entrées sont une paire dans laquelle la deuxième phrase est la phrase suivante dans le document d'origine, tandis que dans les 50 % restants, une phrase aléatoire du corpus est choisie comme deuxième phrase. L'hypothèse est que la phrase aléatoire sera déconnectée de la première phrase.

Pour aider le modèle à distinguer les deux phrases en l'entrainement, l'entrée est traitée de la manière suivante avant d'entrer dans le modèle :

1. Un jeton [CLS] est inséré au début de la première phrase et un jeton [SEP] est inséré à la fin de chaque phrase.
2. Une phrase incorporée indiquant Phrase A ou Phrase B est ajoutée à chaque jeton. Les incorporations de phrases sont similaires dans leur concept aux incorporations de jetons avec un vocabulaire de 2.
3. Une intégration positionnelle est ajoutée à chaque jeton pour indiquer sa position dans la séquence. Le concept et la mise en œuvre de l'intégration positionnelle sont présentés dans l'article Transformer.

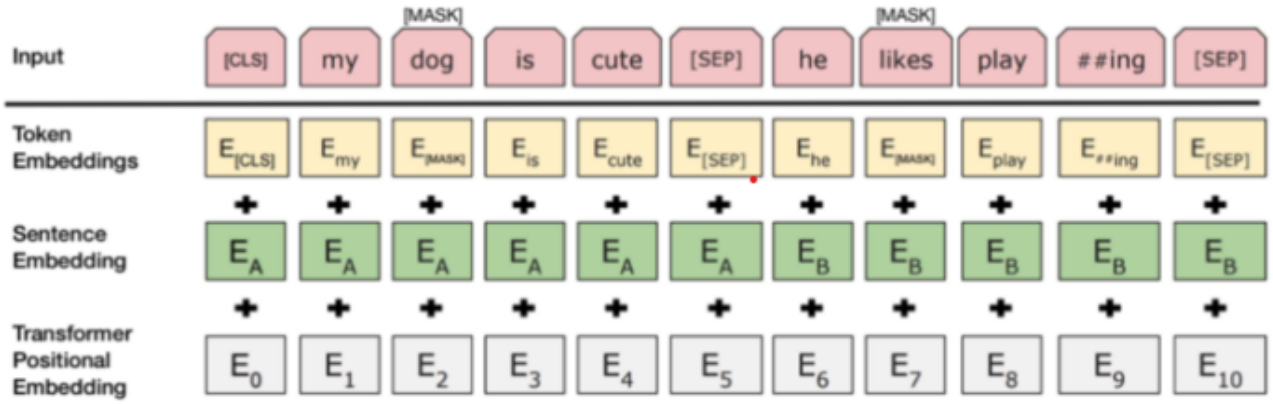


Figure 5 Processus de l'entrainement de BERT

Pour prédire si la deuxième phrase est bien connectée à la première, les étapes suivantes sont effectuées :

1. La séquence d'entrée entière passe par le modèle Transformer.
2. La sortie du jeton [CLS] est transformée en un vecteur de forme 2×1, à l'aide d'une simple couche de classification (matrices apprises de poids et de biais).
3. Calcul de la probabilité de IsNextSequence avec softmax.

Modèle Fine-Tuning

BERT peut être utilisé pour une grande variété de tâches linguistiques, tout en n'ajoutant qu'une petite couche au modèle de base :

1. Les tâches de classification telles que l'analyse des sentiments sont effectuées de la même manière que la classification Next Sentence, en ajoutant une couche de classification au-dessus de la sortie Transformer pour le jeton [CLS].
2. Dans les tâches de réponse aux questions , le logiciel reçoit une question concernant une séquence de texte et doit marquer la réponse dans la séquence. À l'aide de BERT, un modèle QA peut être formé en apprenant deux vecteurs supplémentaires qui marquent le début et la fin de la réponse.
3. Dans Named Entity Recognition (NER), le logiciel reçoit une séquence de texte et doit marquer les différents types d'entités (Personne, Organisation, Date, etc.) qui apparaissent dans le texte. À l'aide de BERT, un modèle NER peut être formé en alimentant le vecteur de sortie de chaque jeton dans une couche de classification qui prédit l'étiquette NER.

