Prédiction du numéro d'épisode à partir de son résumé
The Big Bang Theory subtitles corpus

BELOUADAH Eden BOUHAHA Mariem

8 novembre 2017

1 Introduction

Le développement des sciences des données et de l'apprentissage automatique a aidé dans le développement des outils d'intelligence artificielle notamment celles du traitement automatique du langage naturel. Nous présentons dans ce projet un de ces outils qui permet de montrer l'importance de l'indexation dans un modèle de recherche d'information.

Le but du projet est donc de prédire le numéro d'un épisode et sa saison à partir de son résumé. Nous mettons en pratique nos connaissances du modèle vectoriel sur les cinq premières saisons de la série télévisée *The Big Bang Theory*.

2 Etapes de travail et du raisonnement

2.1 Structure de données

Afin de mettre en pratique le système, il faut charger le contenu des fichiers d'épisodes dans la mémoire centrale. Alors, chaque saison est représentée par une liste où chaque élément est un épisode.

Toutes les saisons sont regroupées dans une liste (de listes) appelée *Corpus*, qui va être par la suite la structure de données principale sur laquelle on travaille.

2.2 Prétraitement des données

Avant d'entraîner un modèle sur nos données, il faut d'abord passer par une étape de prétraitement qui sert à nettoyer le corpus des sous titres. Après avoir récupéré le contenu des fichiers, l'étape de prétraitement est basée sur :

— L'élimination des balises (Ex. : -=http ://sfile. ydy. com=- proudly presents) .

— L'élimination des URL(Ex. : Downloaded From www.AllSubs.org) .

— L'élimination des indicateurs de temps(Ex. : 00 :20 :00,980 –> 00 :20 :04,600).

Le fichier en entrée passe de cette forme :

```
0
00:00:01,000 --> 00:00:04,000
Downloaded From www.AllSubs.org

1
00:00:01,961 --> 00:00:03,293
I just want you both to know,

2
00:00:03,503 --> 00:00:06,630
when I publish my findings,
I won't forget your contributions.

3
00:00:07,129 --> 00:00:08,440
- Great.
- Thanks.

```

Figure 1. L'épisode avant prétraitement

A celle là :

```
I just want you both to know, when I publish my findings, I won't forget your
contributions. - Great. - Thanks. Of course, I can't mention you in my Nobel acceptance
speech, but when I get around to writing my memoirs, you can expect a very effusive
footnote and perhaps a signed copy. We have to tell him. Tell me what? Damn his Vulcan
hearing. You fellows are planning a party for me, aren't you? Okay, Sheldon, sit down. If
there's going to be a theme, I should let you know that I don't care for luau, toga or
"under the sea." Yeah, we'll that in mind. Look... We need to talk to you about something
that happened at the North Pole. If this is about the night the heat went out, there's
nothing to be embarrassed about. - It's not about that. - We agreed to never speak of it
again. So we slept together naked. It was only to our core body temperatures from
plummeting. He's speaking about it. For me, it was a bonding moment. Sheldon, you remember
the first few weeks we were looking for magnetic monopoles and not finding anything and
you were acting like an obnoxious, giant dictator? We were gonna be gentle with him.
That's why I added the "tator." And when we finally got our first positive data, you were
so happy. In the world of emoticons, I was colon, capital "D." Well, in actuality, what
```

Figure 2. L'épisode après prétraitement

L'élimination des mots vides (and, or, the, a...) se fait au fur et à mesure du traitement du texte.

2.3 Processus de calcul

Après avoir récupéré le corpus, nous y faisons entraîner le modèle vectoriel qui utilise la mesure TF-IDF pour la pondération des poids :

$$w_{t,d} = tf_{t,d} \times idf_t = tf_{t,d} \times log_{10} \frac{N}{df_t}$$

où :

— $w_{t,d}$ est le poids du terme t dans l'épisode d,

— $tf_{t,d}$ est la fréquence du terme t dans l'épisode d,

— N est le nombre d'épisodes dans tous le corpus,

— df_t est le nombre d'épisodes qui contiennent le terme t.

