# IFT6285 – Projet 1
# Classification

Maxime MONRAT
Juan Felipe DURAN
Nathan MIGEON

Université de Montréal
Automne 2021

# Table des matières

# I – Analyse de Sentiment avec le SST

## I.1 – Objectif

Dans cette tâche nous entraînerons un modèle sur le *Stanford Sentiment Treebank*[1] pour la classification de sentiments. Nous testerons plusieurs approches avant de choisir la meilleure. Nous prendrons aussi le temps d'analyser l'impact de différents méta-paramètres dans une visée d'optimisation.

## I.2 – Présentation du Corpus

Le *Stanford Sentiment Treebank* est basé sur 11 855 phrases extraites d'évaluations de films. Ces phrases ont ensuite été analysées à l'aide du *Stanford Parser*[2] pour produire un ensemble de 215 154 phrases uniques, annotées par 3 juges humains.

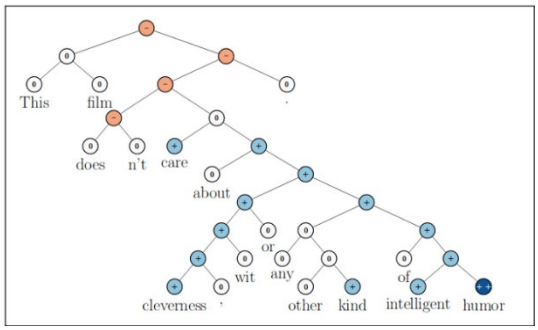


*Figure 1: Exemple d'arbre d'analyse fournie par le SST (source: article original)*

## I.3 – Modèle de Base : Plongements de Mots

Une approche basique du problème de classification de sentiment est de plonger chaque mot $w_i$ de la phrase d'entrée en un vecteur $e_i$ de dimension $n_h$, puis de passer la moyenne de ces plongements dans un unique nœud *softmax*.

Nous utilisons ici les plongements *GloVe*[3] pré-entraînés sur le Common Crawl (1.9 million de tokens, $n_h = 300$)

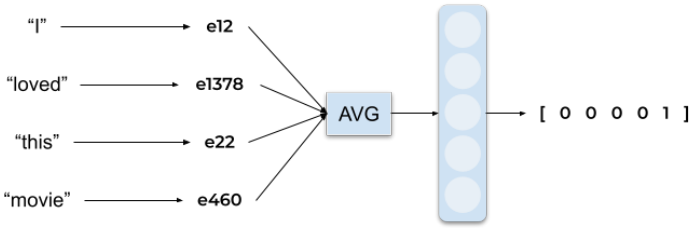


*Figure 2: Modèle 'naïf' de classification de sentiments*

### I.3.a – Formatage des données

Pour notre première approche nous utilisons seulement les données *"brutes"*, c'est-à-dire les phrases racines sans les arbres fournis par le SST. Pour cela nous devons convertir les fichiers fournis par le SST en un format plus facile d'utilisation.

Nous utilisons pour effectuer cette transformation le code fourni par Prashanth Rao dans son exploration du SST, qui utilise la librairie *pytreebank.py*

| *Phrase issue du SST, sans formatage* | *Sortie formatée, sans données d'arborescence* |
|---|---|
| (3 (2 It) (4 (4 (2 's) (4 (3 (2 a) (4 (3 lovely) (2 film))) (3 (2 with) (4 (3 (3 lovely) (2 performances)) (2 (2 by) (2 (2 (2 Buy) (2 and)) (2 Accorsi))))))) (2 .))) | 4 It 's a lovely film with lovely performances by Buy and Accorsi |

*Table 1 Formatage des phrases du corpus d'entraînement*

Par mesure de sécurité, nous devons comparer la distribution de nos données sur les deux ensembles. Ici, pas de problème : les ensembles possèdent bien une distribution similaire ; on peut tout de même noter que nous avons un nombre supérieur d'exemples de classe 1 et 3.