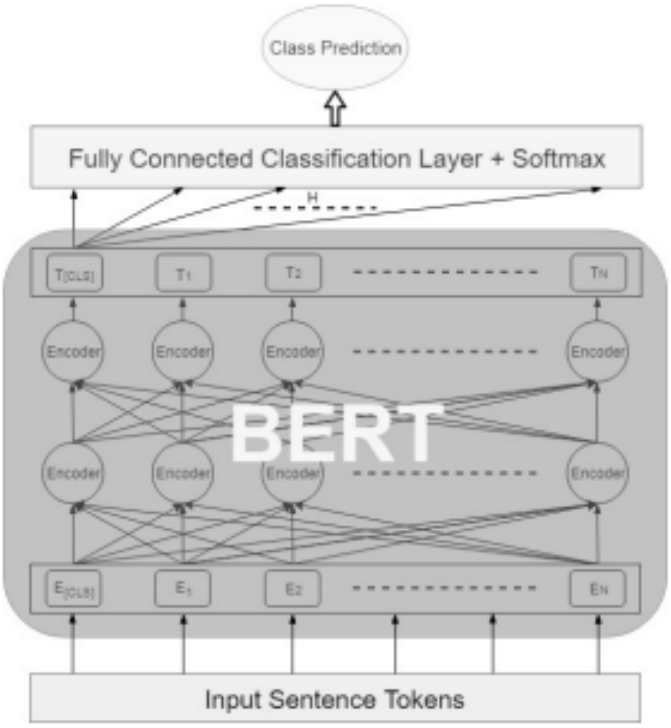


FIGURE 6 Architecture fine-tuning du modèle BERT

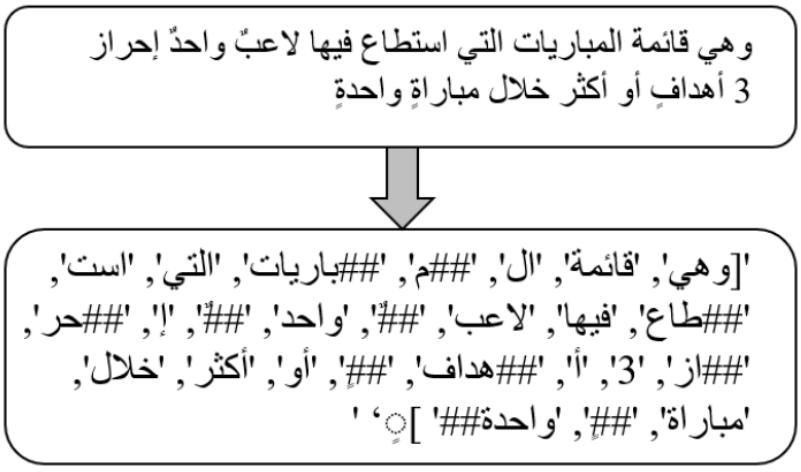


FIGURE 7 Résultats du tokenizer BERT

Chapitre 3

Réalisation

3.1 Fastai :Une API en couches pour le Deep Learning

fastai est une bibliothèque d'apprentissage en profondeur qui fournit aux praticiens des composants de haut niveau qui peuvent fournir rapidement et facilement des résultats de pointe dans des domaines d'apprentissage en profondeur standard, et fournit aux chercheurs des composants de bas niveau qui peuvent être mélangés et associés pour créer de nouveaux approches. Il vise à faire les deux choses sans compromis substantiels en termes de facilité d'utilisation, de flexibilité ou de performances. Ceci est possible grâce à une architecture soigneusement en couches, qui exprime des modèles sous-jacents communs à de nombreuses techniques d'apprentissage en profondeur et de traitement de données en termes d'abstractions découplées. Ces abstractions peuvent être exprimées de manière concise et claire en tirant parti du dynamisme du langage Python sous-jacent et de la flexibilité de la bibliothèque PyTorch. fastai inclut : un nouveau système de distribution de types pour Python ainsi qu'une hiérarchie de types sémantiques pour les tenseurs ; une bibliothèque de vision par ordinateur optimisée pour le GPU qui peut être étendue en Python pur ; un optimiseur qui refactorise la fonctionnalité commune des optimiseurs modernes en deux éléments de base, permettant aux algorithmes d'optimisation d'être implémentés dans 4 à 5 lignes de code ; un nouveau système de rappel bidirectionnel qui peut accéder à n'importe quelle partie des données, du modèle ou de l'optimiseur et le modifier à tout moment pendant la formation ; une nouvelle API de bloc de données ; et beaucoup plus. Nous avons utilisé cette bibliothèque pour créer avec succès un cours complet d'apprentissage en profondeur, que nous avons pu écrire plus rapidement qu'en utilisant les approches précédentes, et le code était plus clair. La bibliothèque est déjà largement utilisée dans la recherche, l'industrie et l'enseignement.

```
Dans le traitement du langage naturel (TALN) moderne, l'approche la plus importante pour créer des modèl
```

3.2 La phase d'analyse du code source original du projet

Language Model

- Chargement du dataset du wikipedia (600k Articles) et le diviser en Training set (90 %) et Testing set (10%)

```
df = pd.read_csv(path/'data'/'all_arabic_wiki_2019.txt.MyD3.tok', sep='\t', header=None, names=['text'])
df.head()
```

	text
0	...+ماء ال+ ماء ماده شفافه عديمه ال+ لون و
1	... رياضيات ال+ رياضيات علم عباره عن مفاهيم
2	... استونيا استونيا , رسميا جمهوريه استونيا
3	... س+ نجاب ال+ سنجاب حيوان من ال+ قوارض ,
4	...تلفاز ال+ تلفاز او ال+ تلفزه او ال+ تلفز

• Les données de fast.ai sont consommées par des modèles de langage utilisant la classe TextLMDataBunch, dont les instances peuvent être construites à partir du format DataFrame que nous venons de préparer à l'aide de la commande suivante :

```
data_lm = TextLMDataBunch.from_df(path, train_df=df_train, valid_df=df_val, bs=bs, tokenizer=ar_tok, text_cols=0, label_cols=None, min_freq=5)
```

```
data_lm.show_batch()
```

idx	text
0	بودن هو لاعب كره قدم هندوراسي يجيد ال+ xxunk جيري ريكاردو xxunk جيري xxbos . جوهانس ريتر جوهانس ريتر هو مدرب كره قدم و+ لاعب كره قدم دانمركي في مركز ال+ وسط . لعب مع نادي نوردشيلاند xxbos . ال+ نوادي , ف+ قد لعب مع و اسامه عبد ال+ واحد اسامه حمدي عبد ال+ واحد , استاذ جامعي مصري xxbos . ال+ ايراني xxunk لعب لك+ مهاجم ل+ نادي
1	ال+ وسطي قبل ال+ ميلاد في شبه ال+ جزيره ال+ كوريه . تم اكتشاف حقول ال+ ارز من xxunk و+ تواريخ ال+ كربون ال+ مشع , يشار ان زراعه ا ل+ ارز في ال+ حقول ال+ جافه قد تكون بدات في وقت مبكر من فتره فخار xxunk عن حبيبات ارز xxunk تي xxunk قاموا ب+ حفر مواقع حقول ال+ ارز في موقع xxunk في xxunk قبل معاهد مثل متحف جامعه
2	ل+ مجلس ال+ قياده , ثم تم اختيار + ه لك+ احد اعضاء ال+ مكتب ال+ سياسي ل+ ال+ حزب ال+ ديمقراطي ال+ كردستاني . في عام رقم م , تم تعيين + ه من قبل والد + ه رئيس اقليم كردستان ال+ سابق مسعود بارزاني رئيسا ل+ مجلس امن اقليم كردستان ل+ ال+ اشراف علي ال+ امن و+ ال+ مخابرات ال+ عسكريه و+ اجهزه ال+ مخابرات ال+ حاليه .
3	اثناء عمليه ال+ صب يري احد ال+ عمال جالسا و+ يمنع ب+ واسطه قطعه . xxunk ثم الي يمين ال+ صوره صب ال+ برونز ال+ منصهر علي لوحه صب بحيث ينتج من + ها لوحا , xxunk و+ هم يقومون ب+ وزن ال+ مواد ب+ ميزان , ثم صهر + ه في عده خشب في يد + ه , نزول ال+ شوائب ال+ طافيه علي سطح ال+ برونز ال+ منصهر
4	جمع ال+ قراءات , الا ان وفاه ال+ شيخ محمد سليم حالت دون ذلك , ف+ اتصل ب+ ولد + ه , ال+ شيخ احمد ال+ حلواني ال+ حفيد , و+ جمع علي + ه ال+ قراءات ال+ عشر من طريق الشاطبيه و+ ال+ دره , ثم جمع بعد ذلك ال+ عشر ال+ صغري ايضا علي ثم اتصل ب+ ال+ شيخ عبد ال+ قادر , xxunk ال+ شيخ محمود فايز