Ce processus nous retourne la matrice de pondération TF-IDF qui donne pour chaque terme dans le corpus, son poids dans chaque épisode du corpus.

Une fois tout est prêt, il ne reste que récupérer le résumé entré par l'utilisateur. Puis calculer la similarité entre le résumé et chaque épisode. Nous faisons retourner l'épisode avec la plus grande similarité.

La mesure de similarité utilisée est celle du cosinus :

$$sim(\overrightarrow{q}, \overrightarrow{d}) = \frac{\sum_{i=1}^{n} w_{i,d} \times w_{i,d}}{\sqrt{\sum w_{i,q}^2} \times \sqrt{\sum w_{i,d}^2}}$$

3 Interface du système

Afin de faciliter l'utilisation du système, nous avons développé une interface graphique simple qui sert comme intermédiaire entre le système et l'utilisateur.

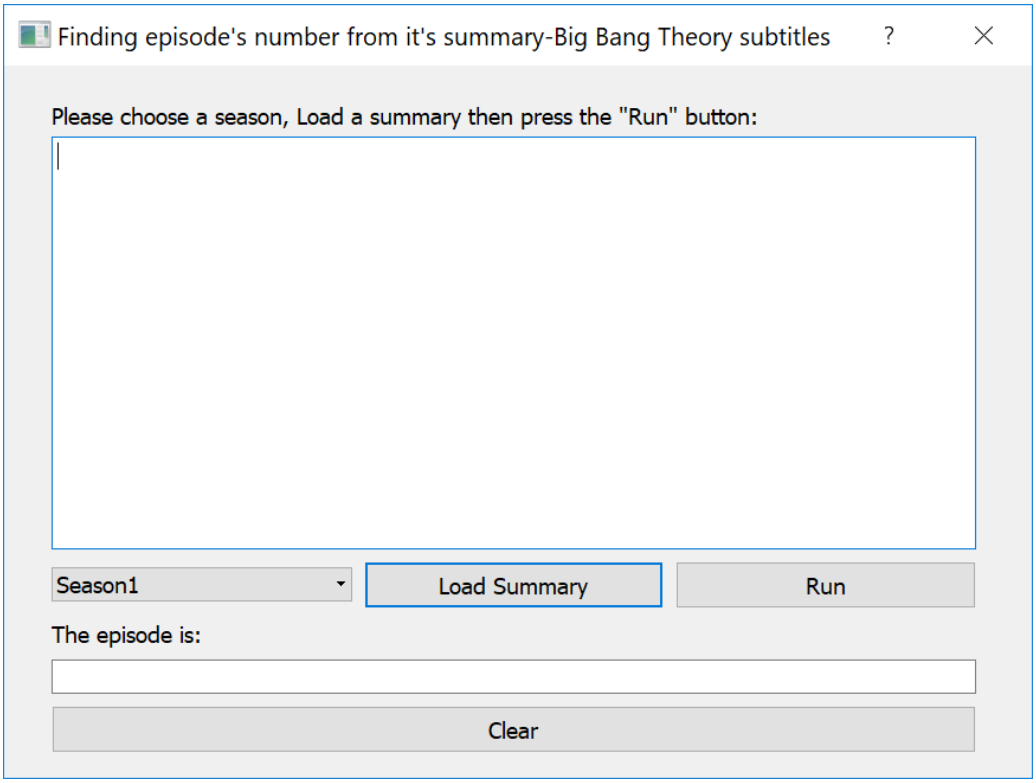
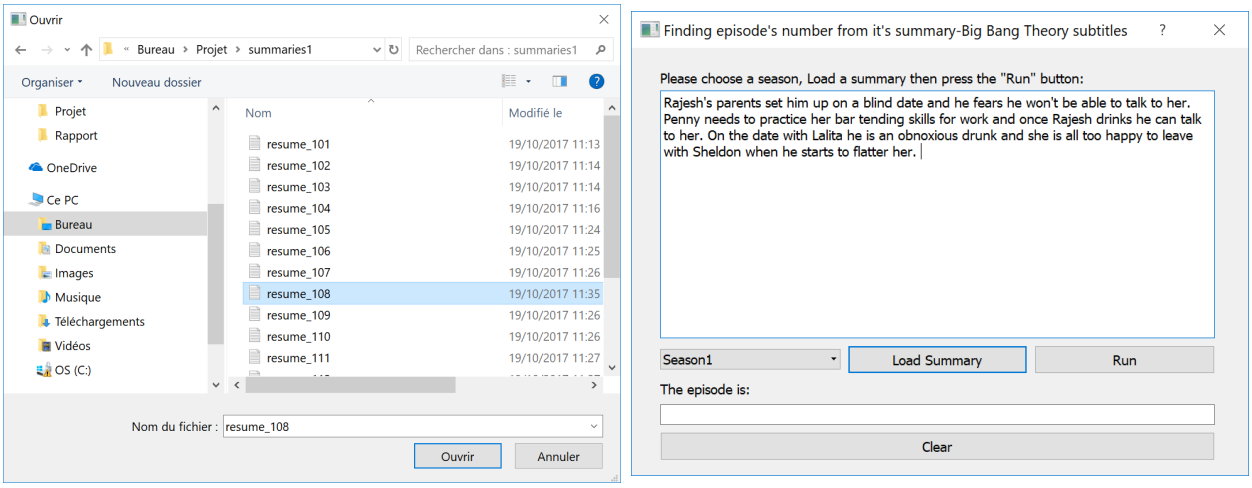