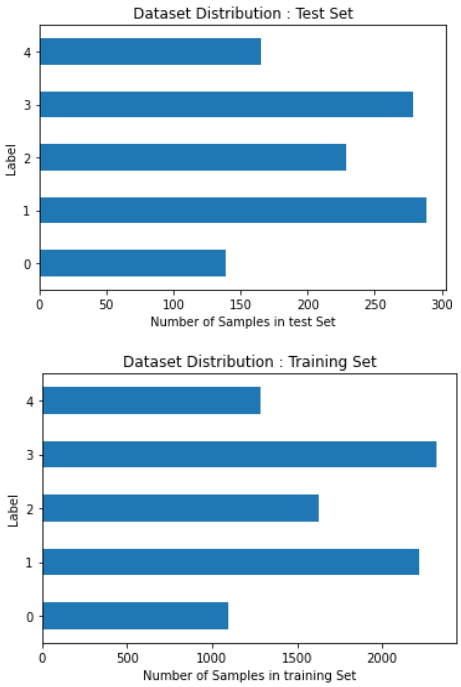


*Figure 3 Distribution de probabilités pour les phrases racines du SST*

### I.3.b – Performances

Nous obtenons une exactitude[4] de **50.2%** sur le train set, et **42.7%** sur le test set. Bien que ces valeurs ne soient pas très bonnes, elles sont déjà supérieures au hasard qui aurait été de 20%. Le diagramme de confusion ci-contre démontre bien la pauvre performance. Notons que le diagramme montre de meilleures performances sur les phrases classées 1 et 3 ; ce n'est en réalité pas très étonnant puisque ces résultats représentent bien la distribution de probabilités telle qu'exposée plus haut.

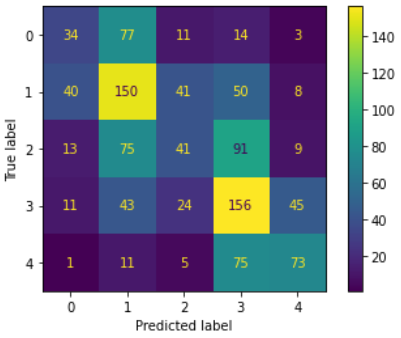


*Figure 4: Matrice de confusion du modèle naïf pour une classification fine*

[1] Socher et al., 2013, *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*

[2] Klein et Manning, 2003, *Fast Exact Inference with a Factored Model for Natural Language Parsing*. Peut être trouvé sur le site web NLP de Stanford.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation

[4] Le terme *exactitude* fait ici référence à la fréquence exemples classés correctement sur l'ensemble des données

| | |
|---|---|
| this is the best movie I have ever seen | 4 |
| I am not a fan of this genre | 1 |
| I did not dislike it | 0 |
| I have seen better movies | 4 |

*Table 2: Exemples de phrases classées par notre premier modèle*

On remarque, sans surprise, que ce modèle simple est incapable de traiter les négations : il fait la moyenne de chaque plongement de mot dans la phrase indépendamment de l'ordre.

### I.4 – Modèle Amélioré : RNN Bi-LSTM

Pour notre seconde approche nous avons utilisé un modèle de réseau neuronal récurrent bidirectionnel (Bi-LSTM). L'idée est de résoudre une partie des problèmes illustrés précédemment, en particulier les négations, puisque les unités LSTM permettent de « garder en mémoire » l'information des mots précédents

#### I.4.a – Présentation du Modèle

Nous avons monté notre modèle en utilisant la bibliothèque Keras et TensorFlow. Pour pouvoir utiliser les mini-batchs pour l'entraînement, toutes les données d'entrée doivent avoir la même taille. Notre solution est de prendre la phrase la plus longue comme taille maximale, puis de compléter toutes les phrases plus courtes par un vecteur nul.

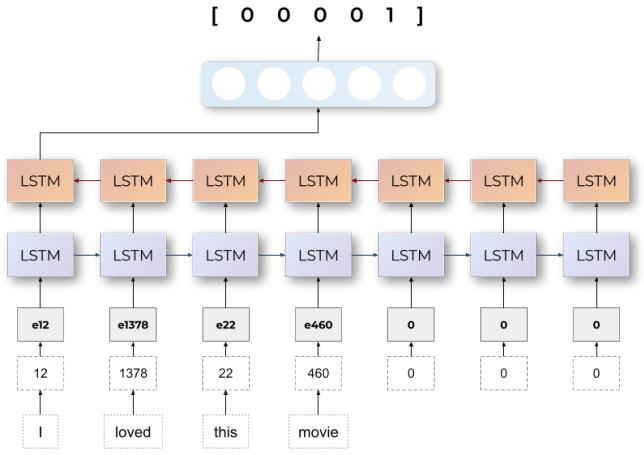


*Figure 5: Architecture du Modèle Bi-LSTM*

#### I.4.b – Performances

La première version de ce modèle démontre une belle exactitude sur le set d'entraînement (**97.01%**), mais une grande variance avec le set de test (**42.4%**). Afin de réduire cet effet d'*overfitting*, nous avons procédé à deux techniques différentes :