• nous devons créer une instance de la classe language_model_learner fast.ai avec la commande suivante :

```
learn_lm = language_model_learner(data_lm, AWD_LSTM, drop_mult=0.1, pretrained=False)
```

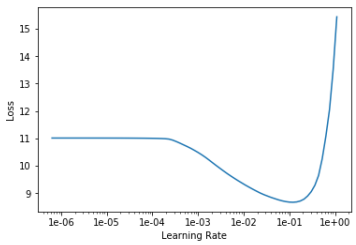
• Ici, AWD_LSTM signifie ASGD weightdropped LSTM. Il s'agit simplement de l'architecture LSTM habituelle dans laquelle certains poids ont été supprimés de manière aléatoire, comme ce que fait la couche de suppression habituelle avec les activations de réseaux neuronaux, par opposition aux poids. Cet ensemble de données , officiellement appelé «all_arabic_wiki_2019 », est une collection d'articles de Wikipédia jugés «bons» par les humains. C'est une belle source propre de données non supervisées qui a été utilisée par un grand nombre d'articles de NLP à des fins d'analyse comparative.

• Maintenant que nous avons chargé une instance du modèle et quelques poids pré-entraînés, nous allons essayer de déterminer le meilleur taux d'apprentissage ou optimal pour affiner notre modèle de langage. Une méthode utilitaire astucieuse dans fast.ai appelée lr_find peut le faire automatiquement pour nous. Il parcourt un certain nombre de taux d'apprentissage et détecte le point auquel la fonction de perte chute le plus rapidement sur la courbe résultante de la perte par rapport au taux d'apprentissage. De manière équivalente, c'est là que le gradient de perte est le plus petit. Nous pouvons le réaliser rapidement en utilisant notre modèle d'apprentissage de la langue comme suit :

```
learn_lm.lr_find()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

```
learn_lm.recorder.plot()
```



Lors de notre exécution du code,le taux d'apprentissage optimal renvoyé était d'environ : 8 e-1

• Après avoir trouvé le taux optimal, nous pouvons maintenant affiner notre modèle LSTM avec perte de poids pré-entraîné avec un taux d'apprentissage triangulaire incliné à l'aide de la commande fit_one_cycle fast.ai comme indiqué ci-dessous :

```
learn_lm.fit_one_cycle(1, 3e-3)
epoch        train_loss        valid_loss        accuracy        time
0            3.337320          3.258085          0.437729          03:02:59
```

• L'exécution de la commande donne une précision de 0,437729 en environ 3 minutes de réglage fin.

- La prédiction du modèle donne le résultat suivant :

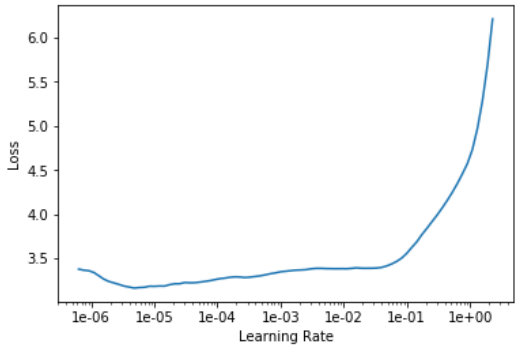
```
learn_lm.predict('درس ال+ ولد ', n_words=33)
```

```
'+درس ال+ ولد في ال+ صف ال+ اول تعلم ال+ صف ال+ اخير كتاب تفسير ابي صامت ل+ ال+ علامه سعيد احمد ف+ ال+ بن كتاب السيوطي ال+ مشرف علي ال+ تكمله في ال+ مدرسه ال'
```

- Après avoir obtenu cette valeur de référence, nous aimerions savoir si un réglage fin discriminatif peut conduire à une amélioration. Pour ce faire, nous dégelons d'abord toutes les couches avec la commande unfreeze, puis nous utilisons la méthode slice pour spécifier les limites supérieure et inférieure de la plage de taux d'apprentissage. Cette commande définit le taux d'apprentissage maximal de la couche la plus proche de la sortie à la limite supérieure et géométriquement diminue - via une division par un facteur constant - le taux d'apprentissage maximal de chaque couche suivante vers la limite inférieure. Le code exact pour ce faire est illustré ci-dessous :

```
learn_lm.unfreeze()
#learn_lm.freeze_to(-1)
learn_lm.lr_find()
learn_lm.recorder.plot()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```



3.3 Fine-Tuning

Rappel

Quelle que soit la généralité du modèle pré-entraîné initial, l'étape finale de déploiement impliquera probablement des données provenant d'une distribution différente. Cela nous motive à affiner le modèle pré-entraîné général sur un nouvel ensemble de données plus petit de la nouvelle distribution pour nous adapter au nouveau scénario. Les auteurs de l'ULMFiT ont découvert que les techniques d'ajustement discriminatif et de taux d'apprentissage inclinés atténuent le double problème de surapprentissage et d'oubli catastrophique rencontré par les chercheurs lors de cette opération. Le réglage fin discriminatoire stipule que, parce que différentes couches du modèle de langage capturent des informations différentes, elles doivent être ajustées à des taux différents. En particulier, les auteurs ont constaté empiriquement qu'il était avantageux de commencer par affiner la toute dernière couche et de noter son taux d'apprentissage optimal.

• Nettoyage des datasets du Fine-tuning

L'analyseur morphologique MADAMIRA (Pasha et al., 2014) : est un système d'analyse morphologique et de désambiguïsation de l'arabe qui exploite certains des meilleurs aspects des deux systèmes existants et les plus utilisés pour le traitement automatique de la langue arabe que sont : MADA ((Habash Rambow, 2005) ; (Habash et al., 2009) ;. (Habash et al., 2013)) et AMIRA (Diab, 2009). En effet, MADAMIRA permet la tokenisation, la lemmatisation, le racinisation, l'étiquetage morpho-syntaxique, la désambiguïsation morphologique, la diacritisation, la reconnaissance des entités nommées, etc.

Les ressources système importantes (analyseurs et modèles) sont indiquées. Le texte d'entrée entre dans le préprocesseur et circule dans le système, chaque composant ajoutant des informations supplémentaires que les composants suivants peuvent utiliser. En fonction de la sortie demandée, le processus peut se terminer et renvoyer des résultats à différentes positions dans la séquence.