Figure 3. IHM du système

Il est possible de charger un résumé à partir d'un fichier ou bien tout simplement le faire taper dans la zone de texte correspondante.



(a) Choisir un résumé *(b) Charger le résumé*

Figure 4. Choisir et charger un résumé

L'utilisateur a le choix de chercher dans une saison précise comme il a le choix de chercher dans toutes les saisons.

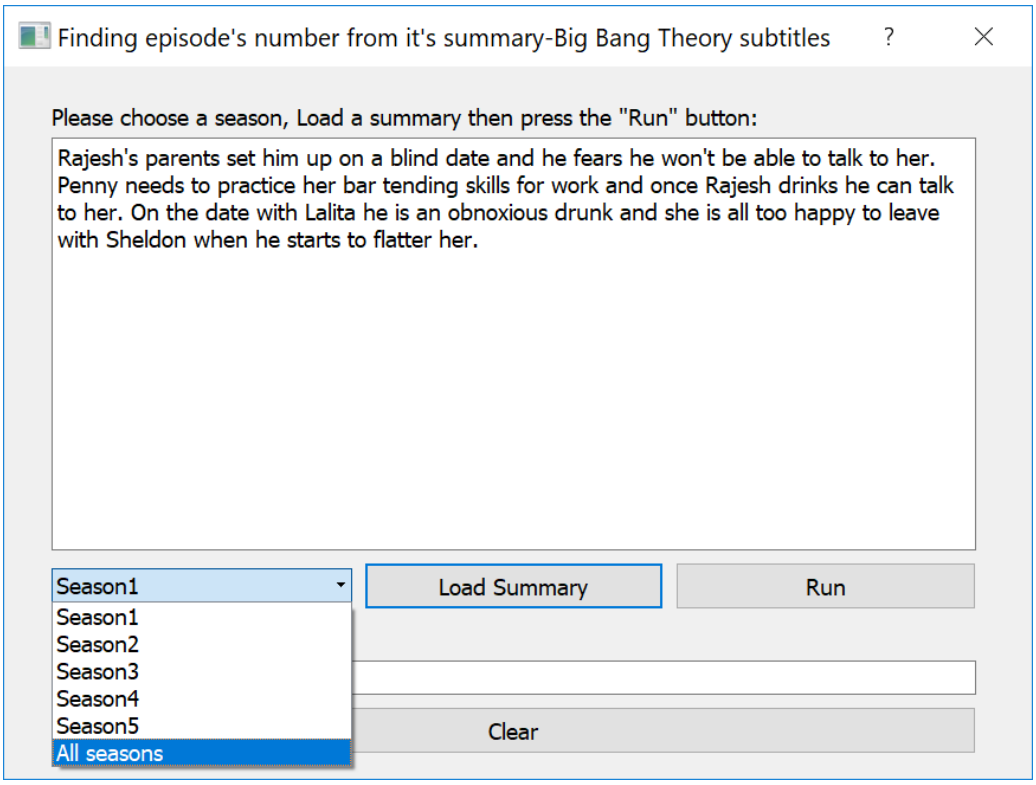


Figure 5. Choisir une ou toutes les saisons