1. Augmenter les données
   a. Le SST contient une classification sur le sentiment pour chaque nœud de l'arbre syntaxique. En découpant chaque échantillon en ses sous-ensembles grammaticaux, on passe de 11 855 échantillons à 215 154!
2. Augmenter la régularisation. Nous avons utilisé deux types de régularisation sur nos données : un *dropout* de 50% après chaque couche LSTM, puis une régularisation de type L2 (RIDGE).

Suite à ces changements nous obtenons finalement une exactitude de **95.6%** sur l'entraînement, et **80.1%** sur le test. Bien que l'on ait encore un petit peu d'*overfitting*, ces valeurs restent proches de celles obtenues par Socher et al. dans l'article original (79.0% pour un RNN, à noter que nous n'avons ici pas utilisé le même set de test puisque nous avons travaillé avec le set de développement)

### I.5 – Bilan

Notre modèle final semble « bien » performer pour l'analyse multi-classe de sentiments sur les données du SST. Une analyse plus profonde de ces résultats montre que le modèle est en réalité meilleur pour les phrases à tendance positive (voir la matrice de confusion normalisée suivante). Le système semble également très performant pour les phrases neutres, mais ce résultat est biaisé par l'abondance de données classées « neutres » dans nos datasets.

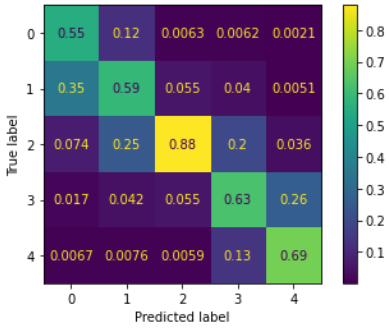


*Figure 6: Matrice de confusion normalisée pour notre modèle Bi-LSTM en classification fine*

Un autre aspect à prendre en compte est que lorsque testé seulement sur les phrases *racines*, notre modèle n'est précis qu'à 43.8%, ce qui est très proche du modèle « naïf »! En fonction de l'utilisation désirée, il est donc important de prendre en compte la taille des échantillons, puisqu'un modèle simple de plongement de document est bien moins gourmand en ressources qu'un RNN tel que celui que nous utilisons.

Tous ces tests étaient effectués sur une classification fine des sentiments et ont montré qu'en *général*, notre modèle BiLSTM est plus performant qu'une moyenne sur les plongements de mots. Pour une classification binaire en revanche, on obtient une exactitude de **85.9%** pour le modèle naïf, contre 86.2% pour le modèle BiLSTM! Ces résultats démontrent, une nouvelle fois, que la taille du modèle doit être adaptée à l'utilisation prévue de nos données afin d'éviter le gaspillage de ressources.

| **Phrase** | **Y** | **Ŷ** |
|---|---|---|
| A rarity among recent Iranian films: It 's a comedy full of gentle humor that chides the absurdity of its protagonist 's plight. | 4 | 1 |
| A sober and affecting chronicle of the leveling effect of loss. | 2 | 1 |
| A celebration of quirkiness, eccentricity, and certain individuals ' tendency to let it all hang out, and damn the consequences. | 3 | 1 |
| Writer/director Joe Carnahan 's grimy crime drama is a manual of precinct cliche, but it moves fast enough to cover its clunky dialogue and lapses in logic. | 3 | 1 |
| One of the best films of the year with its exploration of the obstacles to happiness faced by five contemporary individuals... a psychological masterpiece. | 4 | 1 |
| Not far beneath the surface, this reconfigured tale asks disturbing questions about those things we expect from military epics. | 3 | 1 |
| This surreal Gilliam-esque film is also a troubling interpretation of Ecclesiastes. | 3 | 1 |
| A quiet treasure -- a film to be savored. | 4 | 1 |
| Beautifully observed, miraculously unsentimental comedy-drama. | 4 | 1 |
| We root for (Clara and Paul) , even like them, though perhaps it's an emotion closer to pity. | 3 | 1 |

*Table 3: Exemples de phrases mal classées*

Une rapide analyse des phrases classées très négatives par notre modèle Bi-LSTM montre que la majorité des erreurs viennent de phrases exprimant du sarcasme ou de l'ironie. Ceci n'est pas surprenant, puisque la compréhension de l'ironie demande une haute compréhension du contexte verbal et social.