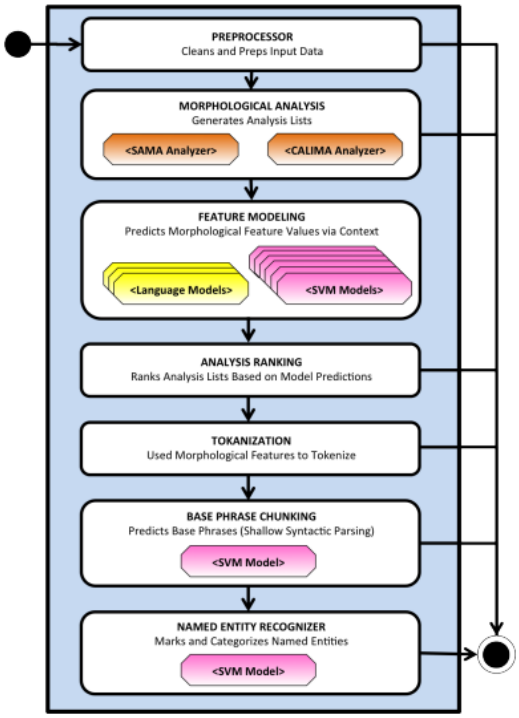


FIGURE 7 Vue d'ensemble de l'architecture MADAMIRA

Et voilà le code suivant dans lequel on va nettoyer nôtre dataset pour l'utiliser après dans une tâche de la classification.

```
import string

#preparing punctuations list
arabic_punctuations = '''`÷×؛<>_()*&^%][ـ،/:"؟.,'{}~¦+|!”…“–ـ''
english_punctuations = string.punctuation
punc_to_remove = ''.join(set(arabic_punctuations + english_punctuations))
punc_to_keep = '+'
punc_to_escape = '''[]-^'''
for p in punc_to_keep: punc_to_remove = punc_to_remove.replace(p, '')
for p in punc_to_escape: punc_to_remove = punc_to_remove.replace(p, '\\{}'.format(p))
print(punc_to_remove)

def pre_process(text):
    text = text.replace('\\n', ' ').replace('\n', ' ')
    text = text.replace('؛', '،')
    text = re.sub(r'\([^)]+\)', '', text)  # remove parentheses and everything in between
    text = re.sub(r'[a-zA-Z]', '', text)  # remove non-arabic characters
    text = re.sub(r'\d+(\.\d+)?', ' رقم ', text)  # replace numbers by special token
    for p in punc_to_remove: text = text.replace(p, '')  # remove punctuations
    text = re.sub(r'(.)\1{2,}', r'\1', text)  # remove repeated chars
    text = re.sub(r'\s+', r' ', text)
    return text
```

NB : On générale dans la partie du Fine-tuning ils ont utiliser plusieurs datasets pour soutenir nôtre modèle a prédire des résultats parfaits.

- **Fine-Tune pre-trained LM**

Chargement du dataset

```
from sklearn.model_selection import train_test_split

df = pd.read_csv(path/'data'/'ASTD_final.csv')
print(df.shape)
#df.dropna(inplace=True)
#print(df.shape)
#df.Tweet = df.Tweet.apply(lambda x: pre_process(x))
#df.label.replace({'very_positive':2, 'positive':1, 'neutral':0, 'negative':-1, 'very_negative':-2}, inplace=True)
np.random.seed(42)
df_train, df_val = train_test_split(df, test_size=0.2)
print(df_train.shape, df_val.shape)
#df.columns
df_train.head()
```

```
(10006, 2)
(8004, 2) (2002, 2)
```

	category	text
7862	-2	إجلاء أكثر من ألفي شخص تحسباً لإعصار الفلبين
1095	-2	للتخلص من مرسي عالدي هو الحل
3123	0	رقم يا أخي أي منصب رفضت المناصب اللي عرضت علي...
8503	1	وحدوا الله يا جماعة وبلاش إشاعات عن مفاجآت عشا...
1561	-1	بالفيديو عكاشة والله هيفاء وهبي ست مش طربة وال...

- Créez un TextDataBunch adapté à la formation d'un modèle de langage,Tous les textes datasets sont concaténés et les étiquettes sont ignorées. Au lieu de cela, la cible est le mot suivant dans la phrase.

```
ar_tok = Tokenizer(lang='ar')
#data_lm = TextLMDataBunch.from_df(path, train_df=df_train, valid_df=df_val, text_cols=0, label_cols=None, tokenizer=ar_tok, bs=32)
data_lm = TextLMDataBunch.from_df(path, train_df=df_train, valid_df=df_val, text_cols=1, label_cols=None, tokenizer=ar_tok, bs=64, include_bos=False, min_freq=4)
```

```
/usr/local/lib/python3.7/dist-packages/fastai/core.py:302: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple c
  return np.array(a, dtype=dtype, **kwargs)
```

```
data_lm.train_ds.x
```

```
LMTextList (8004 items)
xxunk من أكثر xxunk شخص xxunk xxunk xxunk,xxunk من مرسي xxunk يا رقم , هو الحل xxunk أي منصب رفضت المناصب اللي xxunk عليا xxunk اني xxunk xxunk ومش محتاج xxunk xxunk xxunk الله
xxunk xxunk,xxunk جماعة يا الله xxunk xxunk عن xxunk عشان مش عايزين xxunk كل الحكاية ملف xxunk xxunk xxunk مش ست وهبي وهيفاء والله عكاشة بالفيديو مع بعض,xxunk والنبي xxunk خد xxunk بقى xxunk نوابة
القاهرة مصر توفيق عكاشة
Path: /content/drive/MyDrive/TALN PROJET
```

- Créer un Learner avec language model from data_lm et l'architecture .

```
config = awd_lstm_lm_config.copy()
config['n_hid'] = 1150
learn_lm = language_model_learner(data_lm, AWD_LSTM, drop_mult=0.2, pretrained_fnames=pretrained_fnames,config=config)
```

- Maintenant on va déterminer le meilleur taux d'apprentissage pour affiner le modèle

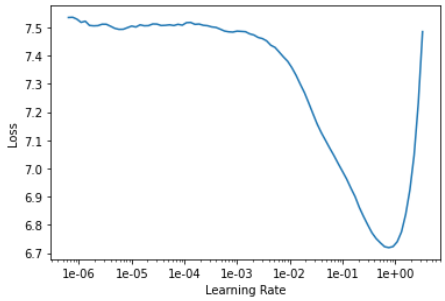
```
learn_lm.lr_find()
learn_lm.recorder.plot()
```