Dans les deux cas, le système, après que l'utilisateur appuie sur le bouton *Run*, retourne le numéro de la saison et le numéro de l'épisode correspondants au résumé entré par l'utilisateur.

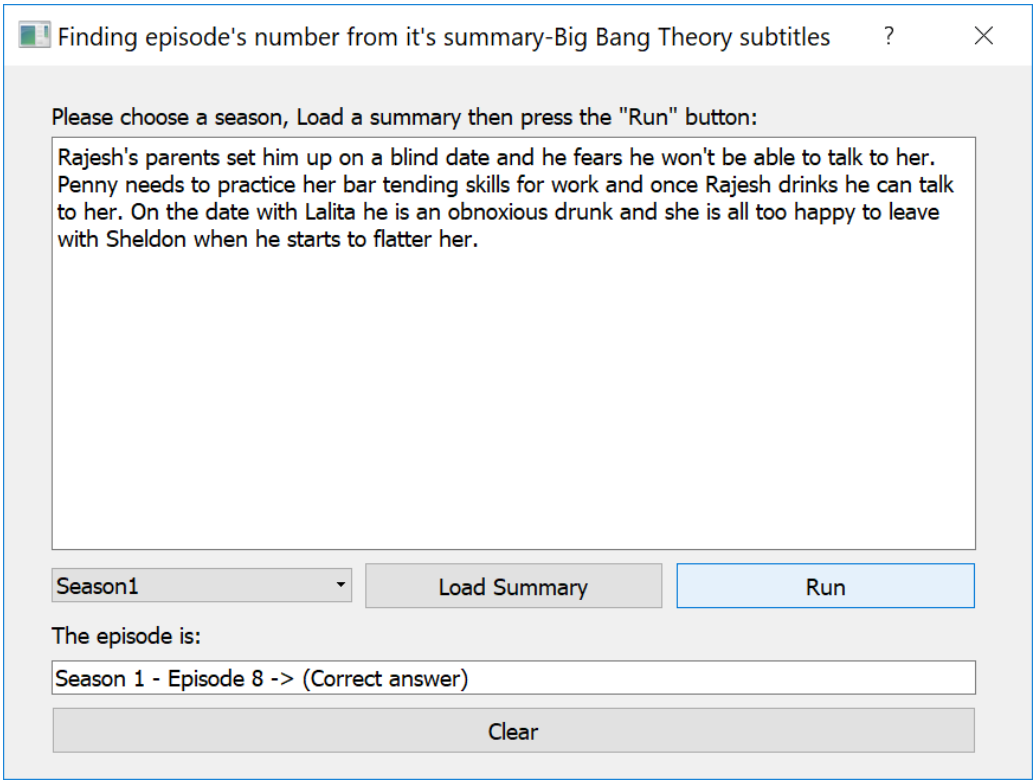


Figure 6. Inférence du modèle

Si le résumé a été chargé à partir d'un fichier, alors le système pourra savoir le vrai numéro de l'épisode à partir du nom du fichier et pourra s'évaluer en indiquant s'il a bien répondu ou non sur la requête de l'utilisateur. Dans le cas où il répond correctement, il affiche le message *Correct answer*, sinon il affiche *Wrong answer*.