# II – Identification de Paraphrases avec *Quora Question Pairs*

## II.1 – Analyse des données

La tache QQP est une tâche binaire qui consiste à déterminer si deux questions posées sur Quora sont des paraphrases. Après avoir analysé les données et révisé de la littérature pertinente[5] nous avons fait les remarques suivantes :

- 63.08% des données ont un étiquetage négatif (0).
- 36.92% des donnes ont un étiquetage positif (1).
- 6228 questions possèdent des caractères non-ASCII
- 79.22% des questions apparaissent plus qu'une fois (une question apparait 158 fois).

## II.2 – Baseline : performance, description et métrique d'évaluation

Nous avons décidé de prendre comme baseline le « doc2vec » de la librairie *Gensim*[6].

Doc2vec est un algorithme d'apprentissage automatique non supervisé utilisé pour convertir un document en vecteur. C'est un algorithme qui utilise la méthode « continuous Bag-of-Words (CBOW) » pour estimer le prochain mot d'une phrase, base sur le contexte de la phrase[7]. (Fahad,2021)

Après avoir mis en place le programme basé sur cette implémentation et entrainé un modèle doc2vec avec les données tokenisées mises en minuscules, nous obtenons une exactitude de **58.26%**. La métrique d'évaluation de ce modèle consiste à observer la distance cosinus entre les plongements des deux questions. Si la distance est assez petite, les questions sont considérées comme des paraphrases.

Nous avons aussi utilisé un modèle de perplexité basé sur NLTK. La perplexité est une mesure de la probabilité qu'une phrase soit grammaticale ou non, basée sur un modèle pré-entrainé. Notre hypothèse est que si deux questions possèdent une perplexité similaire, elles doivent également véhiculer un sens similaire. Avec ce modèle, nous obtenons une exactitude de **59.864%**.

## II.3 – Optimisation du Baseline.

Pour le modèle doc2vec, nous avons regardé les hyperparamètres **vector_size** (la longueur du vecteur), **min_count (**mots avec une fréquence plus petite que min_count sont ignorés) et **epochs** (nombre d'itérations sur la base de données).

| Vector_size | Min_count | Epochs | Accuracy(%) |
|---|---|---|---|
| 10 | 2 | 40 | 60.799 |
| 20 | 2 | 40 | 60.845 |
| 30 | 2 | 40 | 60.858 |
| 40 | 2 | 40 | 60.824 |
| 50 | 2 | 40 | 60.831 |
| 20 | 1 | 40 | 60.86 |
| 20 | 2 | 40 | 60.872 |
| 20 | 5 | 40 | 60.880 |
| 20 | 10 | 40 | 60.843 |
| 20 | 2 | 20 | 60.835 |
| 20 | 2 | 40 | 60.853 |
| 20 | 2 | 60 | 60.858 |
| 20 | 2 | 80 | 60.810 |
| 20 | 2 | 100 | 60.802 |

*Table 4: Réglage des paramètres du modèle doc2vec*

Après voir fait quelques tests, nous avons trouvé comme valeurs optimales **30** pour **vector_size**, **5** pour **min_count** et **60** pour **epochs**.

Pour le modèle de perplexité, Nous avons comparé des modèles *everygrams* qui prennent en considération soit exclusivement des unigrammes, soit des unigrammes et des bigrammes ou alors des unigrammes, bigrammes et trigrammes. Nous avons ensuite cherché la différence en perplexité maximale pour déterminer si deux questions ont le même sens.