75.00% [3/4 09:18<03:06]

epoch	train_loss	valid_loss	accuracy	time
0	7.504794	#na#		03:08
1	7.485963	#na#		03:06
2	6.930494	#na#		03:03

80.77% [21/26 03:45<00:53 9.6823]

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```



- Un faible taux d'apprentissage est lent mais plus précis. À mesure que le taux d'apprentissage augmente, la vitesse d'apprentissage augmente également, jusqu'à ce que le taux d'apprentissage devienne trop important et diverge. Trouver le sweet spot demande de l'expérimentation et de la patience. C'est pour cela on ne prend pas l'optimum exacte pour n'ajuste pas le Language Model et aussi pour le temps d'apprentissage .Et après le test du modèle avec plusieurs paramètres nous avons tombé dans la meilleurs valeur de taux d'apprentissage d'environ : 5 e-2

```
learn_lm.fit_one_cycle(1, 2e-2)
epoch        train_loss        valid_loss        accuracy        time
0            3.337320          3.258085          0.437729          03:02:59
```

Donc il a donner une précision de 0.362.

- Cette commande définit le taux d'apprentissage maximal de la couche la plus proche de la sortie à la limite supérieure.

```
learn_lm.unfreeze()
```

```
learn_lm.fit_one_cycle(6, slice(5e-3/(2.6**4),5e-3))
```

50.00% [3/6 16:13<16:13]

epoch	train_loss	valid_loss	accuracy	time
0	5.076119	4.882254	0.372545	05:28
1	4.954770	4.732567	0.382940	05:23
2	4.781408	4.640286	0.390944	05:21

7.69% [2/26 00:22<04:34 4.7700]

epoch	train_loss	valid_loss	accuracy	time
0	5.076119	4.882254	0.372545	05:28
1	4.954770	4.732567	0.382940	05:23
2	4.781408	4.640286	0.390944	05:21
3	4.596817	4.605157	0.394452	05:21
4	4.432697	4.606258	0.395759	05:21
5	4.311129	4.608343	0.395568	05:22

Donc nous avons arrivé à augmenter la prédiction du modèle.

- Les mots sont choisis au hasard parmi les prédictions, en fonction de la probabilité de chaque indice.

```
learn_lm.predict('الله يبارك في ك+', n_words=30)
```

```
Resultat:

الله يبارك في ك+ الله على أكل عجيب كافيه الزحمة جو لطيف للبيتزا مكان يجمع بين الخصوصية دول لطيف مطعم جميل وراقي جدا من جميع النواحي الطعام نظيف جدا مدهش حلوة من اهم العاملين
```

3.4 Classification

- les données de fast.ai sont consommées par un classificateur spécifique à une tâche à l'aide de la classe TextClas-DataBunch. Nous construisons une instance de cette classe, en préparation de la sous-section suivante, à partir de nos DataFrames en utilisant la commande analogue suivante :

```
ar_tok = Tokenizer(lang='ar')
data_clas = TextClasDataBunch.from_df(path, train_df=df_train, valid_df=df_val, text_cols=1, label_cols=0, tokenizer=ar_tok, bs=32, vocab=data_lm.train_ds.vocab,
```

```
data_clas.save(path/'data'/'data_clas_ASTD_min4.pkl')
```

```
data_clas = load_data(path, path/'data'/'data_clas_HARD.pkl', bs=32)
```

```
data_clas.show_batch()
```

text	target
الداصبي مرسي ف اخر ليل + ه ل + ه استخدم مفرد + ه ال+ شرعيه عشرات ال+ مرات لان عقد + ه ال+ شرعيه عند + ه وصلت حد + ها و+ هذا عقد + ه نقص + ه شرعي + ه قياد + ه ال+ بحارنه من زمان	-1
و+ من يردد + ها هم من لا تريد + هم في مصر لا في ال+ سياسه و+ لا في ال+ برلمان ال+ قادم و+ لك+ فاهم ال+ ضرر الذي احدثوا + ه ل+ ثقافه + دا و+ وعي + دا ال+ سياسي بلغه + هم و+ اساليب + هم	-1
ال+ مصري ال+ حقيقي لك+ ال+ هرم ال+ شامخ رفقه + ه شرف و ال+ تواصل مع + ه حق و نسيان + ه محال و ال+ دعاء ل + ه واجبكل عام و انتم ب+ خير و مصر طيب + ه و حره امن + ه	1
من قام ليل + ه ال+ قدر ايمانا و+ احتسابا غفر ل + ه ما تقدم من ذنب + ه قف بين يدي ال+ رحمن و+ اطل ال+ وقوف و+ قل يا رب اتيه + ك ب+ ذنوب كثير + ه و انت ال+ عفو ال+ كريم	1
ابراهيم عيسي قلم شجاع و+ ضمير يقظ و+ علام + ه بارز + ه في تاريخ ال+ مهنه تحي + ه ل + ه و+ ل+ فريق ال+ دستور و+ س+ تبقي حريه ال+ صحافه في مصر ثمر + ه مستحق + ه ل+ تضال ال+ صحفيين	1

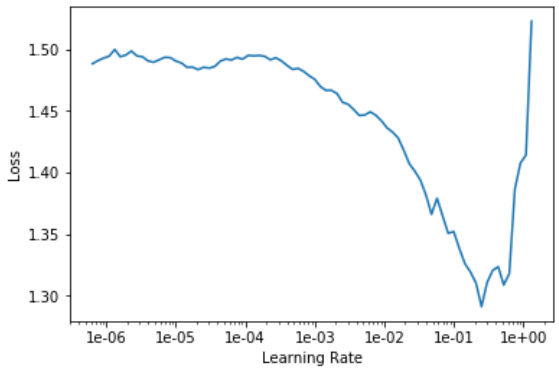
- Nous avons créé un objet pour la consommation de données par le classificateur de tâche cible. Nous avons appelé cette variable data_clas. Comme prochaine étape pour affiner notre classificateur de tâche cible, nous devons instancier une instance d'un apprenant de classificateur, avec la méthode bien nommée text_classifier_learner dans fast.ai. Cela se fait avec le code suivant :

```
learn_clas = text_classifier_learner(data_clas, AWD_LSTM, drop_mult=0.2, metrics=[accuracy,FBeta(average='weighted')])
learn_clas.load_encoder('fine_tuned_encoder_ASTD_acc326_min4')
```