4 Evaluation du système

Le script « *evaluation.py* » permet de tester les performances de notre système sur tous les résumés de toutes les saisons possibles. Ainsi, il calcule le taux de réussite. Avec la configuration expliquée ci-haut, le taux de réussite de notre système pour tous les résumés des épisodes de toutes les saisons est de 89.19%. Nous présentons dans ce tableau le taux de réussite que nous avons obtenu pour les différentes tailles du corpus (en termes de saisons) :

Nombre de saisons	1	2	3	4	5
Taux de réussite	100%	97.5%	92.06%	89.66%	89.19%

Tableau 1. Taux de réussite pour chaque taille de corpus

5 Améliorations et essais

Nous avons essayé d'appliquer une lematisation sur le corpus avant d'entraîner le modèle vectoriel, mais ça n'a pas amélioré les résultats. Le taux de succès sur l'ensemble de toutes les saisons a donné 85.59%. La même chose avec la radicalisation, le taux de réussite est de 87%.

Un autre essai était d'utiliser la méthode *word2vec* pour ne garder que les termes ayant une signification dans l'épisode. Le taux de réussite est devenu 88.17%.

Nous supposons donc que la méthode de découpage en mots utilisée par le modèle vectoriel est assez acceptable.

La difficulté que nous avons rencontré est liée à la structure des textes. En effet, les sous titres contiennent plus d'information non utile que d'information utile. De plus, certaines phrases ne portent pas de sens au texte car elles sont constituées que d'un seul mot (par exemple *Okay*) et il y'en a même d'autres qui contiennent des mots abrégés, hors contexte... etc.

L'autre difficulté c'est que des fois, on trouve les acteurs en train de parler sur ce qui s'est passé dans les épisodes précédents, ce qui va introduire certainement des chevauchements dans les mots clefs des épisodes et donc rendre la tâche de prédiction plus difficile.

Une amélioration possible serait de chercher un modèle mieux adapté à ce genre de données textuelles et qui permet de donner une présentation consistante et unique pour chaque épisode.

6 Conclusion

Ce projet était une chance pour nous de mettre en pratique les connaissances acquises sur le modèle vectoriel. Ce dernier présente des avantages et des inconvénients. En effet, il est facile à comprendre et à implémenter, il utilise la pondération des termes qui est une information plus réaliste que la fréquence des termes et enfin la similarité permet d'ordonner les épisodes par ordre de pertinence. Cependant, ce modèle suppose que les termes du documents sont indépendants et ne présente pas une meilleure expressivité des requêtes. Le modèle vectoriel reste un outil très fort qui peut servir à la base de construction des moteurs de recherche sophistiqués.

7 Références

— Les résumés ont été téléchargé à partir de (*http ://bigbangtheory.wikia.com/wiki/Main_Page*).

— Les sous titres ont été téléchargé à partir de (*http ://www.tvsubtitles.net*).

8 ANNEXE : Répartition du travail

Toutes les parties du projet ont été faites ensemble par le binôme. Quelque soit pour le nettoyage, inférence, collecte des données, interface graphique, tests et rapport, nous avons contribué ensemble avec la même charge de travail pour faire marcher le système.

— Rapport : Eden 50%, Mariem 50%.

— Collecte de données : Mariem 70%, Eden 30%.

— Interface graphique : Eden 70%, Mariem 30%.

— Nettoyage : Eden 50%, Mariem 50%.

— Inférence : Mariem (TF/IDF, racinisation, lemmatisation), Eden (word2vec)

— Test : on les a fait ensemble lors des séances du TP.