| Max perp | Min perp | Ngram | Perp diff | Accuracy(%) |
|---|---|---|---|---|
| 2510929 | 11.909 | 1 | 0.01 | 62.573 |
| 1355.180 | 16.042 | 2 | 0.01 | 62.341 |
| 134.378 | 7.91 | 3 | 0.01 | 62.192 |
| 1355.181 | 16.042 | 2 | 5 | **62.697** |
| 1355.181 | 16.042 | 2 | 2 | 62.684 |
| 1355.181 | 16.042 | 2 | 1 | 62.629 |
| 1355.181 | 16.042 | 2 | 0.1 | 62.367 |
| 1355.181 | 16.042 | 2 | 0.01 | 62.341 |
| 1355.181 | 16.042 | 2 | 0.001 | 62.345 |

Réglage des paramètres du modèle de perplexité

Nous avons trouvé que le modèle ngram=2 avec une différence de perplexité de 5 fonctionnait le mieux. De plus, nous avons considéré les prétraitements suivants :

- **Lem** : lemmatisation des phrases.
- **Rem sw** : retirer les « stop-words ».

Nous observons par ailleurs les propriétés suivantes pour tenter de déterminer si deux phrases sont les mêmes :

- **Len**: les questions ont le même nombre de mots
- **Rem last** : les questions ont le même dernier mot.
- **Contains 2** : les questions ont deux mots en commun.
- **Contains 3** : les questions ont trois mots en commun.

**Cos. Dis.** : distance entre deux vecteurs de phrase pour déterminer si elles ont le même sens (s'applique seulement au modèle doc2vec).

| Lem | Rem sw | Len | Rem last | Contains 2 | Contains 3 | Cos. dis. | Acc(%) |
|---|---|---|---|---|---|---|---|
| **False** | False | False | False | False | False | 1 | 41.132 |
| **False** | False | False | False | False | False | 0.1 | 61.968 |
| **False** | False | False | False | False | False | 0.01 | 62.632 |
| **False** | False | False | False | False | False | 0.001 | 62.702 |
| **False** | False | False | False | False | False | 0.0001 | 62.702 |
| **True** | False | False | False | False | False | 0.1 | 62.130 |
| **False** | True | False | False | False | False | 0.1 | 62.868 |
| **False** | False | True | False | False | False | 0.1 | 61.184 |
| **False** | False | False | True | False | False | 0.1 | 38.33 |
| **False** | False | False | False | True | False | 0.1 | 41.904 |
| **False** | False | False | False | False | True | 0.1 | 46.998 |

*Table 5: Prétraitement pour doc2vec*

| Lem | Rem sw | Len | Rem last | Contains 2 | Contains 3 | Acc(%) |
|---|---|---|---|---|---|---|
| **True** | False | False | False | False | False | 62.606 |
| **False** | True | False | False | False | False | 64.652 |
| **False** | False | True | False | False | False | 60.826 |
| **False** | False | False | True | False | False | 38.3 |
| **False** | False | False | False | True | False | 41.892 |
| **False** | False | False | False | False | True | 46.97 |

*Table 6: Prétraitement pour le modèle de perplexité*

Nous pouvons remarquer que seulement la lemmatisation et le retrait des *stop-words* améliorent le modèle. Pour la distance cosinus, une distance plus petite que **0.001** semble indiquer que les questions sont assez similaires.

## II.4 – Identification du meilleur modèle

Suite à notre analyse et après avoir déterminé les paramètres optimaux nous obtenons les résultats suivants pour les deux modèles:

[5] Sharma et al., 2019 *Natural Language Understanding with the Quora Question Pairs Dataset*

[6] Gensim, 2021, *Doc2Vec implementation*

[7] Medium, 2021, *Doc2Vec — Computing Similarity between Documents*

| Optimal perplexity model | Optimal doc2vec model |
|---|---|
| **65.817** | 62.934 |

Notre meilleur modèle est donc le modèle à comparaison de perplexité.

### II.5 – Test sur les 50 données

Après avoir testé le modèle optimal sur des données faites à la main, nous obtenons un résultat de **48%.**

Nous pensons qu'un résultat si faible peut s'expliquer par le fait que notre modèle étant entraîné sur les questions de la base de données QQS, les questions faites à la main possèdent des mots et des formes différentes, rendant les résultats plus improbables et erratiques.

Par curiosité nous avons alors essayé le modèle doc2vec, et avons obtenu un résultat de **82%.** Clairement le modèle doc2vec semble donc plus généralisable.