- À l'étape suivante, nous utilisons à nouveau la méthode utilitaire fast.ai lr_find pour trouver le taux d'apprentissage optimal, avec le code suivant :

```
learn_clas.lr_find()
learn_clas.recorder.plot()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```



• On voit que le taux optimal est d'environ 2e-2 . Nous formons l'apprenant du classifieur pour une époque, en utilisant des taux d'apprentissage triangulaires inclinés, via le code suivant :

```
learn_clas.fit_one_cycle(1, 2e-2)
```

Total time: 01:04

epoch	train_loss	valid_loss	accuracy	f_beta	time
0	0.851637	0.866255	0.678322	0.641764	01:04

L'exécution du code donne une précision d'environ : 0.678 .

• Quand on **unfreeze** une seule couche et on le **fine-tune** , et on prend une autre couche, on fait la même chose. On répète ce processus en un nombre de fois . Les auteurs de l'ULMFiT ont constaté que l'application de cette méthode lors de la mise au point de l'étape de classification de la tâche cible améliore considérablement les résultats.

```
learn_clas.freeze_to(-2)
#learn_clas.lr_find()
#learn_clas.recorder.plot()
#learn_clas.fit_one_cycle(2, slice(1e-2/(2.6**4),1e-2))
```

```
learn_clas.fit_one_cycle(1, slice(5e-3/(2.6**4),5e-3))
```

Total time: 01:07

epoch	train_loss	valid_loss	accuracy	f_beta	time
0	0.843509	0.848543	0.690310	0.665196	01:07

- Finalement le résultat finale de différentes classifications est comme suit :

Dataset	epoch	train_loss	valid_loss	accuracy	f_beta	seed	test split
ASTD-B	6	0.438674	0.448438	0.864662	0.857988	42	0.1
ASTD	6	0.659341	0.857126	0.698801	0.677395	42	0.2
AJGT	5	0.429798	0.452757	0.805556	0.745665	14	0.2
LEV	16	1.185471	1.251261	0.523750	0.511322	14	0.2
HARD	7	0.121737	0.128373	0.956695	0.956682	43	0.2

3.5 Problèmes et limites d'étude

Il est important de reconnaître que lors de l'étape de l'exécution et de la compréhension du projet hULMonA notre étude est sujette à certaines limites.

En termes de qualité des données, il y avait un manque d'informations sur les ensembles de données de dataset y compris la manière dont ils ont été collectés ou s'il s'agissait d'un échantillon complet ou d'un échantillon partiel.

En savoir plus sur la qualité des données (filtrage des tweets en arabe uniquement lors de la collecte des données) pourrait être amélioré, malheureusement nous avons rencontré un obstacle en raison de la limitation de l'usage de RAM du notebook Google Colab.

De plus, compte tenu du temps et de la puissance de calcul, former le modèle sur un plus grand corpus de texte, nécessite un dispositif matériel de haut niveau (des ordinateurs avec des capacités élevés,..).

Enfin, bien que le réglage fin des paramètres (fine-tuning) et la sélection du modèle n'entraient pas dans le cadre de la classe, plus la recherche manque des ressources (fichiers utiles pour l'exécution deaurait pu être faite sur l'ingénierie des fonctionnalités, en sélectionnant le hyperparamètres utilisés dans la grille de recherche ou comment optimiser les modèles Les résultats de l'analyse peuvent changer si l'ensemble de ressources de ce projet est limité aux fichiers spécifiquement hébergés sur le site GitHub. Cela aurait été intéressant de recueillir tous les fichiers du projet afin de tracer d'une manière précise les étapes de l'élaboration de ce projet.

Ce projet a certainement été utile pour élargir notre compréhension d'un projet de langage naturel et d'apprentissage automatique de bout en bout.La partie suivante mettra en évidence notre contribution pour l'améioration du projet hULMonA.

3.6 Contribution

Comme contributions majeures de ce projet, nous nous référons particulièrement aux éléments suivants :

La première contribution se place dans le cadre de l'apprentissage par transfert séquentiel du domaine source des appréciations du client vers la classification ciblé (bon avis , mauvaise avis,neutre . . .), qui vise à induire un biais inductif pour améliorer la performance des tâches de la NLP dans un régime à faibles ressources.

L'objectif est de mieux exploiter les connaissances acquises dans un modèle source, préalablement entraîné sur le domaine de l'actualité à ressources élevées. Pour cela, nous proposons deux méthodes simples mais efficaces . Dans la première, les connaissances pré-apprises sont transférées au modèle cible sous forme de représentations contextuelles. Dans la deuxième méthode, les connaissances pré-apprises sont transférées sous la forme de poids pré-formés utilisés pour initialiser les paramètres du modèle cible.

La deuxième contribution vise à repérer les limites de la méthode standard d'apprentissage par transfert séquentiel. Plus précisément, à travers un ensemble de méthodes interprétatives, nous étudions comment les représentations internes (neurones individuels) des modèles pré-entraînés sur le domaine de dialects sont mises à jour lors de l'ajustement sur le domaine des avis de restaurants.

Nous constatons que bien que capables de s'adapter à de nouveaux domaines, certains neurones préformés sont biaisés par ce qu'ils ont appris dans l'ensemble de données source, et ont donc du mal à apprendre des modèles inhabituels spécifiques à une cible, ce qui peut expliquer le transfert négatif caché observé.

3.6.0.1 Fine-tuning du model sur dataset du restaurant (Res.csv)

Le code source : Le but de créer un ULM c'est d'augmenter la qualité des modèles dans differents tâches de NLP et meme de réserver un certain bout de temps dans l'entrainement avec la technique de fine tuning . Donc maintenant nous allons appliquer une nouvelle tâche d'analyse des sentiments avec une dataset qui contient des apprétiations sur des restaurants . c'est pour cela en va suivre une approche similaire a l'ancienne en utilisant la bibliotheque FASTAI .

- Tout d'abords on va charger le dataset et faire quelque pré-traitement :

```
pretrained_fnames=['Ar-LM_unfreeze_1e-4_after3_2_acc475','itos']

from sklearn.model_selection import train_test_split

df = pd.read_csv(path/'data'/'RES2.csv')
column_names = ["polarity", "text"]
df = df.reindex(columns=column_names)

for i,text in enumerate(df["text"]):
    df["text"][i] = pre_process(text)
```

	polarity	text
1149	1	مطعم ممتاز لولا بطء الخدمة وارتفاع أسعار الوجب...
965	-1	لا أنصح به سيء والأكل بارد والسعر مرتفع النظاف...
2398	0	مطعم تركي طعام جيد و لذيذ لكنه بالنسبة للطفل ح...
440	1	ابراهيم المطعم من تصميمه و نظافته مبين ان جيد ...
508	1	الطعام الأمريكي مكان يصلح لمحبي الأسلوب الأمري...

- Puis l'étape suivant c'est le fine-tuning . Alors en a besoin de data language model avec la fonction TextLMDataBunch.from_df() :

```
ar_tok = Tokenizer(lang='ar')
#data_lm = TextLMDataBunch.from_df(path, train_df=df_train, valid_df=df_val, text_cols=0, label_cols=None, tokenizer=ar_tok, bs=32)
data_lm = TextLMDataBunch.from_df(path, train_df=df_train, valid_df=df_val, text_cols=1, label_cols=None, tokenizer=ar_tok, bs=64, include_bos=False, min_freq=4)
```

```
/usr/local/lib/python3.7/dist-packages/fastai/core.py:302: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple
  return np.array(a, dtype=dtype, **kwargs)
```

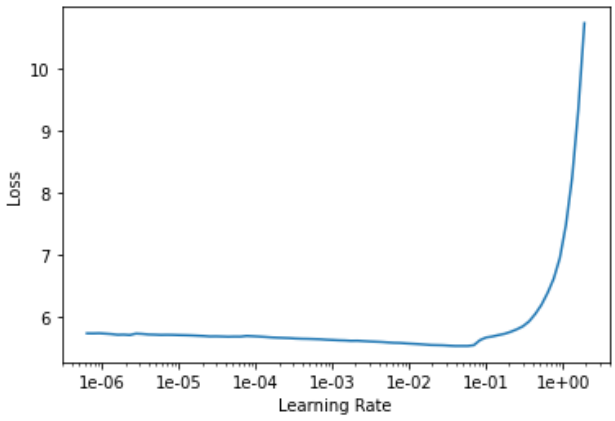
```
data_lm.train_ds.x
```

```
LMTextList (2113 items)
مقارنه xxunk الأسعار المستمر,لا أنصح به سيء والأكل بارد والسعر مرتفع النظافه غير جيدة xxunk أسعار الوجبات من مطاعم الوجبات السريعة المميزة بالنظافة وجودة الأكل يعاب عليه بطء الخدمة xxunk بطء الخدمة مطعم ممتاز
ذات طعم حار و xxunk حال لكن معظم xxunk الدجاج للأطفال على xxunk أخرى xxunk حار يمكن ان تطلب xxunk المطلوب,مطعم تركي طعام جيد و لذيذ لكنه بالنسبة xxunk خارج السعوديه ايضاً المعامله تعاملهم غير جيد xxunk
و أيضاً في الداخل,ابراهيم xxunk لديه مكان للجلوس خارج المطعم xxunk معي قطعة خبز من المطعم و لكن هناك تأخر في تلبية الطلب نظراً xxunk لذيذ جداً لدرجة أنني xxunk على xxunk الشيء اللذيذ جداً هو الخبز الطازج الذي
xxunk حيث يتم xxunk xxunk مكان يصلح لمحبي xxunk بالاسعار و جوده في الماكولات,الطعام xxunk ان يكون xxunk xxunk ان جيد جدا و الاسعار طبعا فيه مرتفعه نظرا للماكولات الجيده xxunk و xxunk المطعم من
xxunk جيدة جدا والسعر مناسب والمكان مناسب للشباب
```

- Ensuite le formateur du modèle qui est responsable à l'ajustement du modèle suivant la nouvelle dataset ça se fait comme illustrer au-dessous :

```
config = awd_lstm_lm_config.copy()
config['n_hid'] = 1150
#####################################################################
learn_lm = language_model_learner(data_lm, AWD_LSTM, drop_mult=0.2, pretrained_fnames=pretrained_fnames,config=config)
```

• Dans le fine-tuning l'information la plus intéressante c'est le taux d'apprentissage donc cette étape consiste à afficher une courbe du perte en fonction du taux d'apprentissage :



• Comme vous voyez ici la perte augmente à partir de $10^{-}2$ c'est pour cela qu'on a décidé de choisir un taux d'apprentissage de $5 * 10^{-}3$ qui se situe avant le début d'augmentation de perte .

• On répète la même étape plusieurs fois mais avec des petits changements , et puis on sauvegarde l'encoder qui est capable d'extraire les caractéristiques des données d'une manière pertinente

```
# learn_lm.save_encoder('att_fine_tuned_encoder')
learn_lm.save_encoder('test3_fine_tuned_encoder')
# 3.935220  4.281686  0.295004
```

• maintenant ce qui reste c'est de faire une petite prédiction comme suivant :

```
learn_lm.predict('الله يبارك في ك+', n_words=30)
```

```
Resultat:

الله يبارك في ك+ الله على أكل عجيب كافيه الزحمة جو لطيف للبيتزا مكان يجمع بين الخصوصية دول لطيف مطعم جميل وراقي جدا من جميع النواحي الطعام نظيف جدا مدهش حلوة من اهم العاملين
```

3.7 Classification

• Premièrement on va crée les données de classification :

```
ar_tok = Tokenizer(lang='ar')
data_clas = TextClasDataBunch.from_df(path, train_df=df_train, valid_df=df_val, text_cols=1,
        label_cols=0, tokenizer=ar_tok, bs=32, vocab=data_lm.train_ds.vocab,
        include_bos=False,
        min_freq=4, num_workers=0)
```

- Puis on crée l'objet learner qui nous permettra d'appliquer cette classification sur nôtre encoder :

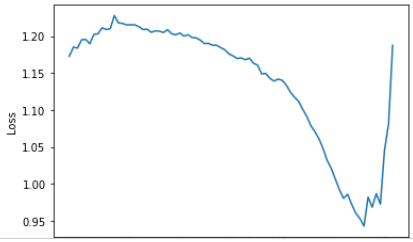
```
learn_clas = text_classifier_learner(data_clas, AWD_LSTM, config=config, drop_mult=0.2, metrics=[accuracy,FBeta(average='weighted')])
learn_clas.load_encoder('/content/drive/MyDrive/hUlMona/models/test3_fine_tuned_encoder')
```

```
Downloading https://s3.amazonaws.com/fast-ai-modelzoo/wt103-fwd.tgz
RNNLearner(data=TextClasDataBunch;