## III – Acceptabilité linguistique avec CoLA

### III.1 – Objectif

Cette tâche consiste à évaluer si une phrase est acceptable, c'est-à-dire grammaticalement correcte ou non, en utilisant le *Corpus of Linguistic Acceptability* (CoLA). Afin d'y parvenir, nous avons essayés différentes approches, basées sur des réseaux de neurones utilisant des plongements de mots.

### III.2 – Présentation des données

L'ensemble des données mis à disposition pour effectuer cette tâche est composé de phrases attachées à un label indiquant si la phrase est acceptable ou non. Il s'agit donc d'une classification binaire (0 = non-acceptable et 1 = acceptable). Voici un exemple de données présentes dans le corpus d'entraînement :

| Label | Phrase |
|---|---|
| 1 | I had the strangest feeling that I knew you. |
| 0 | As you eat the most, you want the least. |
| 1 | Mary listens to the Grateful Dead, she gets depressed. |
| 0 | The box contained the ball from the tree. |
| 0 | We wanted to invite someone, but we couldn't decide who to. |

*Table 7: Exemple d'échantillons du corpus d'entraînement de CoLA*

### III.3 – Distribution des ensembles de données

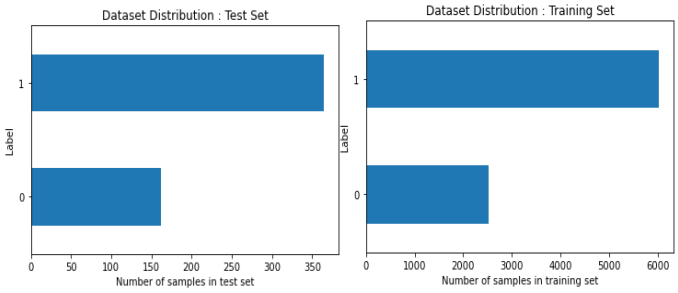


*Figure 7: Distribution des données sur les ensembles de test et d'entraînement*

Une brève analyse montre que la distribution des jeux de test et d'entraînement est bien similaire. On remarque cependant que les deux ensembles de données sont constitués de beaucoup de phrases acceptables. Reste à voir si ce déséquilibre des classes a une influence sur l'entraînement et la prédiction de nouvelles données.

### III.4 – Première approche : Modèle Séquentiel simple

Dans un premier temps, nous avons construit un modèle neuronal séquentiel afin de faire d'effectuer quelques essais. Pour que le modèle puisse traiter les données, il a fallu transformer les phrases en séquences.

Après avoir essayé différentes valeurs d'hyperparamètres dont le nombre d'*epochs* et la taille des plongements de mots, nous avons obtenu comme meilleurs résultats une exactitude de **60.15%** lors de l'entraînement.

Au regard de la matrice de confusion ci-contre, on se rend compte que le modèle semble se tromper beaucoup plus sur les phrases agrammaticales (seulement 33% sont bien classifiées). En revanche, il a bien classifié 70% des phrases acceptables.

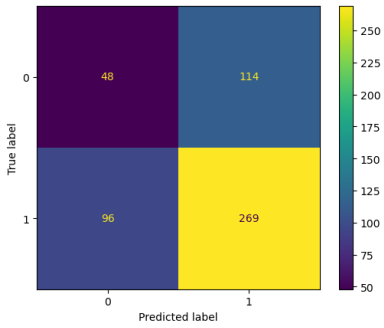


*Figure 8: Matrice de confusion pour notre modèle baseline*

### III.5 – Ajout d'une couche de convolution et d'un max pooling

La deuxième approche consistait à rajouter une couche de convolution et une couche de max pooling à notre premier modèle séquentiel, des procédés permettant de réduire la taille de l'input.

En effectuant quelques nouveaux entraînements en modifiant les hyperparamètres, les résultats se sont légèrement améliorés puisque le modèle atteint désormais **61.67%** d'exactitude. C'est toujours peu, mais cela reste une amélioration à prendre en compte.

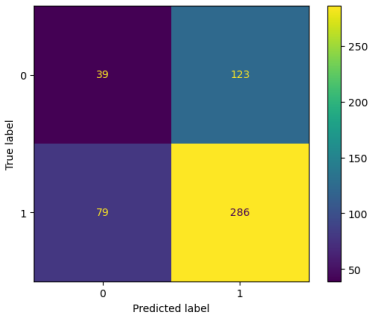


*Figure 9: Matrice de confusion suite à l'étape de convolution*

Malheureusement, la matrice de confusion montre le même problème que sur le modèle précédent : la classification est toujours sensiblement plus efficace pour les phrases acceptables.