Train: LabelList (2113 items)
x: TextList
xxunk الأسعار المستمر,ولا أنصح به سيء والأكل بارد والسعر مرتفع النظافه غير جيدة xxunk أسعار الوجبات من مطاعم الوجبات السريعة المميزة بالنظافة وجودة الأكل يعاب عليه بطء الخدمة xxunk بطء الخدمة xxunk مطعم ممتاز مقارنه
xxunk حال لكن معظم xxunk الدجاج للأطفال على xxunk أخرى xxunk حار يمكن ان تطلب xxunk المطلوب,مطعم تركي طعام جيد و لذيذ لكنه بالنسبة xxunk خارج السعوديه ايضاً المعاله تعاملهم غير جيد xxunk ذات طعم حار و
xxunk الشيء اللذيذ جداً هو الخبز الطازج الذي و أيضاً في الداخل,ابراهيم xxunk لديه مكان للجلوس خارج المطعم xxunk معي قطعة خبز من المطعم و لكن هناك تأخر في تلبية الطلب نظرا xxunk لذيذ جداً لدرجة أنني xxunk على xxunk
xxunk xxunk حيث يتم xxunk مكان يصلح لمحبي xxunk بالاسعار و جوده في الماكولات,الطعام xxunk ان يكون xxunk ان جيد جدا و الاسعار طبعا فيه مرتفعه نظرا للمأكولات الجيده xxunk و xxunk المطعم من
xxunk xxunk جيدة جدا والسعر مناسب والمكان مناسب للشباب
y: CategoryList
1,-1,0,1,1
```

- Ce qui reste c'est ajuster les poids du modèle avec la technique du perte en fonction du taux d'apprentissage

```
learn_clas.lr_find()
learn_clas.recorder.plot()
```

```
50.00% [1/2 01:29<01:29]
epoch  train_loss  valid_loss  accuracy  f_beta  time
    0    1.117292        #na#     01:29

42.42% [28/66 00:44<01:00 3.3635]
/usr/local/lib/python3.7/dist-packages/fastai/text/data.py:124: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tu
  sort_idx = np.concatenate(np.random.permutation(ck_idx[1:])) if len(ck_idx) > 1 else np.array([],dtype=np.int)
<string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different le
/usr/local/lib/python3.7/dist-packages/fastai/text/data.py:124: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tu
  sort_idx = np.concatenate(np.random.permutation(ck_idx[1:])) if len(ck_idx) > 1 else np.array([],dtype=np.int)
<string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different le
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```



- Voila on peut maintenant faire la prédiction :

```
prediction = learn_clas.predict('المطعم سيء بالمرة الاكل سيء كل شيء سيئ')
if str(prediction[0]) == '1':
  print('it is a positive review')
elif str(prediction[0]) == '-1':
  print('it is a negative review')
elif str(prediction[0]) == '0':
  print("it's not a review")
```

```
it is a negative review
```

Chapitre 4

Résultats & Discussion

Après plusieurs tests,nous pouvons dire que le modèle hULMonA a atteint des performances de pointe sur l'analyse des sentiments,et la reconnaissance des entités nommées pour la langue arabe . Cela ajoute de la vérité à l'hypothèse selon laquelle les modèles de langage pré-entraînés sur une seule langue ne dépassent la performance d'un modèle multilingue ,dans notre cas Arabert . Ce saut de performance a de nombreuses explications.

Tout d'abord, la taille des données est un élément facteur d'amélioration des performances. huLMona utilise autour 24Go de data en comparaison avec le Wikipedia 4.3G utilisé pour le BERT multilingue.

Deuxièmement, la taille du vocabulaire dans le jeux des données utilisé dans le BERT multilingue est de 2000 tokens contre 64000 tokens,taille de vocabulaire utilisée pour développer le modele hULMonA.

Troisièmement, avec le grande taille de données, la distribution de pré-formation a plus de diversité. Quant au quatrième point, la pré-segmentation appliquée avant la tokenisation BERT a amélioré les performances sur tâches de l'analyse de sentiment mais réduisez-le sur la tâche NER.

C'est aussi à noté que lors de l'execution de l'étape du pré-traitement des données nous avons remarqué que les développeurs de ce modèle ont pris en considération la complexité de la langue arabe. Par conséquent,c'en excluant les tokens redondants et inutiles qui viennent avec certains préfixes communs que le vocabulaire augmente en efficacité , et aide le modèle à mieux apprendre afin de réduire la complexité du langage.

Nous croyons que ces facteurs ont aidé le modèle hULMonA à atteindre des résultats de pointe sur la tâche de SA sur un ensembles de données (Dataset) différents.

À cela s'ajoute les résultats obtenus dans la partie de la contribution qui indiquent d'avantage que nous avons obtenu avec un jeux de données ; Dataset RES ; considéré pour l'arabe une meilleur compréhension dans un modèle monolingue tel que hULMonA que d'un modèle de langage général formé sur les crawls de Wikipédia comme le BERT multilingue.

Conclusion et Perspectives

Pour conclure ,nous pouvons dire que le modèle hULMonA définit un nouvel état de l'art pour plusieurs tâches en aval pour la langue arabe. Aussi ce modele est plus petit au niveau de la taille de donnes que le modèle BERT multilingue.

En faisant cette etude comparative entre ces deux modèles , nous sortons en fin de compte que hULMona a bien sa marque de fabrique comme le modèle approprié qui sera utilisé pour servir comme nouvelle référence pour les différentes tâches de la TALN en arabe, et espérons que ce travail servira de tremplin à la construction et l'amélioration des futurs modèles de compréhension de la langue arabe.

Actuellement, nous continuons à contribuer au développement de ce projet par la réalisation de la tache de la classification des crimes à partir des textes la langue arabe avec 3 dataset différents, nous sommes également en discussion avec un doctorant de l'université de Saint Joseph en Liban dans le but d'une future collaboration .

Travaux futurs

- Améliorez l'étude en ajoutant d'autres jeux de données (Dataset) en se basnt sur d'aure dialectes de l'arabe tel que l'arabe des pays du Maghreb (Maroc,Tunisie ,Algerie,Lybie) .

- Avec le modèle hUlMona qui a montré de meilleures métriques, créez un système de questions et réponses pour la langue arabe à domaine ouvert dans lequel le contexte d'une question sera un long document.

- Dans le domaine d'application de l'industrie, un système de chatbot peut être créé qui nous permettra de faire des demandes et de fournir des réponses automatiquement en fonction d'un domaine spécifique.

- Former des modèles avec une meilleure compréhension des différents dialectes de la langue arabe dans différents pays arabes.