### III.6 – Ajout d'une 2e couche LSTM

Suite aux résultats des 2 modèles précédents, l'idée était de rajouter une 2e couche LSTM au modèle pour voir ce que ça allait donner. Cette fois-ci, les résultats ont été encore meilleurs avec une précision de **63.19%**. On est malheureusement encore loin des meilleurs résultats que l'on peut trouver dans des benchmarks, mais cela reste une amélioration.

Pour ce modèle, la matrice de confusion souligne le même problème que précédemment, c'est-à-dire le rappel pour les phrases agrammaticales qui est encore très faible (32%).

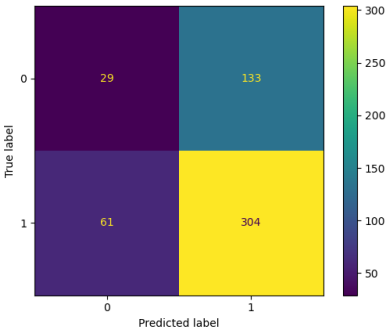


*Figure 10: Matrice de confusion suite à l'ajout de LSTM*

### III.7 – Utilisation du meilleur modèle sur de nouvelles données

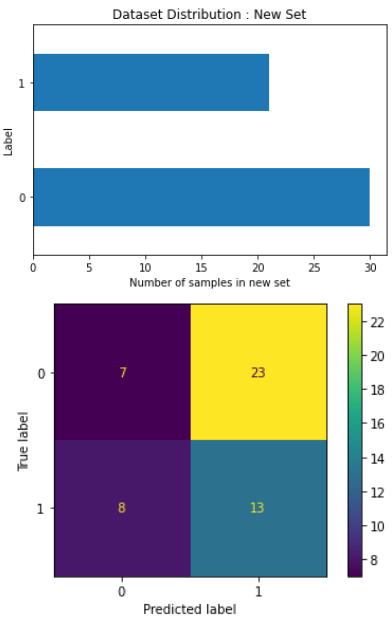


*Figure 11: Distribution de nos données et matrice de confusion correspondante*

Après avoir effectué ces différents tests, nous avons créé nous-mêmes de nouvelles données sur lesquelles évaluer notre meilleur modèle créé pour la tâche CoLA. Nous avons donc généré un fichier csv avec 50 phrases et leur classe. Pour vérifier l'hypothèse selon laquelle le modèle est beaucoup moins performant pour classifier correctement des phrases agrammaticales, nous avons choisi de mettre plus de phrases incorrectes que de phrases acceptables.

Et comme on pouvait l'imaginer, les résultats sont beaucoup moins bons avec ce déséquilibre car on obtient une précision de seulement 39.22%.

### III.8 – Conclusion sur la tâche CoLA

Ces résultats sont finalement peu convaincants mais si on regarde un benchmark pour la tâche CoLA, on se rend compte que c'est une tâche qui n'est pas encore dotée de très bons modèles car elle est assez difficile à traiter (les meilleurs scores sont aux alentours des 70 - 75% la plupart du temps, ce qui est peu comparé aux tâches comme SST par exemple). Cependant, dans notre cas, on a pu voir que le modèle était très inefficace pour les phrases agrammaticales.

## Bibliographie


- Socher et al., 2013, **Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank**
- Klein et Manning, 2003, **Fast Exact Inference with a Factored Model for Natural Language Parsing**
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. **GloVe: Global Vectors for Word Representation**
- NLTK project, 2021, **nltk perplexity implementation**
- Kim, Towards Data Science, **2018, Perplexity Intuition (and its derivation)**
- Shperber, Medium, 2018, **A Gentle Introduction to Doc2Vec**
- Fahad, Medium, 2021, **Doc2Vec — Computing Similarity between Documents**
- Warstadt, A. Singh, Samuel R. Bowman, 2019, **Neural Network Acceptability Judgements**
- J. Brownlee, Machine Learning Mastery, 2016, **Sequence classification with LSTM RNN**
- Sharma et al., 2019, **Natural Language Understanding with the Quora Question Pairs Dataset**
- Radim Rehurek, Gensim, 2021, **Doc2Vec implementation**